МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проєктами

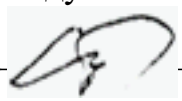
Кафедра програмного забезпечення автоматизованих систем

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма «Інженерія програмного забезпечення»

«Допущений до захисту»

Завідувач кафедри

_  _ проф. Приходько С.Б.

«___» _____ 2025 р.

КВАЛІФІКАЦІЙНАРОБОТА

на здобуття ступеня вищої освіти «бакалавр»

на тему: Розробка програмного забезпечення "Автоматизація обліку навчального процесу приватної школи "СІА: свідомість, іновації, академічність""

Виконав: студент групи 4151

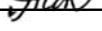
 Логвиненко В. Ю.

(підпис, ПІБ)

Керівник роботи:

доцент, к. ф-м.н.

(посада, науковий ступень вчене звання)

 Латанська Л.О.

(підпис, ПІБ)

Миколаїв – 2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проєктами
Кафедра програмного забезпечення автоматизованих систем
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма «Інженерія програмного забезпечення»

«ЗАТВЕРДЖУЮ»
Гарант освітньої програми
_____ доц. Макарова Л.М.
(підпис)
«_08_» ___04___ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття ступеня вищої освіти «бакалавр»

Студенту Логвиненку В'ячеславу Юрійовичу
(Прізвище, ім'я, по батькові)
1. Тема роботи: Розробка програмного забезпечення "Автоматизація обліку навчального процесу приватної школи "СІА: свідомість, іновації, академічність""
Керівник роботи Латанська Людмила Олексіївна
Затверджені наказом ректора №153 від 08.04.2025 р.
2. Термін подання роботи: 02.06.2025 р.
3. Вихідні дані по роботі:

4. Перелік питань, що належать до розробки (найменування розділів)
Титульний аркуш; Завдання на кваліфікаційну роботу; Анотація (українською та англійською мовами); Зміст; Перелік умовних позначень, символів, одиниць та термінів (при необхідності); Вступ; Аналіз практичної задачі інженерії програмного забезпечення; Проєкт програмного забезпечення; Результати розробки програмного забезпечення; Розділ з охорони праці; Висновки; Список використаних джерел; Додатки (технічне завдання, текст програми, опис програми, інструкція користувача, програма та методика випробувань програмного забезпечення).

5. Перелік презентаційних матеріалів 1 – Титульний слайд, 2 – Мета, завдання та об'єкт кваліфікаційної роботи, 3 – Аналоги програмного забезпечення, 4 – Призначення ПЗ, 5–Аналіз вимог (діаграма варіантів використання), 6–Архітектура ПЗ (діаграма компонентів), 7 – Архітектура ПЗ (діаграма розгортання), 8 – Ескізний проєкт ПЗ (діаграми діяльності), 9 – Ескізний проєкт ПЗ (концептуальна модель даних), 10 – Ескізний проєкт ПЗ (остаточний вигляд інтерфейсу), 11 – Технічний проєкт ПЗ (діаграма класів), 12 – Технічний проєкт ПЗ (діаграма послідовності прецеденту "Додати учня"), 13 – Технічний проєкт ПЗ (діаграма послідовності прецеденту "Додати підсумкову оцінку"), 14 – Технічний проєкт ПЗ (логічна модель даних), 15 – Робочий проєкт (засоби розробки), 16 – Результати розробки (головна сторінка адміністратора), 17 – Результати розробки (головна сторінка завуча), 18 – Результати розробки (головна сторінка бухгалтера), 19 – Результати розробки (головна сторінка медсестри), 20 – Висновки, 21 – Заключний слайд.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
4	Гурець Н.В., ст. викладач		

7. Дата видачі завдання 08.04.2025 р.

К А Л Е Н Д А Р Н И Й П Л А Н

Номер	Назва етапів роботи	Терміни виконання	Примітка
1.	Підготовка розділу ВСТУП	31.03.2025	ПП*
2.	Аналіз практичної задачі інженерії ПЗ	07.04.2025	ПП*
3.	Аналіз вимог до ПЗ	14.04.2025	ПП*
4.	Розробка архітектури ПЗ	21.04.2025	
5.	Розробка проєкту ПЗ	05.05.2025	
6.	Реалізація та тестування ПЗ	12.05.2025	
7.	Підготовка розділу Результати розробки ПЗ	16.05.2025	
8.	Підготовка розділу з охорони праці	19.05.2025	ПП*
9.	Підготовка розділу ВИСНОВКИ	23.05.2025	
10.	Оформлення списку використаних джерел та додатків	26.05.2025	
11.	Подання на кафедру ПЗАС тексту остаточного варіанту роботи, підписаного її керівником, у роздрукованому та електронному форматі разом із заявами щодо самостійності виконання роботи та ідентичності друкованої та електронної версій роботи (Додатки 1 і 2 «Порядку здійснення заходів з перевірки робіт на наявність текстових збігів/ідентичності/схожості із використанням програмно-технічних засобів», який введений в дію наказом ректора НУК за №20 від 20.01.2020 р.)	02.06.2025	не пізніше, ніж за 14 діб до захисту (згідно п.4.1 зазначеного Порядку
12.	Підготовка презентації та доповіді	06.06.2025	
13.	Попередній захист роботи на засіданні кафедри ПЗАС	09.06.2025	
14.	Подання на кафедру ПЗАС електронних версій наступних документів у форматі pdf: кваліфікаційної роботи; файлу-опису кваліфікаційної роботи (згідно Додатку до наказу ректора НУК за №287-уч від 19.05.2020 р.); презентації доповіді, а також Авторського договору з додатком	16.06.2025	

* - за результатами переддипломної практики (ПП), яка була з 03.02.2025 до 23.03.2025 р.

Студент _____ Логвиненко В.Ю.
(підпис) (ПІБ)

Керівник роботи _____ Латанська Л.О.
(підпис) (ПІБ)

АНОТАЦІЯ

Кваліфікаційна робота присвячена розв’язанню практичної задачі інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов, а саме розробці програмного забезпечення для автоматизації обліку навчального процесу приватної школи “СІА: свідомість, іновації, академічність”.

У кваліфікаційній роботі представлено аналіз практичної задачі інженерії програмного забезпечення, розглянуто існуючі програмні продукти та розроблено постановку задачі на створення програмного забезпечення. Виконано аналіз вимог, розроблено архітектуру та ескізний, технічний і робочий проєкти. Представлено результати розробки та складено розділ з охорони праці.

Об’єктом даної роботи є процес розробки програмного забезпечення для автоматизації обліку навчального процесу приватної школи “СІА: свідомість, іновації, академічність”.

Кваліфікаційна робота викладена на 158 сторінках друкованого тексту, містить 21 рисунок, 39 таблиць, список використаних джерел з 24 найменувань та 5 додатків.

ABSTRACT

Qualification work is devoted to solving a practical software engineering problem characterised by complexity and uncertainty of conditions in the development of a software for automating of accounting of the educational process of a private school “SIA: sentience, innovation, academia”.

The qualification work presents an analysis of a practical software engineering problem, reviews existing software products, and develops a software development task statement. Requirements analysis, architecture, and preliminary, technical, and detailed designs were developed. The development results are presented and a section on labour protection is compiled.

The object of this work is the process of development of a software for automating of accounting of the educational process of a private school “SIA: sentience, innovation, academia”.

The qualification work spans 158 pages of printed text, includes 21 figures, 39 tables, a list of references with 24 entries and 5 appendices.

ЗМІСТ

ВСТУП

У сучасних умовах стрімкого розвитку інформаційних технологій та цифровізації освітнього середовища автоматизація обліку навчального процесу стає надзвичайно актуальною, особливо для приватних навчальних закладів. Приватні школи, як окремий сегмент освітньої сфери, стикаються з низкою викликів, серед яких: розрізненість даних, високе навантаження на персонал, складність у координації між учасниками освітнього процесу, а також недостатній рівень доступу до оперативної та достеменної інформації для батьків. Традиційні методи ведення обліку та документообігу вже не відповідають вимогам сучасної освітньої системи, що зумовлює необхідність впровадження інноваційних програмних рішень для підвищення ефективності управління школою.

Потреба у створенні спеціалізованого програмного забезпечення зумовлена необхідністю формування єдиного інформаційного простору, який об’єднає ключові аспекти організації навчального процесу – розклад, оцінювання, сплату за навчання, зберігання різної облікової інформації. Такий підхід дозволить значно зменшити ризик помилок, оптимізувати рутинні процеси та підвищити прозорість навчального процесу.

Метою роботи є розробка програмного забезпечення для автоматизації обліку навчального процесу приватної школи “СІА: свідомість, іновації, академічність”, котре забезпечить автоматизацію та оптимізацію процесів ведення обліку навчального процесу й сприятиме підвищенню ефективності адміністрування й полегшенню взаємодій між учасниками процесу.

Для досягнення поставленої мети необхідно вирішити наступні задачі:

1) Провести аналіз практичної задачі інженерії програмного забезпечення.

2) Здійснити аналіз існуючих аналогів.

3) Виконати постановку задачі на розробку ПЗ для автоматизації обліку навчального процесу приватної школи.

4) Провести аналіз вимог.

5) Розробити архітектуру.

7) Розробити ескізний, технічний і робочий проєкти.

8) Розробити розділ “Охорона праці”.

9) Розробити документацію.

Об'єктом кваліфікаційної роботи є процес розробки програмного забезпечення для автоматизації обліку навчального процесу приватної школи “СІА: свідомість, іновації, академічність”.

Таким чином, розробка програмного забезпечення для автоматизації обліку навчального процесу у приватній школі “СІА: свідомість, іновації, академічність” сприятиме підвищенню ефективності управління закладом. Запропоноване рішення передбачає створення вебзастосунку, що дозволить зменшити навантаження, покращити якість комунікації, забезпечити зручний доступ до освітньої інформації. Дана кваліфікаційна робота спрямована на вирішення цих завдань шляхом розробки комплексного програмного продукту, який відповідатиме сучасним стандартам та потребам у сфері обліку навчального процесу.

1 АНАЛІЗ ПРАКТИЧНОЇ ЗАДАЧІ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ПОСТАНОВКА ЗАДАЧІ НА РОЗРОБКУ ПЗ ДЛЯ АВТОМАТИЗАЦІЇ ОБЛІКУ НАВЧАЛЬНОГО ПРОЦЕСУ ПРИВАТНОЇ ШКОЛИ “СІА: СВІДОМІСТЬ, ІНОВАЦІЇ, АКАДЕМІЧНІСТЬ”

1.1 Аналіз практичної задачі інженерії програмного забезпечення для автоматизації обліку навчального процесу приватної школи “СІА: свідомість, іновації, академічність”

В даній роботі вирішується практична задача інженерії програмного забезпечення, а саме задача розробки програмного забезпечення “Автоматизація обліку навчального процесу приватної школи “СІА: свідомість, іновації, академічність””. При розробці програмного забезпечення потрібно виконати такі стадії: провести аналіз практичної задачі інженерії програмного забезпечення, здійснити аналіз існуючих аналогів, виконати постановку задачі на розробку ПЗ, провести аналіз вимог, розробити архітектуру, розробити ескізний, технічний й робочий проєкти, розробити документацію.

Організація обліку учнів є одним з найважливіших завдань для організації навчального процесу у школі. Облік учнів включатиме у себе облік персональних даних учнів, представників, вчителів; облік предметів, оцінок, сплати за навчання, пільгових категорій, медичних карток; формування розкладу, формування звітності. Школа матиме приватну інформацію про кожного з учнів. При реєстрації нового учня у школі потрібно буде вказати хоча б одного його представника. Також при реєстрації нового учня одразу треба організувати його медичну картку. При навчанні в школі учень буде вивчати основні предмети та буде мати можливість вивчати додаткові предмети. Під час навчання учням будуть ставитись проміжні, семестрові, та річні оцінки.

Кожного навчального року виставлятиметься рахунок на сплату за навчання. При створенні рахунку на сплату, також враховуватимуться наявні в

учня пільгові категорії. Якщо учень матиме пільгову категорію, то буде надаватись знижка на сплату за навчання, вартість за навчання зі знижкою=стандартна вартість*(1-коефіцієнт пільги). Якщо учень матиме декілька пільгових категорій, то при обрахунку знижки враховується тільки одна, що має найбільший коефіцієнт.

Перелік пільгових категорій, котрі можуть мати учні:

1) Діти-сироти і діти, позбавлені батьківського піклування

2) Діти із сімей, які отримують допомогу відповідно до Закону України . «Про державну соціальну допомогу малозабезпеченим сім'ям»

3) Діти з особливими освітніми потребами, які навчаються у спеціальних та інклюзивних класах

4) Діти з числа осіб, визначених у статті 10 Закону України «Про статус ветеранів війни, гарантії їх соціального захисту»

5) Діти, один із батьків (батьки) яких:

а) має статус учасника бойових дій, який: – захищав незалежність, суверенітет та територіальну цілісність України і брав безпосередню участь в антитерористичній операції, забезпеченні її проведення чи у здійсненні заходів із забезпечення національної безпеки і оборони, відсічі і стримування збройної агресії Російської Федерації в Донецькій та Луганській областях, або та на інших територіях, де в період виконання цих завдань велися воєнні (бойові) дії забезпеченні (п. 19-24 ст.6 ЗУ «Про статус ветеранів війни, гарантії їх соціального захисту»); – був направленим, призваним до Афганістану в період ведення там бойових дій (п. 13-15 ст.6 ЗУ «Про статус ветеранів війни, гарантії їх соціального захисту»);

б) має статус особи з інвалідністю внаслідок війни (абз. 3 п. 4, п. 11-15 ч.2 ст.7 ЗУ «Про статус ветеранів війни, гарантії їх соціального захисту»);

в) має статус особи з інвалідністю внаслідок бойових дій у Афганістані;

г) захищав/захищає незалежність, суверенітет та територіальну цілісність України і брав/бере безпосередню участь в антитерористичній операції,

забезпеченні її проведення чи у здійсненні заходів із забезпечення національної безпеки і оборони, відсічі і стримування збройної агресії Російської Федерації в Донецькій та Луганській областях, або та на інших територіях, де в період виконання цих завдань велися воєнні (бойові) дії. Відповідно додатку 4 постанови КМУ №413 від 20.08.2014 року, підтвердженням є документ, який виданий командуванням військової частини, де військовослужбовець проходить службу на даний час, складений на офіційному бланку, зареєстрований в журналі вихідних документів, та затверджений підписом уповноваженої особи з відповідним текстом, в якому вказано термін перебування в районі антитерористичної операції даного військовослужбовця, або та на інших територіях, де в період виконання цих завдань велися воєнні (бойові) дії.

6) Діти, один із батьків (батьки) яких постраждалий учасник Революції Гідності

7) Діти з числа внутрішньо переміщених осіб, діти, які мають статус дитини, яка постраждала внаслідок воєнних дій і збройних конфліктів[1].

Також у школі буде складатися розклад занять.

Для ведення облікової діяльності у школі буде використано наступну інформацію, що представлена у таблиці 1.1.

Таблиця 1.1 – Інформація для ведення облікової діяльності школи

Узагальнення для котрого ця інформація	Інформація
1	2
учень	прізвище, ім'я, по батькові, клас навчання, дата народження, адреса прописки, адреса поточного проживання, електрона пошта, номер телефону, чи поновлене навчання, форма навчання, медична картка
медична картка	стать, зріст, вага, медична група, дата оновлення інформації, учень
ціна за навчання	діапазон класів для котрих ця ціна, ціна за рік навчання
представник учня	прізвище, ім'я, по батькові, номер телефону, місце роботи, електрона пошта
учень – представник учня	представник учня, ким приходиться учню, учень
пільгова категорія	назва, коефіцієнт знижки
учень – пільгова категорія	пільгова категорія, термін дії для учня, учень

Кінець таблиці 1.1

1	2
рахунок	номер документу, дата формування рахунку, дата до котрої треба сплатити рахунок, дата закриття рахунку, статус рахунку, вартість до сплати, вартість до сплати зі знижкою, навчальний рік учня коли був сформований рахунок, учень
вчителі	прізвище, ім'я, по батькові, спеціалізація, посада, електрона пошта, місце отримання вищої професійної освіти, номер телефону, досвід роботи
предмети	назва, тип предмету, рік навчання для котрого цей предмет
розклад	періодичність, день тижня, час початку, час закінчення, кабінет, вчитель, предмет, навчальна група
навчальна група	номер навчальної групи
учень – навчальна група	навчальна група, учень
оцінка	оцінка, дата, учень, предмет, вчитель
підсумкова оцінка	оцінка за перший семестр, оцінка за другий семестр, річна оцінка, рік навчання, навчальний рік, учень, предмет, вчитель.

Облікова робота потребує багато зусиль, тому треба автоматизувати ці процеси, щоб полегшити користувачам роботу з інформацією та зменшити кількість помилок викликаних людським фактором.

1.2 Аналіз існуючих аналогів

На сьогодні існує певний ряд програмних продуктів, котрі забезпечують автоматизацію навчального процесу у школі. Розглянемо деякі з них.

PowerSchool SIS − це популярна система управління інформацією в навчальних закладах, яка дозволяє вести облік учнів, вчителів та навчальних досягнень [2].

Плюси PowerSchool SIS:

− PowerSchool SIS працює в онлайн-режимі, що дозволяє вчителям, батькам і учням отримувати доступ до інформації про успішність та інші дані з будь-якого пристрою з Інтернетом.

− Платформа дозволяє вчителям та батькам спілкуватися, обмінюватися

повідомленнями та важливою інформацією про навчання учнів.

– PowerSchool дозволяє вчителям вести електронний журнал з оцінками і спостерігати за академічним прогресом учнів.

– Батьки можуть легко відстежувати академічний прогрес своїх дітей.

Мінуси PowerSchool SIS:

– Використання PowerSchool SIS вимагає значних фінансових витрат на ліцензію.

– Налаштування системи та інтеграція з іншими інформаційними системами може бути складним завданням.

– Як і будь-яка інша онлайн-платформа, PowerSchool SIS може стати об'єктом кібератак, тому безпека має важливе значення. У грудні 2024, платформа PowerSchool SIS стала жертвою кібератаки, потенційно поставивши під загрозу дані 60 мільйонів користувачів з Північної Америки [3].

– Робота з PowerSchool SIS вимагає доступу до мережі Інтернет, працює виключно на хмарному сервері.

На рисунку 1.1 зображена головна сторінка PowerSchool SIS.

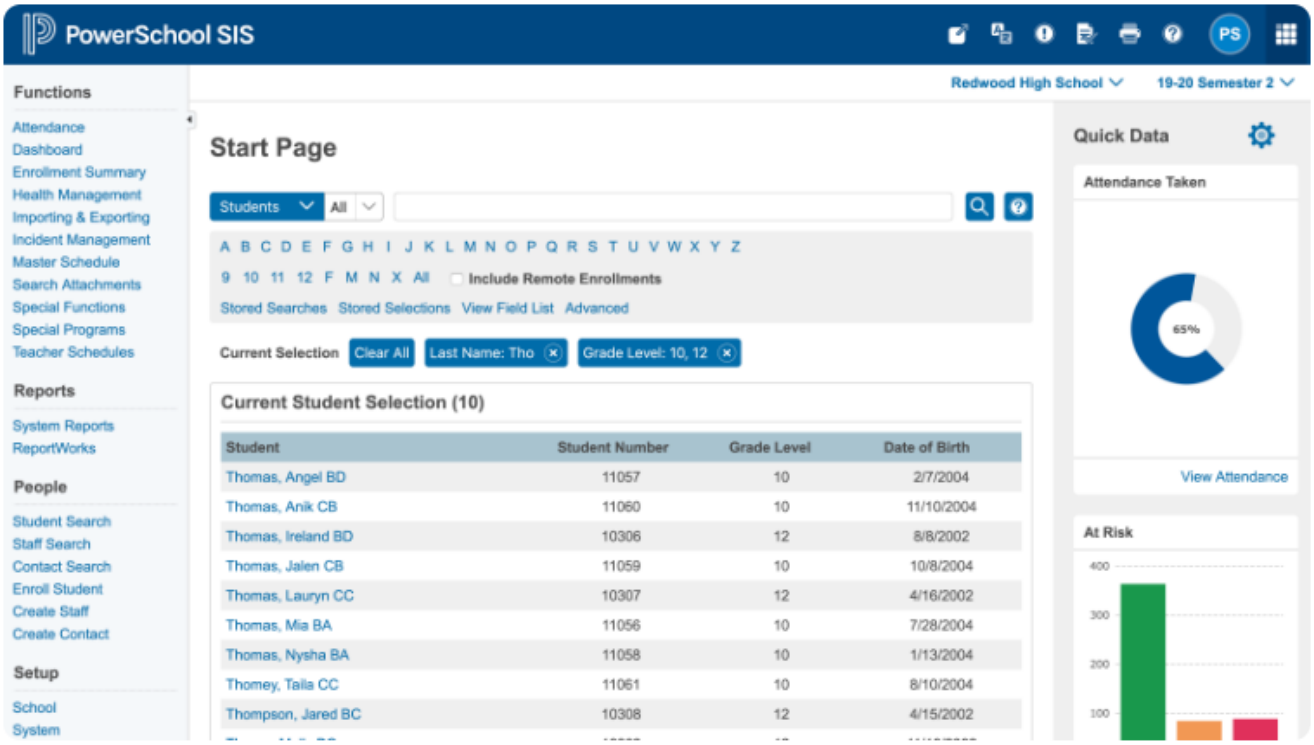


Рисунок 1.1 – Стартова сторінка PowerSchool SIS

ЄДЕБО (Єдина державна електронна база даних освіти) − це централізована система обліку учнів та інформації про навчальні заклади в Україні. Вона була створена для цілей збору, зберігання та обміну даними про освітній процес, учнів та навчальні заклади між різними рівнями управління освітою та іншими учасниками освітнього процесу [4].

Плюси ЄДЕБО:

− ЄДЕБО надає можливість централізованого обліку учнів та навчальних закладів, що допомагає створити єдину базу даних для всієї освітньої системи України.

− Зібрана в ЄДЕБО інформація може бути використана для ефективного планування ресурсів та розробки освітніх програм.

− ЄДЕБО дозволяє створювати звіти та аналізи, які можуть використовуватися в освітній політиці та прийнятті управлінських рішень.

− Збір інформації у централізованій системі сприяє відкритості та доступності даних для учасників освітнього процесу.

Мінуси ЄДЕБО:

− Робота з ЄДЕБО вимагає доступу до мережі Інтернет, працює виключно на хмарному сервері.

− Доєднання до централізованої системи спричинює більше бюрократичних процедур.

− Робота з ЄДЕБО вимагає доступу до Інтернету, що може бути обмежене в деяких областях або для окремих користувачів.

− облік учнів у ЄДЕБО здебільшого спрямований відносно обліку й надання інформації державі, щоб держава мала інформацію про облік учнів, упускаючи певні організаційні шкільні моменти.

На рисунку 1.2 зображено загальний вигляд розділів та модулів, в яких розміщено інформацію в ЄДЕБО.

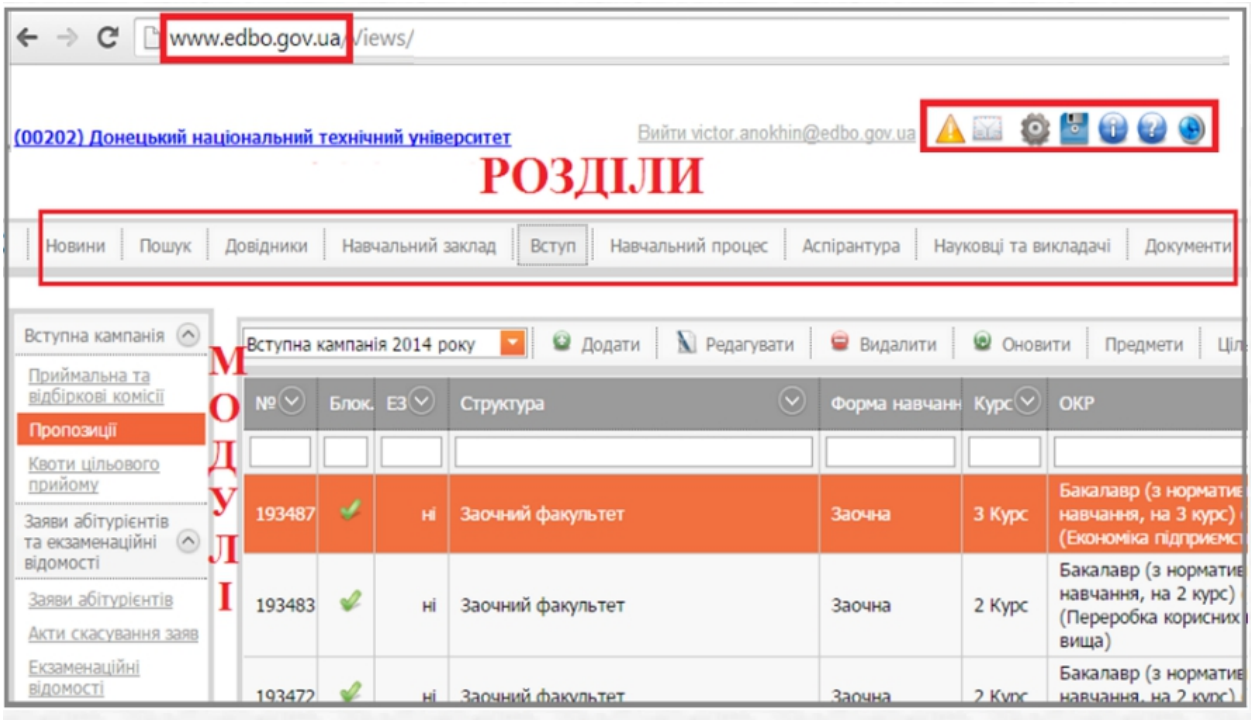


Рисунок 1.2 – Загальний вигляд розділів та модулів, в яких розміщено інформацію в ЄДЕБО [5]

SchoolMate – це комплексне рішення для управління навчальними закладами, орієнтоване переважно на мовні школи, приватні навчальні центри та освітні курси. Система допомагає автоматизувати адміністрування, ведення електронних журналів, розклад занять, фінансовий облік та взаємодію між викладачами, учнями та батьками [6].

Плюси SchoolMate:

– Інтуїтивно зрозумілий веб-інтерфейс і наявність мобільних додатків для студентів та викладачів спрощують використання програми.

– Програма підтримує 17 європейських мов, що робить її зручною для міжнародних навчальних закладів.

– Різні рівні доступу для адміністраторів, викладачів, студентів та батьків підвищують безпеку та персоналізацію роботи з даними.

Мінуси SchoolMate

– Робота з SchoolMate вимагає доступу до мережі Інтернет, працює виключно на хмарному сервері.

– Програма містить багато функцій, що може бути складним для навчальних закладів, яким потрібен лише базовий облік учнів і занять. Впровадження та навчання персоналу може зайняти певний час.

– Для невеликих навчальних закладів ціна підписки не є оптимальнішою. Також, наявність підписки, а не одноразового придбання ПЗ, призведе до значних грошових витрат на великому проміжку часу.

На рисунку 1.3 зображена сторінка зі списком учнів в SchoolMate.

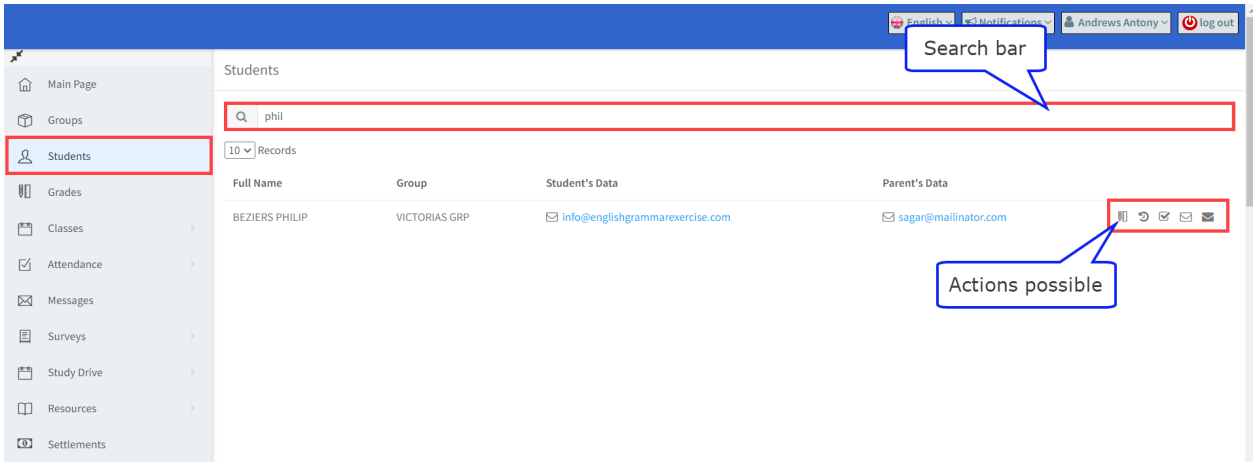


Рисунок 1.3 – Сторінка зі списком учнів в SchoolMate [7]

Для вирішення поставленої задачі наведені програми не є ефективними. Серед перелічених програм можна виділити такі ключові мінуси: велика собівартість, широкий функціонал, також потреба наявності безперебійного інтернет-з'єднання. Розроблювальний програмний продукт буде перекривати ці мінуси: матиме невелику собівартість, тільки необхідний замовнику функціонал, можливість працювати за відсутності з'єднання з мережею Інтернет, а також буде доступний до розширення.

1.3 Постановка задачі на розробку програмного забезпечення для автоматизації обліку навчального процесу приватної школи “СІА: свідомість, іновації, академічність”

Виходячи з вище викладеного сформулюємо постановку задачі. В роботі необхідно розробити програмне забезпечення “Автоматизація обліку навчального процесу приватної школи “СІА: свідомість, іновації, академічність””, яке буде відповідати наступним вимогам:

Вимоги до складу виконуваних функцій

Функції адміністратора:

– додавати інформацію (а саме про: вчителів, предмети, розклад занять, оцінки, підсумкові оцінки, рахунки, особові дані учнів, медичні картки, представників учнів, пільгові категорії);

– переглядати інформацію (а саме про: вчителів, предмети, розклад занять, оцінки, підсумкові оцінки, рахунки, особові дані учнів, медичні картки, представників учнів, пільгові категорії, ціну за навчання);

– редагувати інформацію (а саме про: вчителів, предмети, розклад занять, оцінки, підсумкові оцінки, особові дані учнів, медичні картки, представників учнів, пільгові категорії, ціну за навчання);

– змінювати статус рахунків;

– видаляти інформацію (а саме про: оцінки, підсумкові оцінки, розклад).

– формувати звітність:

-повний звіт про учня (особові дані учня, медичну картку учня, представників учня, пільгові категорії учня, підсумкові оцінки учня, рахунки учня),

-фінансовий звіт про учня (ПІБ учня, клас, електрона пошта, рахунки – усі поля),

-медичний звіт про учня (ПІБ учня, клас, медична картка – усі поля),

-табель успішності учня (ПІБ учня, клас, електрона пошта, предмети –

назва, підсумкові оцінки – усі поля).

Функції завуча:

– додавати нові предмети/оцінки/підсумкові оцінки/розклад;

– переглядати особову інформацію про учнів (ПІБ, клас, електрона пошта, форма навчання, стан у базі даних);

– переглядати інформацію про представників учнів (ПІБ, електрона пошта, номер телефону);

– переглядати інформацію про предмети (усі поля);

– переглядати інформацію про оцінки учнів (усі поля);

– переглядати інформацію про підсумкові оцінки учнів (усі поля);

– переглядати інформацію про розклад (усі поля);

– редагувати та видаляти розклад/оцінки/підсумкові оцінки (усі поля);

– формувати табель успішності учня (ПІБ учня, клас, електрона пошта, предмети – назва, підсумкові оцінки – усі поля).

Функції працівника бухгалтерії:

– переглядати особову інформацію про учнів (ПІБ, клас, електрона пошта, форма навчання, стан у базі даних);

– переглядати інформацію про представників учнів (ПІБ, електрона пошта, номер телефону);

– переглядати інформації про пільгові категорії учнів (усі поля);

– переглядати інформації про рахунки учнів (усі поля);

– переглядати інформацію про предмети учнів (усі поля);

– виставити учню рахунок на сплату навчання, тобто створити новий рахунок;

– змінювати стан рахунка;

– редагувати вартість предметів (вартість у таблиці предмети);

– формувати фінансову звітність учня (ПІБ учня, клас, електрона пошта, рахунки – усі поля).

Функції шкільної медсестри:

– переглядати особову інформацію про учнів (ПІБ, клас, електрона пошта, форма навчання, стан у базі даних);

– переглядати інформацію про представників учнів (ПІБ, електрона пошта, номер телефону);

– переглядати медичні картки учнів (усі поля);

– редагувати медичну картку учня (усі поля);

– формувати медичний звіт стосовно учня (медична картка – усі поля; особові дані учня – ПІБ, клас, електрона адреса).

Функції учня/представника учня:

– переглядати розклад занять учня(усі поля);

– переглядати оцінки учня (усі поля);

– переглядати підсумкові оцінки учня(усі поля).

Функції вчителя:

– переглядати інформацію про власних учнів (ПІБ, клас, електрона пошта, форма навчання, стан у базі даних);

– переглядати власні предмети (всі поля);

– переглядати власні оцінки(всі поля);

– переглядати власний розклад(всі поля);

– переглядати власні підсумкові оцінки (всі поля);

– додавати/редагувати/видаляти свої оцінки (всі поля).

Вимоги до організації вхідних та вихідних даних

Вхідною інформацією є дані, які вводяться вручну (в залежності від типу поля введення, дані можуть бути: числові цілі, числові десяткові, текстові, формату дати, булеві) або обираються із представлених списків (для уникнення можливих помилок, у випадку коли є певна фіксована й невелика кількість опцій

у заповненні інформації, у випадних списках представлені готові варіанти вибору), або отримуються з бази даних MariaDB.

Вихідною інформацією будуть результати операцій по взаємодії з базою даних, повідомлення, таблиці та сформовані звіти.

Вхідними даними будуть наступні:

– учень (ПІБ, клас, номер телефону, електрона пошта, зареєстрована та фактична адреси перебування, чи поновлене навчання, дата народження, форма навчання, стан у базі даних);

– медична картка (стать, медична група, вага, зріст, дата оновлення, учень);

– представник учня (ПІБ, місце роботи, номер телефону, електрона пошта);

– учень-представник учня (представник учня, стосунки з учнем, учень);

– учень-пільгова категорія (пільгова категорія, термін дії для учня, учень);

– підсумкова оцінка (за який рік навчання, оцінка за перший семестр, оцінка за другий семестр, річна оцінка, навчальний рік, учень, вчитель, предмет);

– рахунок (номер рахунку, дата формування рахунку, дата до котрої треба сплатити рахунок, дата закриття рахунку, статус рахунку, вартість до сплати, вартість до сплати з врахуванням пільг, за який рік навчання, учень);

– предмет (назва предмету, тип предмету, рік навчання для котрого цей предмет, вчителя що викладають);

– вчитель (ПІБ, номер телефону, електрона пошта, адреса проживання, дата народження, посада, напрям навчання, досвід роботи, навчальний заклад);

– розклад (періодичність заняття, день тижня для заняття, час початку заняття, час закінчення заняття, номер кабінету, вчитель, предмет, навчальна група);

– навчальна група(ідентифікатор групи);

– учень-навчальна група(учень, навчальна група);

– оцінка (дата оцінки, значення оцінки, учень, вчитель, предмет);

– розцінки (проміжок навчальних років, для котрого розцінка, вартість навчання).

Вимоги до інформаційної та програмної сумісності

Вимогли до програмного забезпечення серверу:

– система управління базами даних Mariadb;

– операційна система не нижче Windows 10;

Вимоги до програмного забезпечення клієнтської машини:

– операційна система не нижче Windows 10;

– браузер Google Chrome версії не нижче 93.0.4577.82 або браузер Mozilla Firefox версії не нижче 91.0.

Вимоги до складу і параметрів технічних засобів

Для коректної роботи програмного забезпечення потрібно мати сервер наступної мінімальної конфігурації:

– Intel Core i3–10100 3,6 ГГц або аналогічний за характеристиками AMD Ryzen 3 3300X;

– оперативна пам'ять 4 ГБ;

– 20 ГБ вільного місця на диску;

– з'єднання по локальній мережі.

Також маємо клієнтську машину наступної мінімальної конфігурації:

– Intel Core i3–10100 3,6 ГГц або аналогічний за характеристиками AMD Ryzen 3 3300X;

– оперативна пам'ять 2 ГБ;

– з'єднання по локальній мережі.

Повністю постановка задачі сформульована у технічному завданні, яке наведено у додатку А.

2 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЗАЦІЇ НАВЧАЛЬНОГО ПРОЦЕСУ ПРИВАТНОЇ ШКОЛИ “СІА: СВІДОМІСТЬ, ІНОВАЦІЇ, АКАДЕМІЧНІСТЬ”

2.1 Аналіз вимог до програмного забезпечення

2.1.1 Побудова моделі варіантів використання

В ході аналізу предметної області та формулювання вимог до ПЗ було виявлено та визначено шість акторів для взаємодії з ПЗ “Автоматизація обліку навчального процесу приватної школи “СІА: свідомість, іновації, академічність”: адміністратор, медсестра (вона ж “шкільна медсестра”), завуч (він же “завуч відповідальний за навчальний процес”), бухгалтер (він же “працівник бухгалтерії”), учень (він же “учень/представники учня”), вчитель.

Стислий опис дій адміністратора, медсестри, завуча, бухгалтера, учня, вчителя наведено у таблиці 2.1.

Таблиця 2.1 – Актори ПЗ

Актор	Короткий опис
Адміністратор	Відповідає за цілісність бази даних, має найвищі права в ПЗ. Відповідає за облік інформації про учня
Завуч	Відповідає за облік навчального процесу учнів.
Бухгалтер	Відповідає за контроль цін та контроль рахунків.
Медсестра	Відповідає за облік здоров'я учнів.
Учень	Переглядає інформацію по навчанню
Вчитель	Відповідає за оцінювання учнів

Виявлені варіанти використання ПЗ “Автоматизація обліку навчального процесу приватної школи “СІА: свідомість, іновації, академічність” представлена у таблиці 2.2.

Таблиця 2.2 – Варіанти використання ПЗ

Актор	Назва	Короткий опис
1	2	3
Адміністратор	Змінити пароль	Змінити поточний пароль авторизованого користувача на інший, котрий введе користувач.
Адміністратор	Переглянути інструкцію користувача	Висвітити користувачу у новому вікні інструкцію по користуванню програмним забезпеченням.
Адміністратор	Додати учня	Здійснити взаємодію з реєстром особистих даних, відправивши у цей реєстр запит на додавання нового запису до цього реєстру, одночасно відправивши запит на додавання нового запису у реєстр медичних карток.
Адміністратор	Редагувати учня	Здійснити взаємодію з реєстром особистих даних, відправивши у цей реєстр запит на редагування запису цього реєстру.
Адміністратор	Додати представника	Здійснити взаємодію з реєстром представників, відправивши у цей реєстр запит на додавання запису цього реєстру.
Адміністратор	Редагувати представника	Здійснити взаємодію з реєстром представників, відправивши у цей реєстр запит на редагування запису цього реєстру.
Адміністратор	Додати вчителя	Здійснити взаємодію з реєстром вчителів, відправивши у цей реєстр запит на додавання запису цього реєстру.
Адміністратор	Редагувати вчителя	Здійснити взаємодію з реєстром вчителів, відправивши у цей реєстр запит на редагування запису цього реєстру.
Адміністратор	Формувати повний звіт щодо учня	Здійснити взаємодію з реєстрами підсумкових оцінок, особистих даних, предметів, вчителів, медичних карток, рахунків та представників, щоб сформувати табель успішності учня.
Адміністратор	Редагувати медичну картку	Здійснити взаємодію з реєстром медичних карток, відправивши у цей реєстр запит на редагування запису цього реєстру.
Адміністратор	Формувати медичний звіт	Здійснити взаємодію з реєстрами медичних карток, особистих даних, щоб сформувати медичний звіт.
Адміністратор	Додати оцінку	Здійснити взаємодію з реєстром оцінок, відправивши у цей реєстр запит на додавання запису цього реєстру.
Адміністратор	Редагувати оцінку	Здійснити взаємодію з реєстром оцінок, відправивши у цей реєстр запит на редагування запису цього реєстру.
Адміністратор	Видалити оцінку	Здійснити взаємодію з реєстром оцінок, відправивши у цей реєстр запит на видалення запису цього реєстру.
Адміністратор	Додати розклад	Здійснити взаємодію з реєстром розкладу, відправивши у цей реєстр запит на додавання запису цього реєстру.
Адміністратор	Редагувати розклад	Здійснити взаємодію з реєстром розкладу, відправивши у цей реєстр запит на редагування запису цього реєстру.
Адміністратор	Видалити розклад	Здійснити взаємодію з реєстром розкладу, відправивши у цей реєстр запит на видалення запису цього реєстру.

Продовження таблиці 2.2

1	2	3
Адміністратор	Додати підсумкову оцінку	Здійснити взаємодію з реєстром підсумкових оцінок, відправивши у цей реєстр запит на додавання запису цього реєстру.
Адміністратор	Редагувати підсумкову оцінку	Здійснити взаємодію з реєстром підсумкових оцінок, відправивши у цей реєстр запит на редагування запису цього реєстру.
Адміністратор	Видалити підсумкову оцінку	Здійснити взаємодію з реєстром підсумкових оцінок, відправивши у цей реєстр запит на видалення запису цього реєстру.
Адміністратор	Додати предмет	Здійснити взаємодію з реєстром предметів, відправивши у цей реєстр запит на додавання запису цього реєстру.
Адміністратор	Редагувати предмет	Здійснити взаємодію з реєстром предметів, відправивши у цей реєстр запит на редагування запису цього реєстру.
Адміністратор	Формувати табель успішності учня	Здійснити взаємодію з реєстрами підсумкових оцінок, особистих даних, предметів та вчителів, щоб сформувати табель успішності учня.
Адміністратор	Виставити всім учням підсумкові оцінки	Здійснити взаємодію з реєстром підсумкових оцінок, відправивши у цей реєстр запит на генерацію підсумкових оцінок для кожного учня.
Адміністратор	Виставити всім учням предмети у відповідність	Здійснити взаємодію з реєстром предметів, відправивши у цей реєстр запит на виставлення зв'язків між учнями та предметами, котрі ці учні вивчають.
Адміністратор	Додати рахунок	Здійснити взаємодію з реєстром рахунків, відправивши у цей реєстр запит на додавання запису цього реєстру.
Адміністратор	Відмінити рахунок	Здійснити взаємодію з реєстром рахунків, відправивши у цей реєстр запит на відміну запису цього реєстру.
Адміністратор	Закрити рахунок	Здійснити взаємодію з реєстром рахунків, відправивши у цей реєстр запит на закриття запису цього реєстру.
Адміністратор	Додати пільгову категорію	Здійснити взаємодію з реєстром пільгових категорій, відправивши у цей реєстр запит на додавання запису цього реєстру.
Адміністратор	Редагувати пільгову категорію	Здійснити взаємодію з реєстром пільгових категорій, відправивши у цей реєстр запит на редагування запису цього реєстру.
Адміністратор	Формувати фінансову звітність учня	Здійснити взаємодію з реєстрами пільгових категорій, особистих даних, рахунків, щоб сформувати фінансову звітність учня.
Вчитель	Змінити пароль	Змінити поточний пароль авторизованого користувача на інший, котрий введе користувач.

Продовження таблиці 2.2

1	2	3
Вчитель	Переглянути інструкцію користувача	Висвітити користувачу у новому вікні інструкцію по користуванню програмним забезпеченням.
Вчитель	Додати оцінку	Здійснити взаємодію з реєстром оцінок, відправивши у цей реєстр запит на додавання запису цього реєстру.
Вчитель	Редагувати оцінку	Здійснити взаємодію з реєстром оцінок, відправивши у цей реєстр запит на редагування запису цього реєстру.
Вчитель	Видалити оцінку	Здійснити взаємодію з реєстром оцінок, відправивши у цей реєстр запит на видалення запису цього реєстру.
Завуч	Змінити пароль	Змінити поточний пароль авторизованого користувача на інший, котрий введе користувач.
Завуч	Переглянути інструкцію користувача	Висвітити користувачу у новому вікні інструкцію по користуванню програмним забезпеченням.
Завуч	Додати оцінку	Здійснити взаємодію з реєстром оцінок, відправивши у цей реєстр запит на додавання запису цього реєстру.
Завуч	Редагувати оцінку	Здійснити взаємодію з реєстром оцінок, відправивши у цей реєстр запит на редагування запису цього реєстру.
Завуч	Видалити оцінку	Здійснити взаємодію з реєстром оцінок, відправивши у цей реєстр запит на видалення запису цього реєстру.
Завуч	Додати оцінку	Здійснити взаємодію з реєстром оцінок, відправивши у цей реєстр запит на додавання запису цього реєстру.
Завуч	Редагувати оцінку	Здійснити взаємодію з реєстром оцінок, відправивши у цей реєстр запит на редагування запису цього реєстру.
Завуч	Видалити оцінку	Здійснити взаємодію з реєстром оцінок, відправивши у цей реєстр запит на видалення запису цього реєстру.
Завуч	Додати розклад	Здійснити взаємодію з реєстром розкладу, відправивши у цей реєстр запит на додавання запису цього реєстру.
Завуч	Редагувати розклад	Здійснити взаємодію з реєстром розкладу, відправивши у цей реєстр запит на редагування запису цього реєстру.
Завуч	Видалити розклад	Здійснити взаємодію з реєстром розкладу, відправивши у цей реєстр запит на видалення запису цього реєстру.
Завуч	Додати підсумкову оцінку	Здійснити взаємодію з реєстром підсумкових оцінок, відправивши у цей реєстр запит на додавання запису цього реєстру.
Завуч	Редагувати підсумкову оцінку	Здійснити взаємодію з реєстром підсумкових оцінок, відправивши у цей реєстр запит на редагування запису цього реєстру.
Завуч	Видалити підсумкову оцінку	Здійснити взаємодію з реєстром підсумкових оцінок, відправивши у цей реєстр запит на видалення запису цього реєстру.

Продовження таблиці 2.2

1	2	3
Завуч	Додати предмет	Здійснити взаємодію з реєстром предметів, відправивши у цей реєстр запит на додавання запису цього реєстру.
Завуч	Редагувати предмет	Здійснити взаємодію з реєстром предметів, відправивши у цей реєстр запит на редагування запису цього реєстру.
Завуч	Формувати табель успішності учня	Здійснити взаємодію з реєстрами підсумкових оцінок, особистих даних, предметів та вчителів, щоб сформувати табель успішності учня.
Завуч	Виставити всім учням підсумкові оцінки	Здійснити взаємодію з реєстром підсумкових оцінок, відправивши у цей реєстр запит на генерацію підсумкових оцінок для кожного учня.
Завуч	Виставити всім учням предмети у відповідність	Здійснити взаємодію з реєстром предметів, відправивши у цей реєстр запит на виставлення зв'язків між учнями та предметами, котрі ці учні вивчають.
Медсестра	Змінити пароль	Змінити поточний пароль авторизованого користувача на інший, котрий введе користувач.
Медсестра	Переглянути інструкцію користувача	Висвітити користувачу у новому вікні інструкцію по користуванню програмним забезпеченням.
Медсестра	Редагувати медичну картку	Здійснити взаємодію з реєстром медичних карток, відправивши у цей реєстр запит на редагування запису цього реєстру.
Медсестра	Формувати медичний звіт	Здійснити взаємодію з реєстрами медичних карток, особистих даних, щоб сформувати медичний звіт.
Бухгалтер	Змінити пароль	Змінити поточний пароль авторизованого користувача на інший, котрий введе користувач.
Бухгалтер	Переглянути інструкцію користувача	Висвітити користувачу у новому вікні інструкцію по користуванню програмним забезпеченням.
Бухгалтер	Додати рахунок	Здійснити взаємодію з реєстром рахунків, відправивши у цей реєстр запит на додавання запису цього реєстру.
Бухгалтер	Відмінити рахунок	Здійснити взаємодію з реєстром рахунків, відправивши у цей реєстр запит на відміну запису цього реєстру.
Бухгалтер	Закрити рахунок	Здійснити взаємодію з реєстром рахунків, відправивши у цей реєстр запит на закриття запису цього реєстру.
Бухгалтер	Додати пільгову категорію	Здійснити взаємодію з реєстром пільгових категорій, відправивши у цей реєстр запит на додавання запису цього реєстру.
Бухгалтер	Редагувати пільгову категорію	Здійснити взаємодію з реєстром пільгових категорій, відправивши у цей реєстр запит на редагування запису цього реєстру.

Кінець таблиці 2.2

1	2	3
Бухгалтер	Формувати фінансову звітність учня	Здійснити взаємодію з реєстрами пільгових категорій, особистих даних, рахунків, щоб сформувати фінансову звітність учня.
Учень	Змінити пароль	Змінити поточний пароль авторизованого користувача на інший, котрий введе користувач.
Учень	Переглянути інструкцію користувача	Висвітити користувачу у новому вікні інструкцію по користуванню програмним забезпеченням.
Учень	Переглянути розклад	Здійснити взаємодію з реєстрами розкладу, предметів, особистих даних та вчителів, щоб відобразити розклад учня.
Учень	Переглянути табель	Здійснити взаємодію з реєстрами предметів, особистих даних, вчителів та підсумкових оцінок, щоб відобразити табель учня.
Учень	Переглянути щоденник	Здійснити взаємодію з реєстрами предметів, особистих даних, вчителів та оцінок, щоб відобразити щоденник учня.

В перелічених варіантах використання розкривається як користувачі будуть взаємодіяти з ПЗ “Автоматизація обліку навчального процесу приватної школи “СІА: свідомість, іновації, академічність”” та які функції і можливості повинні бути реалізовані для задоволення їх потреб. Аналіз та уточнення цих варіантів використання допомагає забезпечити ефективну взаємодію з користувачами ПЗ. На рисунку 2.1 зображено діаграму варіантів використання для акторів: адміністратор, завуч, медсестра, бухгалтер, учень, вчитель.

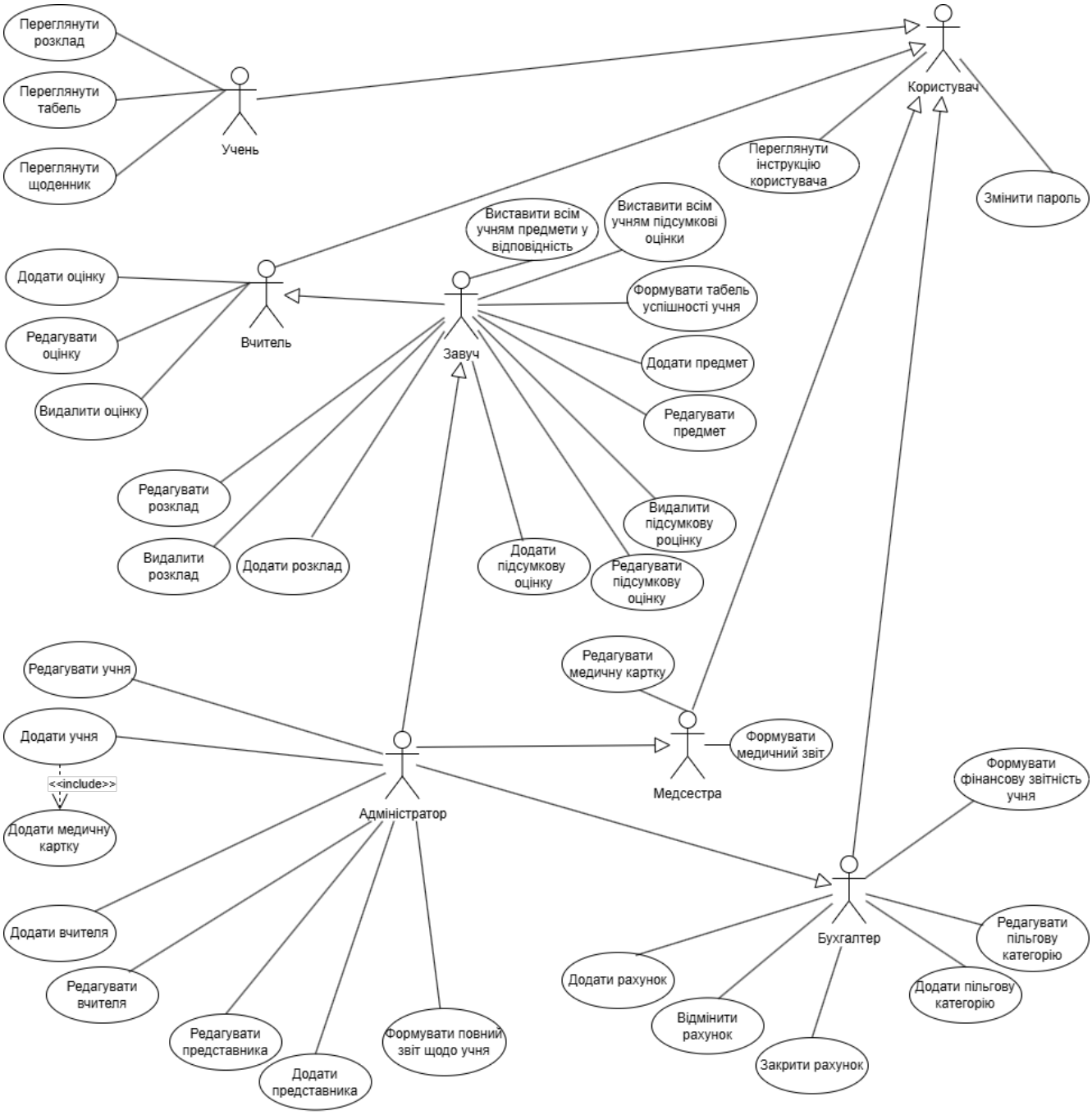


Рисунок 2.1 – Діаграма варіантів використання ПЗ “Автоматизація обліку навчального процесу приватної школи “СІА: свідомість, іновації, академічність””

Для отриманих варіантів використання розробимо специфікації.

2.1.2 Специфікації варіантів використання

Специфікації варіантів використання ПЗ “Автоматизація обліку навчального процесу приватної школи “СІА: свідомість, іновації, академічність”” приведено у таблицях 2.3-2.36.

Таблиця 2.3 – Специфікація варіанту використання “Додати учня”

Складова	Опис
Опис прецеденту	Призначений для додавання учня до бази даних
Дійові особи прецеденту	Адміністратор
Передумови	Адміністратор авторизований у системі й знаходиться на Головній сторінці (сторінці, що зустрічає його при успішній авторизації), на цій сторінці перед собою бачить перелік учнів
Базовий потік	– Користувач обирає опцію “Додати учня” – Система відображає на екрані перелік представників – Користувач обирає зі списку представника та обирає опцію “Обрати” для обраного представника – Система відображає форму вводу особистих даних учня – Користувач заповнює поля форми й обирає опцію “Зберегти” – Система додає новий запис до бази даних. Якщо не вдається, то перенаправляє на головну сторінку.
Альтернативні потоки	Немає
Постумова	Якщо варіант використання виконався успішно, то учень буде доданий до бази даних (будуть додані особисті дані учня). Також, для учня, автоматично додасться медична картка до бази даних. А користувача ПЗ перенаправить на сторінку переліку учнів.

Таблиця 2.4 – Специфікація варіанту використання “Редагувати учня”

Складова	Опис
1	2
Опис прецеденту	Призначений для зміни учня у базі даних
Дійові особи прецеденту	Адміністратор
Передумови	Адміністратор авторизований у системі й знаходиться на Головній сторінці (сторінці, що зустрічає його при успішній авторизації), на цій сторінці перед собою бачить перелік учнів

Кінець таблиці 2.4

1	2
Базовий потік	– Користувач обирає учня зі списку учнів та обирає опцію “Детальніше” – Система відображає на екрані більш детальну інформацію про обраного учня, включаючи його фотографію, та кнопки – Користувач обирає опцію на кнопку “Редагувати учня” – Система відображає форму вводу особистих даних учня, в котрій у полях знаходяться поточні значення – Користувач змінює поля форми й обирає опцію “Оновити” – Система зберігає змінений запис у базі даних. Якщо не вдається, то перенаправляє на головну сторінку.
Альтернативні потоки	Немає
Постумова	Якщо варіант використання виконався успішно, то особисті дані учня буде змінено у базі даних. А користувача ПЗ перенаправить на сторінку переліку учнів.

Таблиця 2.5 – Специфікація варіанту використання “Додати представника”

Складова	Опис
Опис прецеденту	Призначений для додавання представника до бази даних
Дійові особи прецеденту	Адміністратор
Передумови	Адміністратор авторизований у системі й знаходиться на Головній сторінці (сторінці, що зустрічає його при успішній авторизації), на цій сторінці перед собою бачить перелік учнів
Базовий потік	– Користувач обирає опцію “Представники” – Система відображає на екрані перелік представників з формою для додавання – Користувач заповнює поля форми й обирає опцію “Додати представника” – Система додає новий запис до бази даних. Якщо не вдається, то перенаправляє на головну сторінку.
Альтернативні потоки	Немає
Постумова	Якщо варіант використання виконався успішно, то представник буде доданий до бази даних. А користувача ПЗ перенаправить на сторінку з переліком представників.

Таблиця 2.6 – Специфікація варіанту використання “Редагувати представника”

Складова	Опис
Опис прецеденту	Призначений для зміни представника у базі даних
Дійові особи прецеденту	Адміністратор
Передумови	Адміністратор авторизований у системі й знаходиться на Головній сторінці (сторінці, що зустрічає його при успішній авторизації), на цій сторінці перед собою бачить перелік учнів
Базовий потік	– Користувач обирає опцію “Представники” – Система відображає на екрані перелік представників – Користувач обирає представника зі списку й обирає опцію “Редагувати” для обраного представника – Система відображає форму вводу даних представника, в котрій у полях знаходяться поточні значення – Користувач змінює поля форми й обирає опцію “Оновити” – Система зберігає змінений запис у базі даних. Якщо не вдається, то перенаправляє на головну сторінку.
Альтернативні потоки	Немає
Постумова	Якщо варіант використання виконався успішно, то дані представника буде змінено у базі даних. А користувача ПЗ перенаправить на сторінку з переліком представників.

Таблиця 2.7 – Специфікація варіанту використання “Додати вчителя”

Складова	Опис
Опис прецеденту	Призначений для додавання вчителя до бази даних
Дійові особи прецеденту	Адміністратор
Передумови	Адміністратор авторизований у системі й знаходиться на Головній сторінці (сторінці, що зустрічає його при успішній авторизації), на цій сторінці перед собою бачить перелік учнів
Базовий потік	– Користувач обирає опцію “Вчителі” – Система відображає на екрані перелік вчителів з формою для додавання – Користувач заповнює поля форми й обирає опцію “Додати вчителя” – Система додає новий запис до бази даних. Якщо не вдається, то перенаправляє на головну сторінку.
Альтернативні потоки	Немає
Постумова	Якщо варіант використання виконався успішно, то вчитель буде доданий до бази даних. А користувача ПЗ перенаправить на сторінку з переліком вчителів.

Таблиця 2.8 – Специфікація варіанту використання “Редагувати вчителя”

Складова	Опис
Опис прецеденту	Призначений для зміни вчителя у базі даних
Дійові особи прецеденту	Адміністратор
Передумови	Адміністратор авторизований у системі й знаходиться на Головній сторінці (сторінці, що зустрічає його при успішній авторизації), на цій сторінці перед собою бачить перелік учнів
Базовий потік	– Користувач обирає опцію “Вчителі” – Система відображає на екрані перелік вчителів – Користувач обирає вчителя зі списку й обирає опцію “Редагувати” для обраного вчителя – Система відображає форму вводу даних вчителя, в котрій у полях знаходяться поточні значення – Користувач змінює поля форми й обирає опцію “Оновити” – Система зберігає змінений запис у базі даних. Якщо не вдається, то перенаправляє на головну сторінку.
Альтернативні потоки	Немає
Постумова	Якщо варіант використання виконався успішно, то дані вчителя буде змінено у базі даних. А користувача ПЗ перенаправить на сторінку з переліком вчителів.

Таблиця 2.9 – Специфікація варіанту використання “Редагувати медичну картку”

Складова	Опис
1	2
Опис прецеденту	Призначений для зміни медичної картки у базі даних
Дійові особи прецеденту	Адміністратор, Медсестра
Передумови	Адміністратор/Медсестра авторизований у системі й знаходиться на Головній сторінці (сторінці, що зустрічає його при успішній авторизації), на цій сторінці перед собою бачить перелік учнів
Базовий потік	– Користувач обирає учня зі списку учнів та обирає опцію “Детальніше” – Система відображає на екрані більш детальну інформацію про обраного учня, включаючи його фотографію, та кнопки – Користувач обирає опцію на кнопку “Медична картка учня” – Система відображає на екрані медичну картку учня – Користувач обирає опцію “Редагувати” – Система відображає форму вводу даних медичної картки, в котрій у полях знаходяться поточні значення – Користувач змінює поля форми й обирає опцію “Оновити” – Система зберігає змінений запис у базі даних. Якщо не вдається, то перенаправляє на головну сторінку.

Кінець таблиці 2.9

1	2
Альтернативні потоки	– Користувач обирає учня зі списку учнів та обирає опцію “Медична картка” – Система відображає на екрані медичну картку учня – Користувач обирає опцію “Редагувати” – Система відображає форму вводу даних медичної картку, в котрій у полях знаходяться поточні значення – Користувач змінює поля форми й обирає опцію “Оновити”
Постумова	Якщо варіант використання виконався успішно, то дані медичної картки буде змінено у базі даних. А користувача ПЗ перенаправить на сторінку медичної картки учня.

Таблиця 2.10 – Специфікація варіанту використання “Додати пільгову категорію”

Складова	Опис
Опис прецеденту	Призначений для додавання пільгової категорії до бази даних
Дійові особи прецеденту	Адміністратор, Бухгалтер
Передумови	Адміністратор/Бухгалтер авторизований у системі й знаходиться на Головній сторінці (сторінці, що зустрічає його при успішній авторизації), на цій сторінці перед собою бачить перелік учнів
Базовий потік	– Користувач обирає опцію “Пільгові категорії” – Система відображає на екрані перелік пільгових категорій з формою для додавання – Користувач заповнює поля форми й обирає опцію “Додати пільгову категорію” – Система додає новий запис до бази даних. Якщо не вдається, то перенаправляє на головну сторінку.
Альтернативні потоки	Немає
Постумова	Якщо варіант використання виконався успішно, то пільгова категорія буде додана до бази даних. А користувача ПЗ перенаправить на сторінку переліку пільгових категорій.

Таблиця 2.11 – Специфікація варіанту використання “Редагувати пільгову категорію”

Складова	Опис
1	2
Опис прецеденту	Призначений для зміни пільгової категорії у базі даних
Дійові особи прецеденту	Адміністратор, Бухгалтер

Кінець таблиці 2.11

1	2
Передумови	Адміністратор/Бухгалтер авторизований у системі й знаходиться на Головній сторінці (сторінці, що зустрічає його при успішній авторизації), на цій сторінці перед собою бачить перелік учнів
Базовий потік	– Користувач обирає опцію “Пільгові категорії” – Система відображає на екрані перелік пільгових категорій – Користувач обирає пільгову категорію зі списку й обирає опцію “Редагувати” навпроти обраної пільгової категорії – Система відображає форму вводу даних пільгової категорії, в котрій у полях знаходяться поточні значення – Користувач змінює поля форми й обирає опцію “Оновити” – Система зберігає змінений запис у базі даних. Якщо не вдається, то перенаправляє на головну сторінку.
Альтернативні потоки	Немає
Постумова	Якщо варіант використання виконався успішно, то дані пільгової категорії буде змінено у базі даних. А користувача ПЗ перенаправить на сторінку пільгових категорій.

Таблиця 2.12 – Специфікація варіанту використання “Додати рахунок”

Складова	Опис
Опис прецеденту	Призначений для додавання рахунку до бази даних
Дійові особи прецеденту	Адміністратор, Бухгалтер
Передумови	Адміністратор/Бухгалтер авторизований у системі й знаходиться на Головній сторінці (сторінці, що зустрічає його при успішній авторизації), на цій сторінці перед собою бачить перелік учнів
Базовий потік	– Користувач обирає учня зі списку учнів та обирає опцію “Детальніше” – Система відображає на екрані більш детальну інформацію про обраного учня, включаючи його фотографію, та кнопки – Користувач обирає опцію на кнопку “Рахунки” – Система відображає на екрані перелік рахунків обраного учня – Користувач обирає опцію “Додати рахунок” – Система відображає форму вводу даних рахунку – Користувач заповнює поля форми й обирає опцію “Зберегти” – Система додає новий запис до бази даних. Якщо не вдається, то перенаправляє на головну сторінку.
Альтернативні потоки	Немає
Постумова	Якщо варіант використання виконався успішно, то рахунок буде доданий до бази даних. А користувача ПЗ перенаправить на сторінку з переліком рахунків учня.

Таблиця 2.13 – Специфікація варіанту використання “Закрити рахунок”

Складова	Опис
Опис прецеденту	Призначений для зміни стану рахунку на “СПЛАЧЕНО” у базі даних
Дійові особи прецеденту	Адміністратор, Бухгалтер
Передумови	Адміністратор/Бухгалтер авторизований у системі й знаходиться на Головній сторінці (сторінці, що зустрічає його при успішній авторизації), на цій сторінці перед собою бачить перелік учнів
Базовий потік	– Користувач обирає учня зі списку учнів та обирає опцію “Детальніше” – Система відображає на екрані більш детальну інформацію про обраного учня, включаючи його фотографію та кнопки – Користувач обирає опцію на кнопку “Рахунки” – Система відображає на екрані перелік рахунків обраного учня – Користувач обирає рахунок зі списку й обирає опцію “Сплачено” для обраного рахунку – Система зберігає змінений запис у базі даних. Змінюючи статус рахунку на “СПЛАЧЕНО” й у полі дата закриття рахунку виставляється поточна дата. Якщо не вдається, то перенаправляє на головну сторінку.
Альтернативні потоки	Немає
Постумова	Якщо варіант використання виконався успішно, то дані рахунку буде змінено у базі даних. А користувача ПЗ перенаправить на сторінку рахунків учня.

Таблиця 2.14 – Специфікація варіанту використання “Відмінити рахунок”

Складова	Опис
1	2
Опис прецеденту	Призначений для зміни стану рахунку на “ВІДМІНЕНО” у базі даних
Дійові особи прецеденту	Адміністратор, Бухгалтер
Передумови	Адміністратор/Бухгалтер авторизований у системі й знаходиться на Головній сторінці (сторінці, що зустрічає його при успішній авторизації), на цій сторінці перед собою бачить перелік учнів
Базовий потік	– Користувач обирає учня зі списку учнів та обирає опцію “Детальніше” – Система відображає на екрані більш детальну інформацію про обраного учня, включаючи його фотографію, та кнопки – Користувач обирає опцію на кнопку “Рахунки” – Система відображає на екрані перелік рахунків обраного учня – Користувач обирає рахунок зі списку й обирає опцію “Відмінено” для обраного рахунку – Система зберігає змінений запис у базі даних. Змінюючи статус рахунку на “ВІДМІНЕНО”. Якщо не вдається, то перенаправляє на головну сторінку.

Кінець таблиці 2.14

1	2
Альтернативні потоки	Немає
Постумова	Якщо варіант використання виконався успішно, то дані рахунку буде змінено у базі даних. А користувача ПЗ перенаправить на сторінку рахунків учня.

Таблиця 2.15 – Специфікація варіанту використання “Додати предмет”

Складова	Опис
Опис прецеденту	Призначений для додавання предмету до бази даних
Дійові особи прецеденту	Адміністратор, Завуч
Передумови	Адміністратор/Завуч авторизований у системі й знаходиться на Головній сторінці (сторінці, що зустрічає його при успішній авторизації), на цій сторінці перед собою бачить перелік учнів
Базовий потік	– Користувач обирає опцію “Предмети” – Система відображає на екрані перелік предметів з формою для додавання – Користувач заповнює поля форми й обирає опцію “Додати предмет” – Система додає новий запис до бази даних. Якщо не вдається, то перенаправляє на головну сторінку.
Альтернативні потоки	Немає
Постумова	Якщо варіант використання виконався успішно, то предмет буде доданий до бази даних. А користувача ПЗ перенаправить на сторінку з переліком предметів.

Таблиця 2.16 – Специфікація варіанту використання “Редагувати предмет”

Складова	Опис
1	2
Опис прецеденту	Призначений для зміни предмету у базі даних
Дійові особи прецеденту	Адміністратор, Завуч
Передумови	Адміністратор/Завуч авторизований у системі й знаходиться на Головній сторінці (сторінці, що зустрічає його при успішній авторизації), на цій сторінці перед собою бачить перелік учнів
Базовий потік	– Користувач обирає опцію “Предмети” – Система відображає на екрані перелік предметів – Користувач обирає предмет зі списку й обирає опцію “Редагувати” для обраного предмету – Система відображає форму вводу даних предмету, в котрій у полях знаходяться поточні значення – Користувач змінює поля форми й обирає опцію “Оновити” – Система зберігає змінений запис у базі даних. Якщо не вдається, то перенаправляє на головну сторінку.

Кінець таблиці 2.16

1	2
Альтернативні потоки	Немає
Постумова	Якщо варіант використання виконався успішно, то дані предмету буде змінено у базі даних. А користувача ПЗ перенаправить на сторінку з переліком предметів.

Таблиця 2.17 – Специфікація варіанту використання “Додати розклад”

Складова	Опис
1	2
Опис прецеденту	Призначений для додавання розкладу до бази даних
Дійові особи прецеденту	Адміністратор, Завуч
Передумови	Адміністратор/Завуч авторизований у системі й знаходиться на Головній сторінці (сторінці, що зустрічає його при успішній авторизації), на цій сторінці перед собою бачить перелік учнів
Базовий потік	– Користувач обирає опцію “Розклад” – Система відображає на екрані перелік розкладу – Користувач обирає опцію “Додати розклад” – Система відображає список предметів – Користувач обирає основний предмет зі списку й обирає опцію “Обрати” для обраного предмету – Система відображає список вчителів – Користувач обирає вчителя зі списку й обирає опцію “Обрати” для обраного вчителя – Система відображає список класів у котрих ведуть цей предмет – Користувач обирає клас зі списку й обирає опцію “Обрати” для обраного класу – Система відображає форму вводу даних розкладу – Користувач заповнює поля форми й обирає опцію “Зберегти” – Система додає новий запис до бази даних. Якщо не вдається, то перенаправляє на головну сторінку.
Альтернативні потоки	– Користувач обирає опцію “Розклад” – Система відображає на екрані перелік розкладу – Користувач обирає опцію “Додати розклад” – Система відображає список предметів – Користувач обирає додатковий предмет зі списку й обирає опцію “Обрати” для обраного предмету – Система відображає список вчителів – Користувач обирає вчителя зі списку й обирає опцію “Обрати” для обраного вчителя – Система відображає скільки учнів вивчають цей предмет – Користувач натискає кнопку “Назначити всім” – Система відображає форму вводу даних розкладу – Користувач заповнює поля форми й обирає опцію “Зберегти” – Система додає новий запис до бази даних. Якщо не вдається, то перенаправляє на головну сторінку.

Кінець таблиці 2.17

1	2
Постумова	Якщо варіант використання виконався успішно, то розклад буде доданий до бази даних. А користувача ПЗ перенаправить на сторінку з переліком розкладу.

Таблиця 2.18 – Специфікація варіанту використання “Редагувати розклад”

Складова	Опис
Опис прецеденту	Призначений для зміни розкладу у базі даних
Дійові особи прецеденту	Адміністратор, Завуч
Передумови	Адміністратор/Завуч авторизований у системі й знаходиться на Головній сторінці (сторінці, що зустрічає його при успішній авторизації), на цій сторінці перед собою бачить перелік учнів
Базовий потік	– Користувач обирає опцію “Розклад” – Система відображає на екрані перелік розкладу – Користувач обирає розклад зі списку й обирає опцію “Редагувати” для обраного розкладу – Система відображає форму вводу даних розкладу, в котрій у полях знаходяться поточні значення – Користувач змінює поля форми й обирає опцію “Оновити” – Система зберігає змінений запис у базі даних. Якщо не вдається, то перенаправляє на головну сторінку.
Альтернативні потоки	Немає
Постумова	Якщо варіант використання виконався успішно, то дані розкладу буде змінено у базі даних. А користувача ПЗ перенаправить на сторінку з переліком розкладу.

Таблиця 2.19 – Специфікація варіанту використання “Видалити розклад”

Складова	Опис
1	2
Опис прецеденту	Призначений для видалення розкладу з бази даних
Дійові особи прецеденту	Адміністратор, Завуч
Передумови	Адміністратор/Завуч авторизований у системі й знаходиться на Головній сторінці (сторінці, що зустрічає його при успішній авторизації), на цій сторінці перед собою бачить перелік учнів
Базовий потік	– Користувач обирає опцію “Розклад” – Система відображає на екрані перелік розкладу – Користувач обирає розклад зі списку й обирає опцію “Видалити” для обраного розкладу – Система відображає попередження про те що дані цього розкладу буде видалено назавжди. – Користувач обирає опцію “Видалити розклад” – Система видаляє запис з бази даних. Якщо не вдається, то перенаправляє на головну сторінку.

Кінець таблиці 2.19

1	2
Альтернативні потоки	Немає
Постумова	Якщо варіант використання виконався успішно, то дані розкладу буде видалено з бази даних. А користувача ПЗ перенаправить на сторінку з переліком розкладу.

Таблиця 2.20 – Специфікація варіанту використання “Додати оцінку”

Складова	Опис
Опис прецеденту	Призначений для додавання оцінки до бази даних
Дійові особи прецеденту	Адміністратор, Завуч, Вчитель
Передумови	Адміністратор/Завуч/Вчитель авторизований у системі й знаходиться на Головній сторінці (сторінці, що зустрічає його при успішній авторизації), на цій сторінці перед собою бачить перелік учнів
Базовий потік	– Користувач обирає зі списку учнів учня та обирає опцію “Детальніше” – Система відображає на екрані більш детальну інформацію про обраного учня, включаючи його фотографію та кнопки – Користувач обирає опцію на кнопку “Оцінки учня” – Система відображає на екрані перелік оцінок учня – Користувач обирає опцію “Додати оцінку” – Система відображає список предметів – Користувач обирає предмет зі списку й обирає опцію “Обрати” для обраного предмету – Система відображає список вчителів, що викладають обраний предмет – Користувач обирає вчителя зі списку й обирає опцію “Обрати” для обраного вчителя – Система відображає форму вводу даних оцінки – Користувач заповнює поля форми й обирає опцію “Зберегти” – Система додає новий запис до бази даних. Якщо не вдається, то перенаправляє на головну сторінку.
Альтернативні потоки	– Користувач обирає зі списку предметів предмет й натискає “Учні” – Система відображає на екрані перелік учнів – Користувач обирає учня зі списку й обирає “Поставити оцінку” – Система відображає форму вводу даних оцінки – Користувач заповнює поля форми й обирає опцію “Зберегти” – Система додає новий запис до бази даних. Якщо не вдається, то перенаправляє на головну сторінку.
Постумова	Якщо варіант використання виконався успішно, то оцінка буде додана до бази даних. А користувача ПЗ перенаправить на сторінку з переліком оцінок учня. У випадку альтернативного потоку, користувача перенаправить на сторінку з переліком учнів.

Таблиця 2.21 – Специфікація варіанту використання “Редагувати оцінку”

Складова	Опис
Опис прецеденту	Призначений для зміни оцінки у базі даних
Дійові особи прецеденту	Адміністратор, Завуч, Вчитель
Передумови	Адміністратор/Завуч/Вчитель авторизований у системі й знаходиться на Головній сторінці (сторінці, що зустрічає його при успішній авторизації), на цій сторінці перед собою бачить перелік учнів
Базовий потік	– Користувач обирає зі списку учнів учня та обирає опцію “Детальніше” – Система відображає на екрані більш детальну інформацію про обраного учня, включаючи його фотографію, та кнопки – Користувач обирає опцію на кнопку “Оцінки учня” – Система відображає на екрані перелік оцінок учня – Користувач обирає оцінку зі списку й обирає опцію “Редагувати” навпроти обраної оцінки – Система відображає форму вводу даних оцінки, в котрій у полях знаходяться поточні значення – Користувач змінює поля форми й обирає опцію “Оновити” – Система зберігає змінений запис у базі даних. Якщо не вдається, то перенаправляє на головну сторінку.
Альтернативні потоки	Немає
Постумова	Якщо варіант використання виконався успішно, то дані оцінки буде змінено у базі даних. А користувача ПЗ перенаправить на сторінку з переліком оцінок учня.

Таблиця 2.22 – Специфікація варіанту використання “Видалити оцінку”

Складова	Опис
1	2
Опис прецеденту	Призначений для видалення оцінки з бази даних
Дійові особи прецеденту	Адміністратор, Завуч, Вчитель
Передумови	Адміністратор/Завуч/Вчитель авторизований у системі й знаходиться на Головній сторінці (сторінці, що зустрічає його при успішній авторизації), на цій сторінці перед собою бачить перелік учнів

Кінець таблиці 2.22

1	2
Базовий потік	– Користувач обирає зі списку учнів учня та обирає опцію “Детальніше” – Система відображає на екрані більш детальну інформацію про обраного учня, включаючи його фотографію, та кнопки – Користувач обирає опцію на кнопку “Оцінки учня” – Система відображає на екрані перелік оцінок учня – Користувач обирає оцінку зі списку й обирає опцію “Видалити” навпроти обраної оцінки – Система відображає попередження про те що дані цієї оцінки буде видалено назавжди. – Користувач обирає опцію “Видалити оцінку” – Система видаляє запис з бази даних. Якщо не вдається, то перенаправляє на головну сторінку.
Альтернативні потоки	Немає
Постумова	Якщо варіант використання виконався успішно, то дані оцінки буде видалено з бази даних. А користувача ПЗ перенаправить на сторінку з переліком оцінок учня.

Таблиця 2.23 – Специфікація варіанту використання “Додати підсумкову оцінку”

Складова	Опис
1	2
Опис прецеденту	Призначений для додавання підсумкової оцінки до бази даних
Дійові особи прецеденту	Адміністратор, Завуч
Передумови	Адміністратор/Завуч авторизований у системі й знаходиться на Головній сторінці (сторінці, що зустрічає його при успішній авторизації), на цій сторінці перед собою бачить перелік учнів
Базовий потік	– Користувач обирає зі списку учнів учня та обирає опцію “Детальніше” – Система відображає на екрані більш детальну інформацію про обраного учня, включаючи його фотографію, та кнопки – Користувач обирає опцію на кнопку “Підсумкові оцінки учня” – Система відображає на екрані перелік підсумкових оцінок учня – Користувач обирає опцію “Додати підсумкову оцінку” – Система відображає список предметів – Користувач обирає предмет зі списку й обирає опцію “Обрати” для обраного предмету – Система відображає список вчителів що викладають обраний предмет – Користувач обирає вчителя зі списку й обирає опцію “Обрати” для обраного вчителя – Система відображає форму вводу даних підсумкової оцінки – Користувач заповнює поля форми й обирає опцію “Зберегти” – Система додає новий запис до бази даних. Якщо не вдається, то перенаправляє на головну сторінку.

Кінець таблиці 2.23

1	2
Альтернативні потоки	Немає
Постумова	Якщо варіант використання виконався успішно, то підсумкова оцінка буде додана до бази даних. А користувача ПЗ перенаправить на сторінку з переліком підсумкових оцінок учня.

Таблиця 2.24 – Специфікація варіанту використання “Редагувати підсумкову оцінку”

Складова	Опис
Опис прецеденту	Призначений для зміни підсумкової оцінки у базі даних
Дійові особи прецеденту	Адміністратор, Завуч
Передумови	Адміністратор/Завуч авторизований у системі й знаходиться на Головній сторінці (сторінці, що зустрічає його при успішній авторизації), на цій сторінці перед собою бачить перелік учнів
Базовий потік	– Користувач обирає зі списку учнів учня та обирає опцію “Детальніше” – Система відображає на екрані більш детальну інформацію про обраного учня, включаючи його фотографію, та кнопки – Користувач обирає опцію на кнопку “Підсумкові оцінки учня” – Система відображає на екрані перелік підсумкових оцінок учня – Користувач обирає оцінку зі списку й обирає опцію “Редагувати” навпроти обраної підсумкової оцінки – Система відображає форму вводу даних підсумкової оцінки, в котрій у полях знаходяться поточні значення – Користувач змінює поля форми й обирає опцію “Оновити” – Система зберігає змінений запис у базі даних. Якщо не вдається, то перенаправляє на сторінку підсумкових оцінок учня.
Альтернативні потоки	Немає
Постумова	Якщо варіант використання виконався успішно, то дані підсумкової оцінки буде змінено у базі даних. А користувача ПЗ перенаправить на сторінку підсумкових оцінок учня.

Таблиця 2.25 – Специфікація варіанту використання “Видалити підсумкову оцінку”

Складова	Опис
1	2
Опис прецеденту	Призначений для видалення підсумкової оцінки з бази даних
Дійові особи прецеденту	Адміністратор, Завуч

Кінець таблиці 2.25

1	2
Передумови	Адміністратор/Завуч авторизований у системі й знаходиться на Головній сторінці (сторінці, що зустрічає його при успішній авторизації), на цій сторінці перед собою бачить перелік учнів
Базовий потік	– Користувач обирає зі списку учнів учня та обирає опцію “Детальніше” – Система відображає на екрані більш детальну інформацію про обраного учня, включаючи його фотографію, та кнопки – Користувач обирає опцію на кнопку “Підсумкові оцінки учня” – Система відображає на екрані перелік підсумкових оцінок учня – Користувач обирає підсумкову оцінку зі списку й обирає опцію “Видалити” навпроти обраної підсумкової оцінки – Система відображає попередження про те що дані цієї підсумкової оцінки буде видалено назавжди. – Користувач обирає опцію “Видалити підсумкову оцінку” – Система видаляє запис з бази даних. Якщо не вдається, то перенаправляє на головну сторінку.
Альтернативні потоки	Немає
Постумова	Якщо варіант використання виконався успішно, то дані підсумкової оцінки буде видалено з бази даних. А користувача ПЗ перенаправить на сторінку підсумкових оцінок учня

Таблиця 2.26 – Специфікація варіанту використання “Переглянути розклад”

Складова	Опис
Опис прецеденту	Призначений для перегляду учнем, свого розкладу
Дійові особи прецеденту	Учень
Передумови	Учень авторизований у системі й знаходиться на Головній сторінці (сторінці, що зустрічає його при успішній авторизації), на цій сторінці перед собою бачить перелік учнів
Базовий потік	– Користувач обирає опцію “Розклад занять” – Система звертається до бази даних, запитує інформацію й формує сторінку з розкладом учня. Якщо не вдається, то перенаправляє на головну сторінку.
Альтернативні потоки	Немає
Постумова	Якщо варіант використання виконався успішно, то користувач на екрані побачить розклад своїх занять

Таблиця 2.27 – Специфікація варіанту використання “Переглянути табель”

Складова	Опис
Опис прецеденту	Призначений для перегляду учнем, свого табелю успішності
Дійові особи прецеденту	Учень
Передумови	Учень авторизований у системі й знаходиться на Головній сторінці (сторінці, що зустрічає його при успішній авторизації), на цій сторінці перед собою бачить перелік учнів
Базовий потік	– Користувач обирає опцію “Табель успішності” – Система звертається до бази даних, запитує інформацію й формує сторінку з табелем успішності учня. Якщо не вдається, то перенаправляє на головну сторінку.
Альтернативні потоки	Немає
Постумова	Якщо варіант використання виконався успішно, то користувач на екрані побачить табель успішності

Таблиця 2.28 – Специфікація варіанту використання “Переглянути щоденник”

Складова	Опис
Опис прецеденту	Призначений для перегляду учнем, свого щоденнику
Дійові особи прецеденту	Учень
Передумови	Учень авторизований у системі й знаходиться на Головній сторінці (сторінці, що зустрічає його при успішній авторизації), на цій сторінці перед собою бачить перелік учнів
Базовий потік	– Користувач обирає опцію “Щоденник” – Система звертається до бази даних, запитує інформацію й формує сторінку з щоденником учня. Якщо не вдається, то перенаправляє на головну сторінку.
Альтернативні потоки	Немає
Постумова	Якщо варіант використання виконався успішно, то користувач на екрані побачить свій щоденник

Таблиця 2.29 – Специфікація варіанту використання “Формувати повний звіт щодо учня”

Складова	Опис
1	2
Опис прецеденту	Призначений для формування повного звіту щодо учня
Дійові особи прецеденту	Адміністратор
Передумови	Адміністратор авторизований в системі, знаходиться на головній сторінці, на цій сторінці перед собою бачить перелік учнів

Кінець таблиці 2.29

1	2
Базовий потік	– Користувач обирає зі списку учнів учня та обирає опцію “Детальніше” – Система відображає на екрані більш детальну інформацію про обраного учня, включаючи його фотографію, та кнопки – Користувач обирає опцію на кнопку “Сформувати повний звіт” – Система звертається до бази даних, запитує інформацію й формує повний звіт, у форматі .pdf, щодо обраного учня й відкриває його у новій вкладці браузеру.
Альтернативні потоки	Немає
Постумова	Якщо варіант використання виконався успішно, то буде сформований та відображений повний звіт щодо учня.

Таблиця 2.30 – Специфікація варіанту використання “Формувати медичний звіт”

Складова	Опис
Опис прецеденту	Призначений для формування медичного звіту
Дійові особи прецеденту	Медсестра, Адміністратор
Передумови	Медсестра/Адміністратор авторизована в системі, знаходиться на головній сторінці, на цій сторінці перед собою бачить перелік учнів
Базовий потік	– Користувач обирає зі списку учнів учня та обирає опцію “Медична картка” – Система відображає на екрані медичну картку учня – Користувач обирає опцію на кнопку “Медичний звіт” – Система звертається до бази даних, запитує інформацію й формує медичний звіт, у форматі .pdf, щодо обраного учня й відкриває його у новій вкладці браузеру.
Альтернативні потоки	– Користувач обирає зі списку учнів учня та обирає опцію “Детальніше” – Система відображає на екрані детальніше про учня включаючи його фотографію – Користувач обирає опцію на кнопку “Медичний звіт” – Система звертається до бази даних, запитує інформацію й формує медичний звіт, у форматі .pdf, щодо обраного учня й відкриває його у новій вкладці браузеру.
Постумова	Якщо варіант використання виконався успішно, то буде сформований та відображений медичний звіт.

Таблиця 2.31 – Специфікація варіанту використання “Формувати фінансову звітність учня”

Складова	Опис
Опис прецеденту	Призначений для формування фінансової звітності учня
Дійові особи прецеденту	Бухгалтер/Адміністратор
Передумови	Бухгалтер/Адміністратор авторизований в системі, на цій сторінці перед собою бачить перелік учнів
Базовий потік	– Користувач обирає зі списку учнів учня та обирає опцію “Детальніше” – Система відображає на екрані більш детальну інформацію про обраного учня, включаючи його фотографію, та кнопки – Користувач обирає опцію на кнопку “Фінансова звітність” – Система звертається до бази даних, запитує інформацію й формує фінансову звітність, у форматі .pdf, щодо обраного учня й відкриває його у новій вкладці браузеру.
Альтернативні потоки	– Користувач обирає зі списку учнів учня та обирає опцію “Детальніше” – Система відображає на екрані детальніше про учня включаючи його фотографію – Користувач обирає опцію на кнопку “Фінансова звітність” – Система звертається до бази даних, запитує інформацію й формує фінансову звітність, у форматі .pdf, щодо обраного учня й відкриває його у новій вкладці браузеру.
Постумова	Якщо варіант використання виконався успішно, то буде сформована та відображена фінансова звітність учня.

Таблиця 2.32 – Специфікація варіанту використання “Формувати табель успішності учня”

Складова	Опис
1	2
Опис прецеденту	Призначений для формування табелю успішності учня
Дійові особи прецеденту	Завуч/Адміністратор
Передумови	Завуч/Адміністратор авторизований в системі, знаходиться на головній сторінці, на цій сторінці перед собою бачить перелік учнів
Базовий потік	– Користувач обирає зі списку учнів учня та обирає опцію “Детальніше” – Система відображає на екрані більш детальну інформацію про обраного учня, включаючи його фотографію, та кнопки – Користувач обирає опцію на кнопку “Табель успішності” – Система звертається до бази даних, запитує інформацію й формує табель успішності, у форматі .pdf, щодо обраного учня й відкриває його у новій вкладці браузеру.

Кінець таблиці 2.32

1	2
Альтернативні потоки	Немає
Постумова	Якщо варіант використання виконався успішно, то буде сформований та відображений табель успішності учня.

Таблиця 2.33 – Специфікація варіанту використання “Виставити всім учням підсумкові оцінки”

Складова	Опис
Опис прецеденту	Призначений для виставлення всім учням підсумкових оцінок за поточний навчальний рік
Дійові особи прецеденту	Завуч/Адміністратор
Передумови	Завуч/Адміністратор авторизований в системі, знаходиться на будь-якій сторінці.
Базовий потік	– Користувач обирає “Додатково” – Система відображає на екрані сторінку з додатковими опціями – Користувач обирає опцію “Виставити всім учням підсумкові оцінки” – Система звертається до бази даних, й запитує додати всім учням підсумкові оцінки. Учням додаються підсумкові оцінки. Якщо не вдається, то перенаправляє на головну сторінку.
Альтернативні потоки	Немає
Постумова	Якщо варіант використання виконався успішно, то користувача перенаправить на сторінку з додатковими опціями.

Таблиця 2.34 – Специфікація варіанту використання “Виставити всім учням предмети у відповідність”

Складова	Опис
1	2
Опис прецеденту	Призначений для виставлення всім учням предмети у відповідність до їх року навчання та до поточного переліку предметів у базі даних
Дійові особи прецеденту	Завуч/Адміністратор
Передумови	Завуч/Адміністратор авторизований в системі, знаходиться на будь-якій сторінці.
Базовий потік	– Користувач обирає “Додатково” – Система відображає на екрані сторінку з додатковими опціями – Користувач обирає опцію “Виставити всім учням предмети у відповідність” – Система звертається до бази даних, й запитує привести усім учням зв'язки з предметами у відповідності до класу учня. Учням виставляються предмети що вони вивчають, у відповідності до їх класу. Якщо не вдається, то перенаправляє на головну сторінку.

Кінець таблиці 2.34

Альтернативні потоки	Немає
Постумова	Якщо варіант використання виконався успішно, то користувача перенаправить на сторінку з додатковими опціями.

Таблиця 2.35 – Специфікація варіанту використання “Змінити пароль”

Складова	Опис
Опис прецеденту	Призначений для зміни користувачем, паролю свого акаунту
Дійові особи прецеденту	Завуч/Адміністратор/Учень/Вчитель/Бухгалтер/Медсестра
Передумови	Завуч/Адміністратор/Учень/Вчитель/Бухгалтер/Медсестра авторизований в системі, знаходиться на будь-якій сторінці.
Базовий потік	– Користувач обирає “Додатково” – Система відображає на екрані сторінку з додатковими опціями – Користувач обирає опцію “Змінити пароль” – Система відображає сторінку для зміни пароля – Користувач вводить старий пароль, вводить новий й повторює введення нового паролю й обирає “Змінити пароль” – Система змінює пароль авторизованого користувача на новий вказаний користувачем. Якщо не вдається, то перенаправляє на головну сторінку.
Альтернативні потоки	Немає
Постумова	Якщо варіант використання виконався успішно, то користувача перенаправить на сторінку з додатковими опціями.

Таблиця 2.36 – Специфікація варіанту використання “Переглянути інструкцію користувача”

Складова	Опис
Опис прецеденту	Призначений для перегляду користувачем інструкції з використання програмного забезпечення
Дійові особи прецеденту	Завуч/Адміністратор/Учень/Вчитель/Бухгалтер/Медсестра
Передумови	Завуч/Адміністратор/Учень/Вчитель/Бухгалтер/Медсестра авторизований в системі, знаходиться на будь-якій сторінці.
Базовий потік	– Користувач обирає “Інструкція користувача” – Система у новому вікні відкриває інструкцію користувача
Альтернативні потоки	Немає
Постумова	Якщо варіант використання виконався успішно, то користувача перенаправить на сторінку з додатковими опціями. А у новому вікні відкриється інструкція користувача.

Виконавши розробку діаграми варіантів використання та специфікації варіантів використання було визначено як користувачі взаємодіють з програмним забезпеченням.

2.2 Архітектура програмного забезпечення

У таблиці 2.37 наведено порівняння впливу різних архітектурних стилів на атрибути якості для ПЗ “Автоматизація обліку навчального процесу приватної школи “CIA: свідомість, іновації, академічність””.

Таблиця 2.37 – Вплив архітектурних стилів на атрибути якості для ПЗ “Автоматизація обліку навчального процесу приватної школи “CIA: свідомість, іновації, академічність””

	Зручність	Супроводжуваність	Ефективність	Надійність	Переносимість
Архітектура шарів абстракцій	0	+2	-1	+1	+2
Архітектура потоків даних	-1	+1	-1...+1	-2	+2
Класна дошка	0	-2...+2	-1	0	0
Модель-Вид-Управління (MVC)	+2	-1	-1	0	-1

+2 – сильно сприяє атрибуту якості; +1 – добре підтримує атрибут якості; 0 – не впливає на атрибут якості; -1 – не підтримує атрибут якості; -2 – не сприятливий атрибуту якості

1) Архітектура шарів абстракцій

Зручність використання (0): Архітектура шарів розділяє систему на модулі (шари), кожен з яких відповідає за певний аспект роботи. Вона впливає на зручність лише побічно – через чітке відокремлення логіки, однак сама по собі не орієнтована на кінцевого користувача. Отже, її вплив на зручність нейтральний.

Супроводжуваність (+2): Чітке розділення логіки на шари полегшує роботу з кодом: кожен шар можна змінювати окремо, не впливаючи на інші. Можна оновити бізнес–логіку, не торкаючись шару представлення або доступу до даних. Це значно знижує ризик помилок при внесенні змін.

Ефективність (–1): Недоліком архітектури шарів є те, що кожен шар додає додаткові виклики (наприклад, виклики функцій між шарами), що може знижувати швидкість роботи. Запит від клієнта до бази даних повинен пройти через всі шари, навіть якщо частина з них просто перенаправляє дані.

Надійність (+1): Завдяки відокремленню шарів помилки в одному шарі не призводять до збоїв у всій системі, що підвищує загальну надійність. Помилка в інтерфейсі користувача не впливає на коректність роботи бізнес–логіки.

Переносимість (+2): Завдяки незалежності шарів (бізнес–логіка не залежить від специфіки конкретної платформи) систему легко адаптувати до нових платформ. Вебдодаток може бути перенесений на мобільну платформу, змінюючи лише шар представлення.

2) Архітектура потоків даних

Зручність використання (–1): У потоках даних користувачі можуть відчувати складність у взаємодії, оскільки структура потоків не орієнтована на кінцевого користувача. Якщо потоки складно спроектовані, це може зробити систему важкою для розуміння.

Супроводжуваність (+1): Логічна структура потоків полегшує їх обслуговування. Якщо один із компонентів відповідає лише за збір даних, його легко ізолювати і модифікувати без впливу на інші компоненти.

Ефективність (–1…+1): Ефективність залежить від реалізації: якщо потоки добре оптимізовані, система працює швидко. Однак надмірна складність потоків може знижувати продуктивність. Дублювання даних у різних потоках може уповільнити систему.

Надійність (–2): Потоки даних залежать один від одного. Помилка в одному потоці може призвести до помилок у всій системі. Якщо дані неправильно

обробляються на ранньому етапі, всі наступні обчислення будуть некоректними.

Переносимість (+2): Потоки даних відокремлені від конкретної платформи. Наприклад, можна перенести потоки з однієї бази даних на іншу, не змінюючи основну логіку.

3) Класна дошка

Зручність використання (0): Класна дошка не впливає прямо на взаємодію з кінцевим користувачем. Її мета – організація обміну даними між компонентами.

Супроводжуваність (–2…+2): Залежить від реалізації. Якщо класна дошка добре спроектована, її легко супроводжувати. Якщо всі компоненти дотримуються стандартного протоколу взаємодії. Але якщо протоколів багато або є складна логіка обробки, супроводжуваність ускладнюється.

Ефективність (–1): Затримки можуть виникати через складність обміну даними між компонентами. Кожен компонент повинен перевіряти дані на дошці, що займає час.

Надійність (0): Класна дошка є лише середовищем передачі даних, тому її вплив на надійність мінімальний.

Переносимість (0): Зазвичай класна дошка не прив'язана до конкретної платформи, тому її легко перенести.

4) Модель – вид – управління (MVC)

Зручність використання (+2): MVC чітко розділяє інтерфейс користувача, бізнес–логіку і обробку подій, що значно покращує зручність для кінцевого користувача. Легко змінити інтерфейс без впливу на бізнес–логіку.

Супроводжуваність (–1): Взаємодія між моделлю, видом і контролером може бути складною, особливо якщо система розростається.

Ефективність (–1): Розділення компонентів додає накладні витрати на синхронізацію даних, що може уповільнити систему.

Надійність (0): MVC не гарантує додаткової надійності, оскільки компоненти мають складну взаємодію.

Переносимість (–1): Компоненти MVC можуть бути тісно пов'язані із специфікою конкретної платформи або фреймворку, що ускладнює перенесення.

Ґрунтуючись на специфіці ПЗ “Автоматизація обліку навчального процесу приватної школи “CIA: свідомість, іновації, академічність””, можна зробити висновок що найважливіший атрибут якості для цього ПЗ – це зручність використання. Найважливішість цього критерію обумовлена тим, що програмою буде користуватись широкий спектр людей, а саме: робітники школи (медсестра, бухгалтер, завуч, директор), учні й представники учнів.

Розглянемо інші параметри якості по відношенню до розроблювального ПЗ “Автоматизація обліку навчального процесу приватної школи “CIA: свідомість, іновації, академічність”.

Супроводжуваність є атрибутом якості низької важливості, оскільки при плануванні проєкту було визначено, що супровід проєкту зведений до мінімуму й замовник не планує на майбутнє супровід проєкту іншими розробниками.

Ефективність є атрибутом якості нейтральної важливості, оскільки розроблювальне ПЗ потребує малого об'єму ресурсів, також при постановці задачі не поставлена за ціль оптимізована розробка під слабкі пристрої й також специфіка розроблювального ПЗ не потребує часової ефективності.

Надійність є атрибутом якості середньої важливості, оскільки розроблювальне ПЗ у цілому не має високих вимог по надійності, однак, при взаємодії з рахунками й оцінками учнів, бажана зменшена кількість збоїв.

Переносимість є атрибутом якості низької важливості, оскільки розраховується, що ПЗ буде розгорнуте у певному статичному оточенні, а також ПЗ матиме мало взаємодій з іншими програмами.

Ґрунтуючись на вище наведеному, доходимо до висновку, що найліпше для ПЗ “Автоматизація обліку навчального процесу приватної школи “CIA: свідомість, іновації, академічність” підходить MVC архітектура.

На рисунку 2.2 представлена діаграма компонентів ПЗ “Автоматизація обліку навчального процесу приватної школи “CIA: свідомість, іновації, академічність””. На рисунку 2.3 представлена діаграма розгортання ПЗ

“Автоматизація обліку навчального процесу приватної школи “CIA: свідомість, іновації, академічність”.

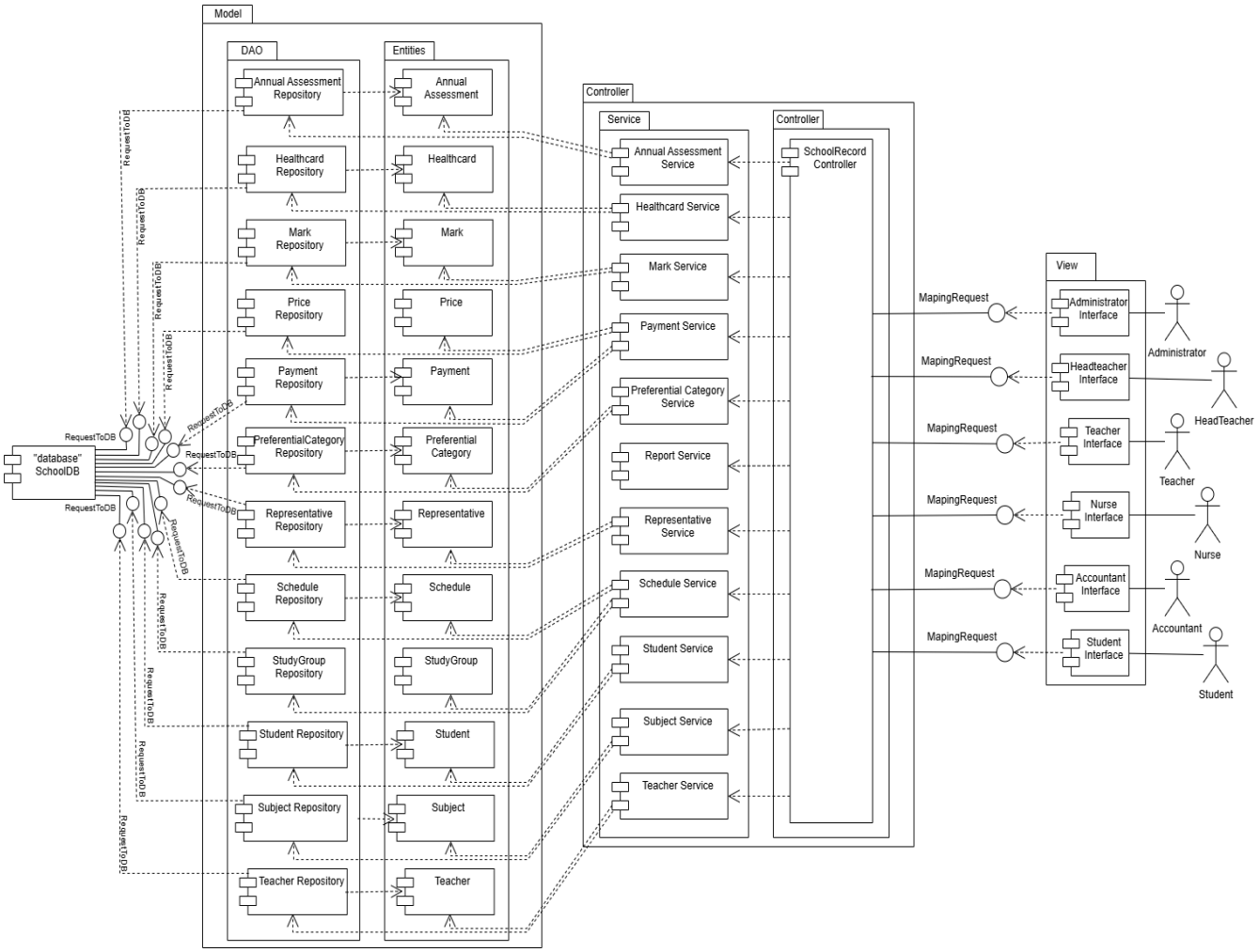


Рисунок 2.2 – Діаграма компонентів ПЗ “Автоматизація обліку навчального процесу приватної школи “CIA: свідомість, іновації, академічність”

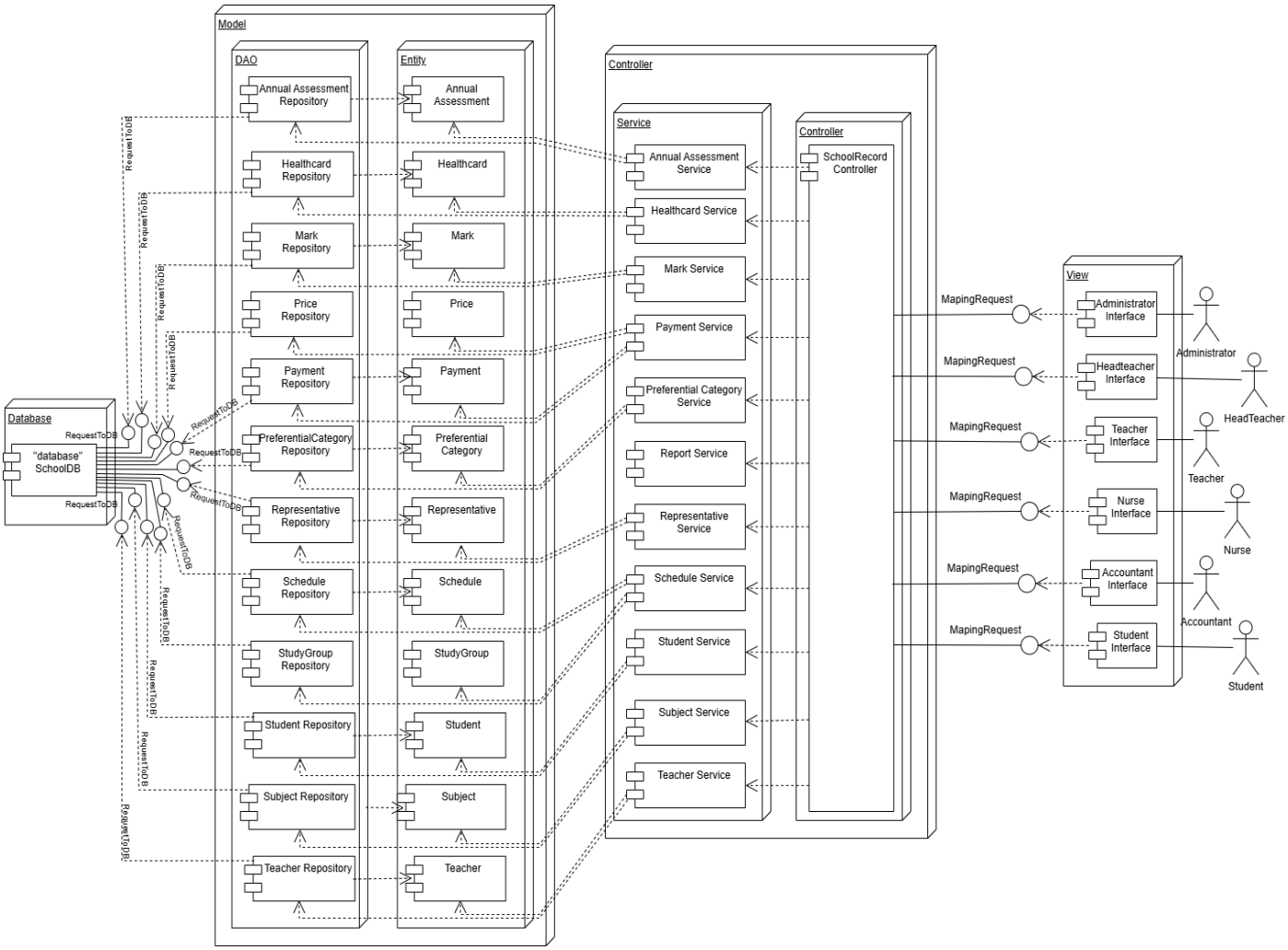


Рисунок 2.3 – Діаграма розгортання ПЗ “Автоматизація обліку навчального процесу приватної школи “СІА: свідомість, іновації, академічність”

Опис елементів:

View – відповідає за взаємодію між користувачами й сервером.

Nurse Interface – інтерфейс, що дає змогу користувачу, авторизованому як медсестра, взаємодіяти з сервером.

Administrator Interface – інтерфейс, що дає змогу користувачу, авторизованому як адміністратор, взаємодіяти з сервером.

Student Interface – інтерфейс, що дає змогу користувачу, авторизованому як учень, взаємодіяти з сервером.

Headteacher Interface – інтерфейс, що дає змогу користувачу, авторизованому як завуч, взаємодіяти з сервером.

Accountant Interface – інтерфейс, що дає змогу користувачу, авторизованому

як бухгалтер, взаємодіяти з сервером.

Teacher Interface – інтерфейс, що дає змогу користувачу, авторизованому як вчитель, взаємодіяти з сервером.

Controller – відповідає за обробку запитів.

Services – відповідає за бізнес-логіку.

AnnualAssessmentService – відповідає за логіку, пов'язану з таблицею Підсумкові оцінки.

HealthcardService – відповідає за логіку, пов'язану з таблицею Підсумкові оцінки Медичні Картки.

MarkService – відповідає за логіку, пов'язану з таблицею Підсумкові оцінки Оцінки.

PaymentService – відповідає за логіку, пов'язану з таблицями Рахунки та Розцінки.

PreferencialCategoryService – відповідає за логіку, пов'язану з таблицею Пільгові категорії.

RepresentativeService – відповідає за логіку, пов'язану з таблицею Представники.

ScheduleService – відповідає за логіку, пов'язану з таблицями Розклад, Навчальні групи.

StudentService – відповідає за логіку, пов'язану з таблицею Учні.

SubjectService – відповідає за логіку, пов'язану з таблицею Предмети.

TeacherService – відповідає за логіку, пов'язану з таблицею Вчителі.

Controller.Controller – приймає запити від View, викликає потрібні для бізнес-логіки Services й надає відповіді View

SchoolRecordController – є загальним контролером для усієї програми.

Model – відповідає за представлення даних.

DAO – (Data Access Object) відповідає за взаємодію з базою даних.

AnnualAssessmentRepository – відповідає за взаємодію з Базою Даних з таблицею Підсумкові оцінки.

HealthcardRepository – відповідає за взаємодію з Базою Даних з таблицею Медичні Картки.

MarkRepository – відповідає за взаємодію з Базою Даних з таблицею Оцінки.

PriceRepository – відповідає за взаємодію з Базою Даних з таблицею Розцінки.

PaymentRepository – відповідає за взаємодію з Базою Даних з таблицею Рахунки.

PreferencialCategoryRepository – відповідає за взаємодію з Базою Даних з таблицею Пільгові категорії.

RepresentativeRepository – відповідає за взаємодію з Базою Даних з таблицею Представники.

ScheduleRepository – відповідає за взаємодію з Базою Даних з таблицею Розклад.

StudyGroupRepository – відповідає за взаємодію з Базою Даних з таблицею Навчальні групи.

StudentRepository – відповідає за взаємодію з Базою Даних з таблицею Учні.

SubjectRepository – відповідає за взаємодію з Базою Даних з таблицею Предмети.

TeacherRepository – відповідає за взаємодію з Базою Даних з таблицею Вчителі.

Entities – відповідає за представлення таблиць бази даних.

AnnualAssessment – відповідає за представлення таблиці Підсумкові оцінки.

Healthcard – відповідає за представлення таблиці Медичні Картки.

Mark – відповідає за представлення таблиці Оцінки.

Price – відповідає за представлення таблиці Розцінки.

Payment – відповідає за представлення таблиці Рахунки.

PreferencialCategory – відповідає за представлення таблиці Пільгові

категорії.

Representative – відповідає за представлення таблиці Представники.

Schedule – відповідає за представлення таблиці розклад.

StudyGroup – відповідає за представлення таблиці Навчальні групи.

Student – відповідає за представлення таблиці Учні.

Subject – відповідає за представлення таблиці Предмети.

Teacher – відповідає за представлення таблиці Вчителі.

2.3 Проєкт програмного забезпечення “Автоматизація обліку навчального процесу приватної школи “CIA: свідомість, іновації, академічність””

2.3.1 Ескізний проєкт ПЗ

Діаграми діяльності ПЗ “Автоматизація обліку навчального процесу приватної школи “CIA: свідомість, іновації, академічність””

Діаграма діяльності варіанту використання “Додати оцінку” представлена на рисунку 2.4.

Діаграма діяльності варіанту використання “Додати учня” представлена на рисунку 2.5.

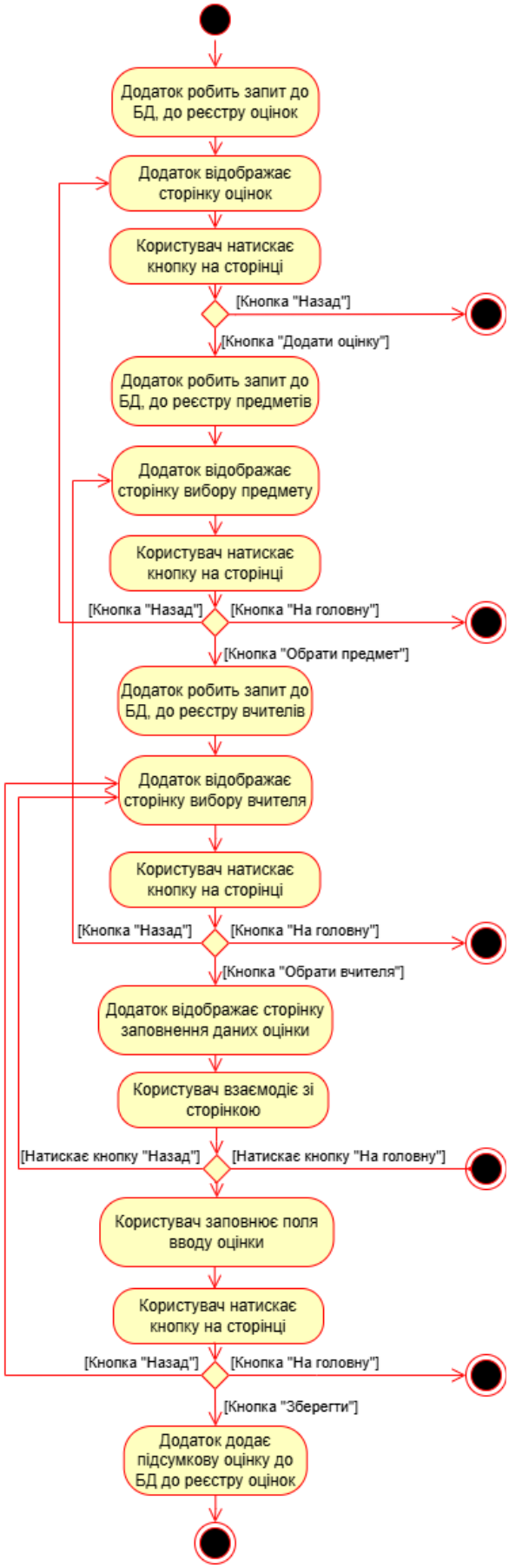


Рисунок 2.4 – Діаграма діяльності варіанту використання “Додати оцінку”

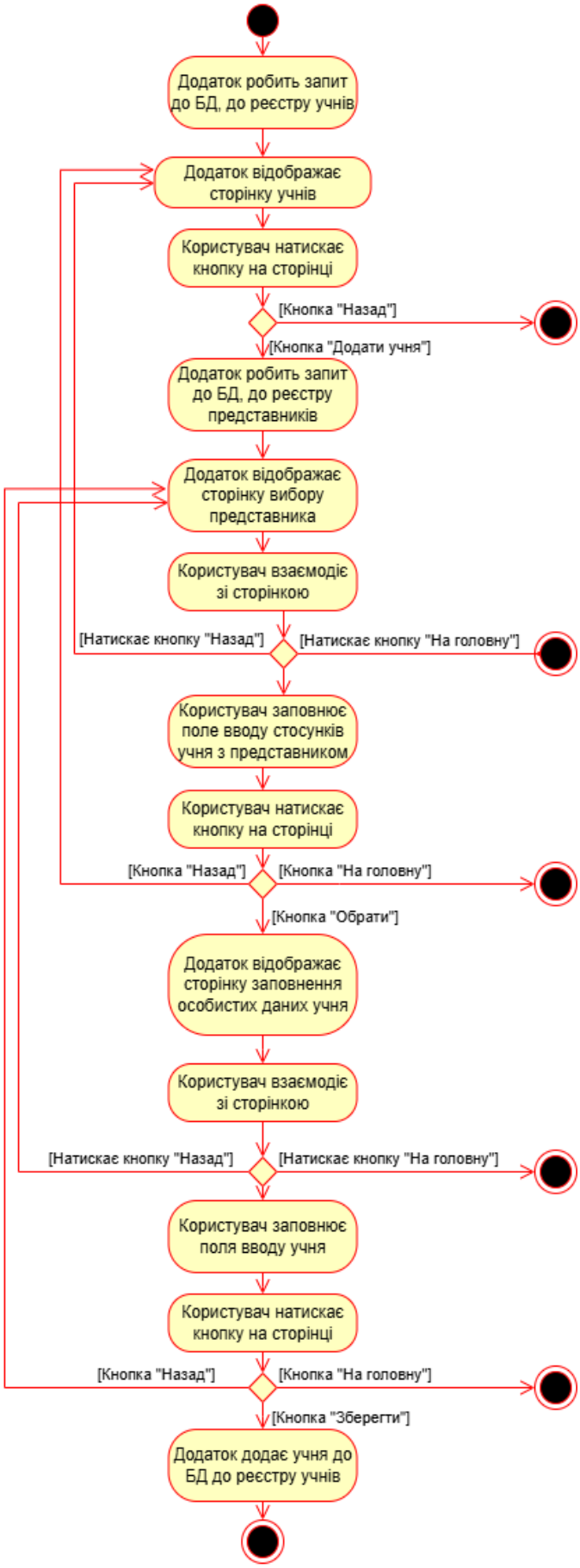


Рисунок 2.5 – Діаграма діяльності варіанту використання “Додати учня”

Побудувавши діаграми діяльності, було змодельовано сценарії варіантів використання.

Концептуальна модель даних ПЗ “Автоматизація обліку навчального процесу приватної школи “CIA: свідомість, іновації, академічність””

Визначення сутностей предметної галузі

Сутність – це деякий об'єкт реального світу. Вона має екземпляри, які відрізняються один від одного значеннями атрибутів.

В роботі можна виділити такі сутності:

– Student (учень) – є головним компонентом даної БД, у цій сутності зберігаються особисті дані учнів.

– Health_card (медична картка) – сутність, в котрій зберігається інформація, пов'язана зі здоров'ям учня (не може існувати без сутності Student).

– Representative (представник) – сутність, в котрій зберігається інформація, про представника учня.

– Subject (предмет) – сутність, в котрій зберігається інформація щодо предметів, що викладаються.

– Teacher (вчитель) – сутність, в котрій зберігається інформація щодо викладачів, що викладають предмети.

– Annual_assessment (підсумкова оцінка) – сутність, в котрій зберігається інформація щодо підсумкової оцінки учня (не може існувати без сутностей Student, Subject, Teacher).

– Preferential_category (пільгова категорія) – сутність, в котрій зберігається інформація щодо викладачів що викладають предмети.

– Payment (рахунок) – сутність, в котрій зберігається інформація щодо рахунків на сплату предметів (не може існувати без сутності Student).

– Mark (оцінка) – сутність, в котрій зберігається інформація щодо оцінок, що отримують по предметам учні (не може існувати без сутностей Teacher, Student, та Subject).

– Schedule (розклад) – сутність, в котрій зберігається інформація щодо

розкладу занять учнів (не може існувати без сутності Subject).

– Study_group (навчальна група) – сутність, котра зв'язує учнів та розклад.

– Prices (розцінки) – сутність, в котрій зберігається інформація щодо розцінок за навчання.

Визначення атрибутів сутностей

Сутність Student:

– id – індивідуальний ідентифікатор учня, котрий генерується автоматично при реєстрації учня в БД.

– name – ім'я учня;

– surname – прізвище учня;

– patronymic – по батькові учня;

– class – клас, в якому вчиться учень;

– phone – номер мобільного телефону учня;

– address – домашня адреса учня;

– current_location_address – адреса фактичного перебування;

– renewed_study – чи поновлене навчання у учня, тобто чи перевівся він з іншої школи;

– birthdate – дата народження учня;

– study_form – форма навчання учня;

– state – стан у базі даних учня, чи він все ще вчиться у школі, чи випустився, чи відрахувався;

– email – електронна пошта учня.

Сутність Health_card:

– id – індивідуальний ідентифікатор медичної картки, котрий генерується автоматично при додаванні нової медичної картки.

– gender – стать учня;

– medical_group – фізкультурна група учня

– weight – вага учня;

– height – зріст учня;

– update_date – дата останнього оновлення медичної картки.

Сутність Representative:

– id – індивідуальний ідентифікатор представника, котрий генерується автоматично при додаванні нового представника.

– name – ім'я представника;

– surname – прізвище представника;

– patronymic – по батькові представника;

– work_place – місце роботи представника;

– phone_number – номер мобільного телефону представника;

– email – електрона пошта представника.

Сутність Subject:

– id – індивідуальний ідентифікатор предмету, котрий генерується автоматично при додаванні нового предмета.

– name – назва предмету;

– type – тип предмету;

– study_year – рік навчання для котрого цей предмет.

Сутність Teacher:

– id – індивідуальний ідентифікатор вчителя, котрий генерується автоматично при додаванні нового вчителя.

– name – ім'я вчителя;

– surname – прізвище вчителя;

– patronymic – по батькові вчителя;

– phone_number – номер мобільного телефону вчителя;

– email – електронна пошта вчителя;

– address – домашня адреса вчителя;

– birthdate – дата народження вчителя;

– position – обіймана посада вчителя;

– direction – викладацька напрямленість вчителя;

– work_expirience – розмір робочого стажу;

– graduated_place – місце отримання фахової освіти.

Сутність Annual_assessment:

– id – індивідуальний ідентифікатор річної оцінки, котрий генерується автоматично при додаванні нової річної оцінки.

– year_grade – підсумкова оцінка;

– f_semester_grade – оцінка за перший семестр;

– s_semester_grade – оцінка за другий семестр;

– study_year – рік навчання за котрий було отримано підсумкову оцінку;

– calendar_years – календарні роки в котрі проходив рік навчання за котрий було отримано підсумкову оцінку.

Сутність Preferential_category:

– id – індивідуальний ідентифікатор пільгової категорії, котрий генерується автоматично при додаванні нової пільгової категорії;

– name – назва пільгової категорії;

– coefficient – коефіцієнт знижки для цієї пільгової категорії.

Сутність Payment:

– id – індивідуальний ідентифікатор рахунку, котрий генерується автоматично при додаванні нової рахунку.

– document_number – номер документу;

– status – статус рахунку;

– payment_formed – дата формування рахунку;

– payment_deadline_date – крайня дата сплати рахунку;

– payment_closing_date – фактична дата закриття рахунку;

– payment_study_year – рік навчання для котрого рахунок;

– price – вартість до сплати;

– price_with_discounts – вартість до сплати з врахуванням пільг.

Сутність Mark:

– date – дата поставлення оцінки;

– mark – чисельне значення оцінки.

Сутність Schedule:

– frequency – періодичність проведення заняття;

– weekday – день тижня проведення заняття;

– begin – час початку заняття;

– end – час закінчення заняття;

– study_room – номер кабінету де буде проходити заняття.

Сутність Study_group:

– id – ідентифікатор групи.

Сутність Prices:

– range_begin – нижня межа діапазону для котрого розцінка;

– range_end – верхня межа діапазону для котрого розцінка;

– price – вартість навчання.

У таблиці 2.38 наведений опис зв'язків між сутностями.

Таблиця 2.38 – Зв'язки між сутностями

Назва сутності	Назва сутності	Тип зв'язку	Пояснення
Student	Health_card	1;1	Одному учню належить одна медична картка
Student	Representative	M:N	Багато учнів мають багато представників
Student	Preferential_category	M:N	Багато учнів мають багато пільгових категорій
Student	Study_group	M:N	Багато учнів мають багато навчальних груп
Study_Group	Schedule	1:N	Одна навчальна група має багато розкладу
Student	Subject	M:N	Багато учнів мають багато предметів
Student	Annual_assessment	1:M	Один учень багато підсумкових оцінок
Annual_assessment	Teacher	M:1	Один вчитель має багато підсумкових оцінок
Annual_assessment	Subject	M:1	Один предмет має багато підсумкових оцінок
Subject	Teacher	M:N	Багато вчителів мають багато предметів
Subject	Payment	M:N	Багато предметів мають багато платіжок
Student	Mark	1:M	Один учень має багато оцінок
Schedule	Subject	M:1	Багато елементів розкладу мають один предмет
Mark	Teacher	M:1	Багато оцінок ставить один вчитель
Mark	Subject	M:1	Багато оцінок ставлять по одному предмету
Schedule	Teacher	M:1	Багато уроків проводить один вчитель

Побудова діаграми сутність – зв'язок

На основі минулих підпунктів робимо побудову діаграми сутність-зв'язок (рис. 2.6).

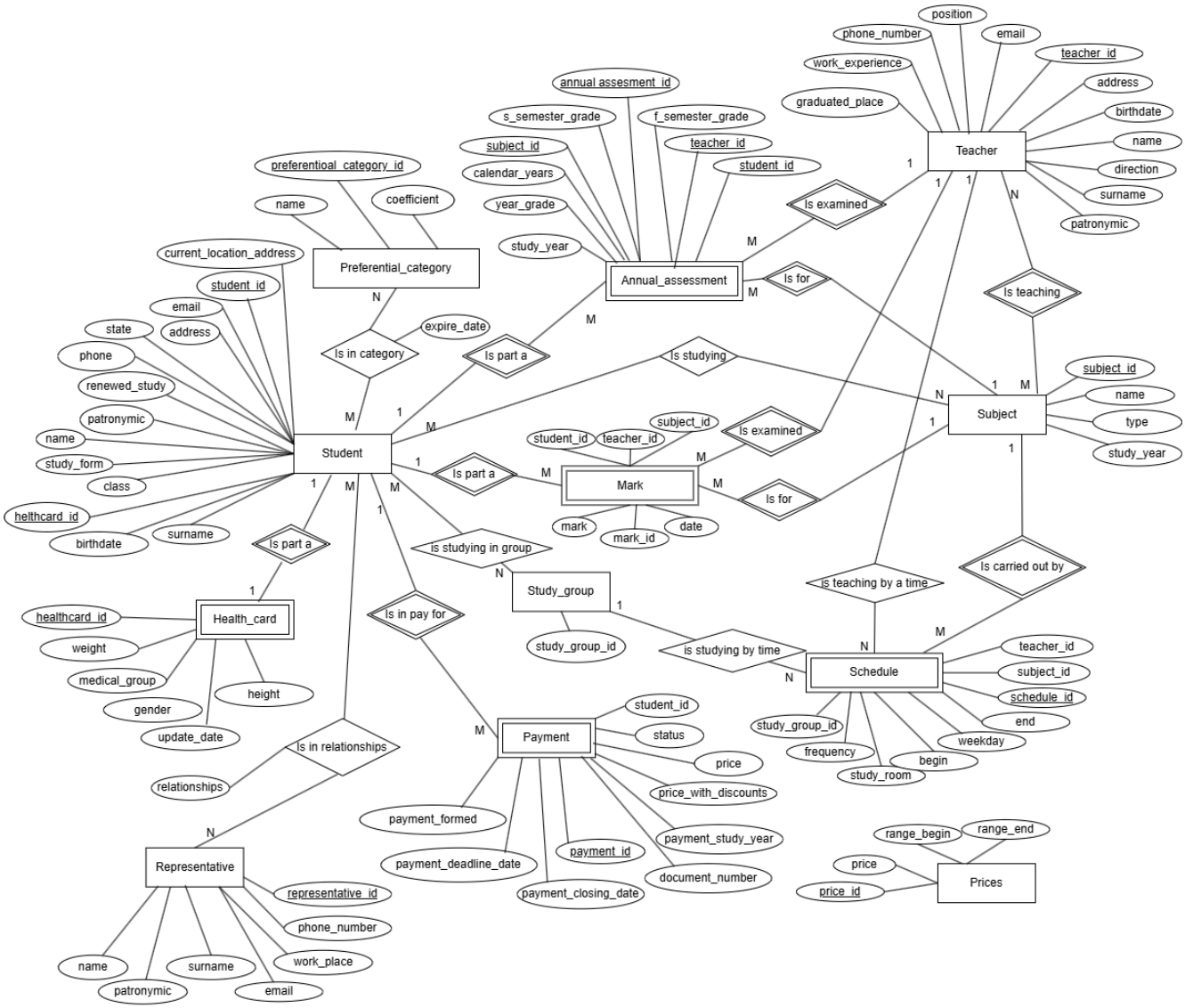


Рисунок 2.6 – Діаграма сутність зв'язок програмного забезпечення “Автоматизація обліку навчального процесу приватної школи “CIA: свідомість, іновації, академічність””

Розробивши концептуальну модель даних, було визначено які сутності матимуть які атрибути та які зв'язки з іншими сутностями.

Проєкт користувальницького інтерфейсу ПЗ “Автоматизація обліку навчального процесу приватної школи “CIA: свідомість, іновації, академічність”” На рисунках 2.7 – 2.9 представлені 3 варіанти ескізів графічного інтерфейсу.

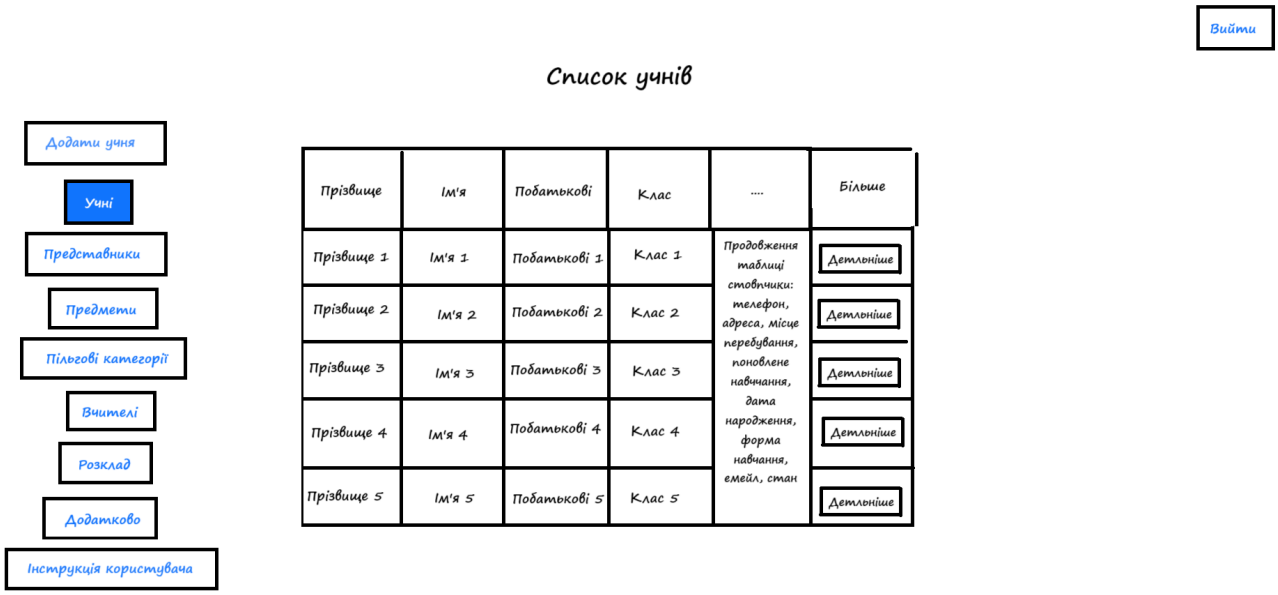


Рисунок 2.7 – Перший варіант ескізу графічного інтерфейсу сайту

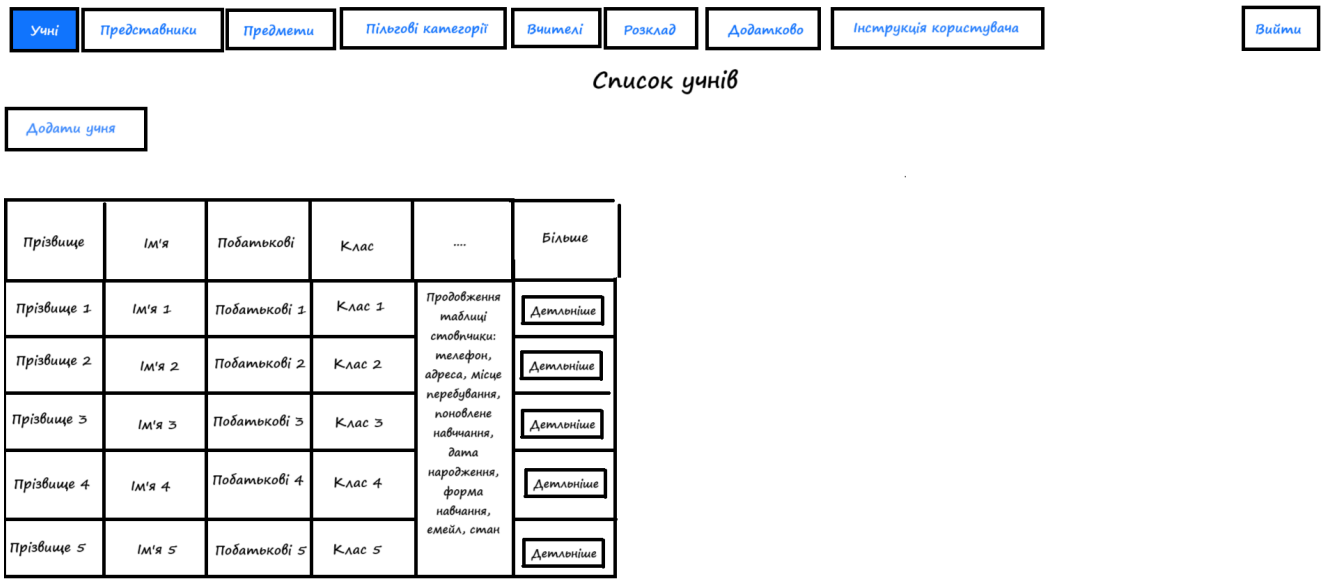


Рисунок 2.8 – Другий варіант ескізу графічного інтерфейсу сайту

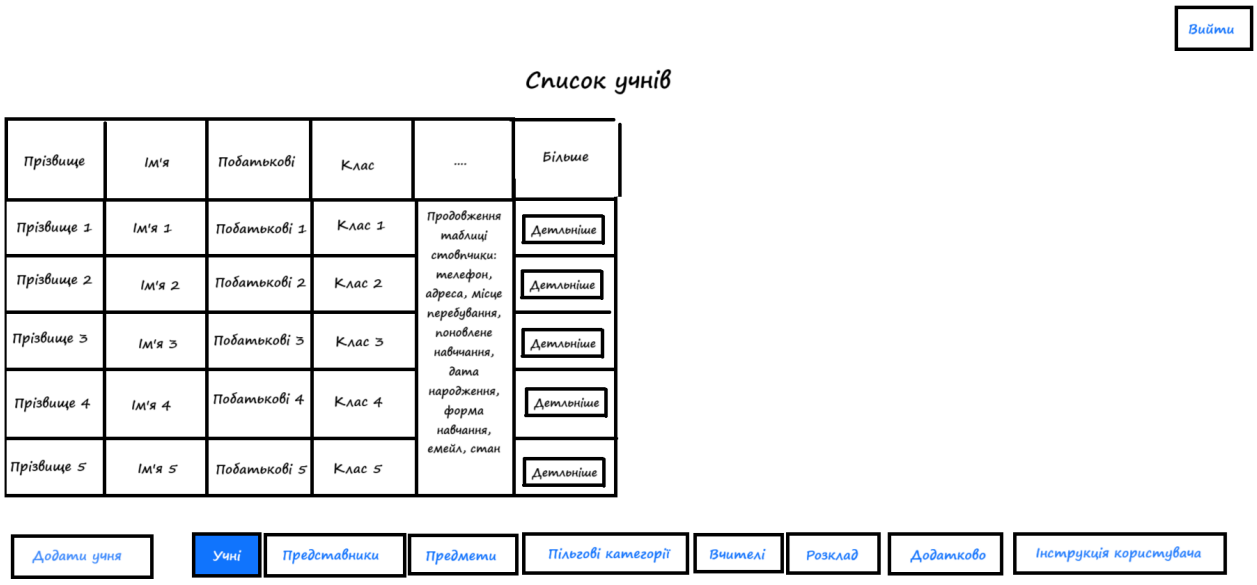


Рисунок 2.9 – Третій варіант ескізу графічного інтерфейсу сайту

Найбільш вдалим є другий варіант, так як на відміну від інших двох, він має найбільш вдале розташування таблиці по відношенню до інших елементів. Таблиця має багато полів (тобто треба більше місця в горизонталі), отже перший варіант гірший за інші два, бо у ньому кнопки займають частину місця, котру могла б займати таблиця. Також, оскільки це таблиця даних з БД, то при використанні буде збільшуватись кількість записів в цій таблиці, отже третій варіант, з кнопками знизу, не підходить, оскільки до цих кнопок буде важко дістатись.

Підбір кольору для оформлення фону сайту приведено на рисунках 2.10-2.12.

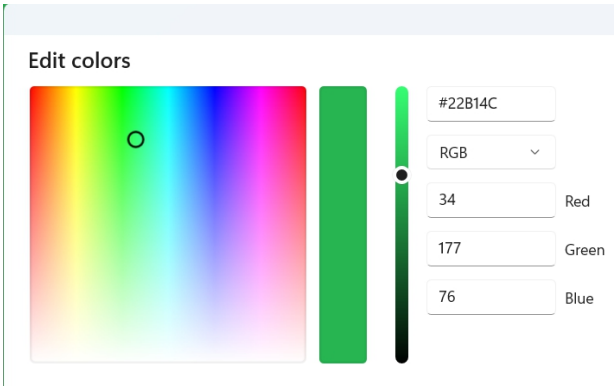


Рисунок 2.10 – Перший варіант кольору для оформлення сайту

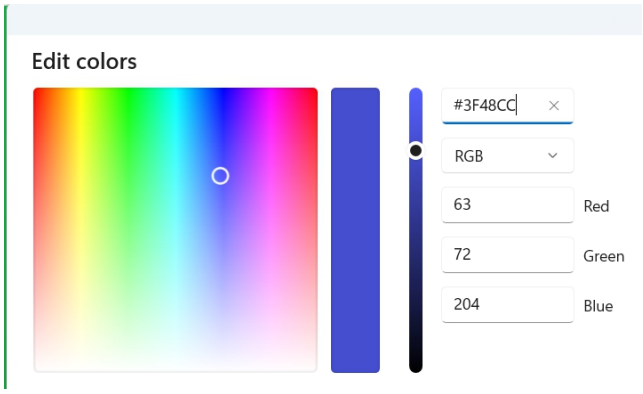


Рисунок 2.11 – Другий варіант кольору для оформлення сайту

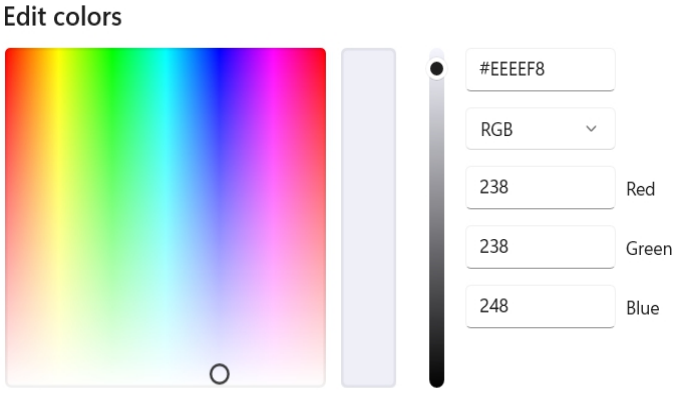


Рисунок 2.12 – Третій варіант кольору для оформлення сайту

Найбільш вдалим є третій варіант кольору, так як він не дуже яскравий, приємний для зору. #EEEEF8 – загальний колір фону (світло-сірувато-блакитний), #E6E6FA – колір фону головного меню (лавандовий).

Остаточний варіант інтерфейсу представлений на рисунку 2.13.

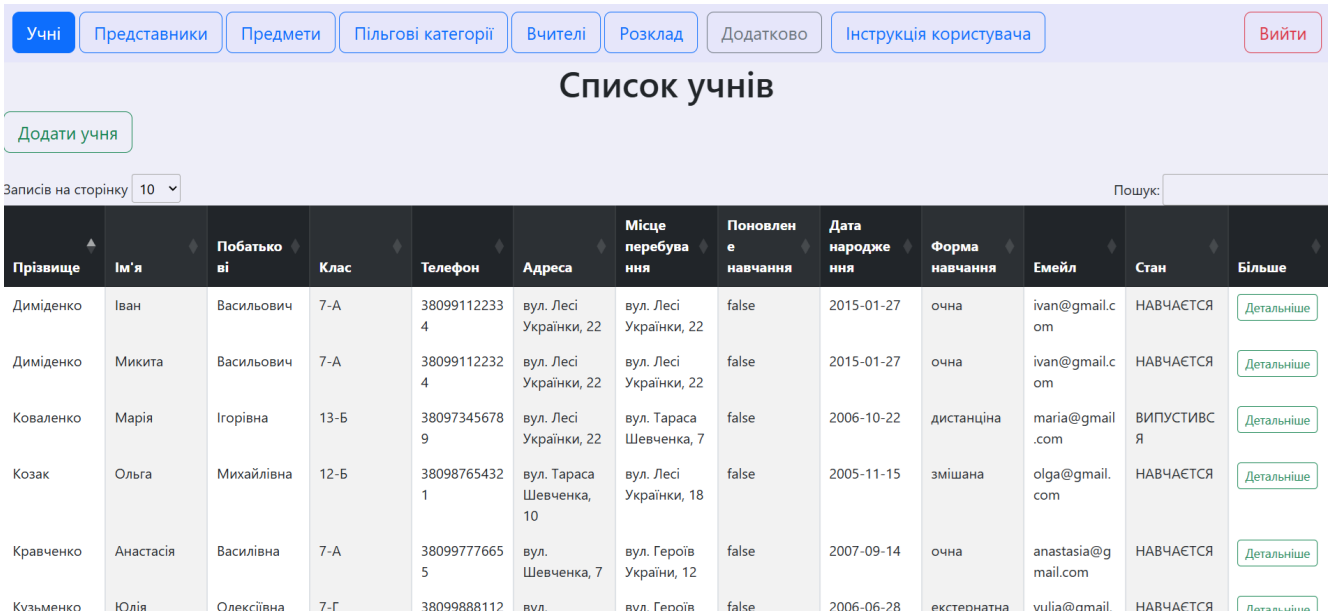


Рисунок 2.13 – Остаточний вигляд інтерфейсу ПЗ “Автоматизація обліку навчального процесу приватної школи “CIA: свідомість, іновації, академічність”””

Надалі проєкт інтерфейсу буде врахований при реалізації програмного забезпечення.

2.3.2 Технічний проєкт ПЗ

Статична модель ПЗ “Автоматизація обліку навчального процесу приватної школи “CIA: свідомість, іновації, академічність”””

Статична модель ПЗ представлена у вигляді діаграми класів (рис. 2.14), на котрій зображено зв'язки класів з двох пакетів: controllers та services.

Також у ПЗ присутні й інші класи, які не наведено на діаграмі класів, пакети: entities, dto, dao. Пакет entities – класи є представленням однойменних таблиць БД, поля класів поставлені у відповідність полям(стовпчикам) таблиць, а самі по собі, класи мають між собою такі ж зв'язки, як мають у БД, з методів ці класи мають тільки гетери й сетери. Пакет dao – data access object, має у собі інтерфейси, що наслідуються від Spring JpaRepository, використовуються для взаємодії з БД, по одному інтерфейсу для кожного класу з пакету entities. Пакет dto – data transfer object використовується у комунікації між класами.

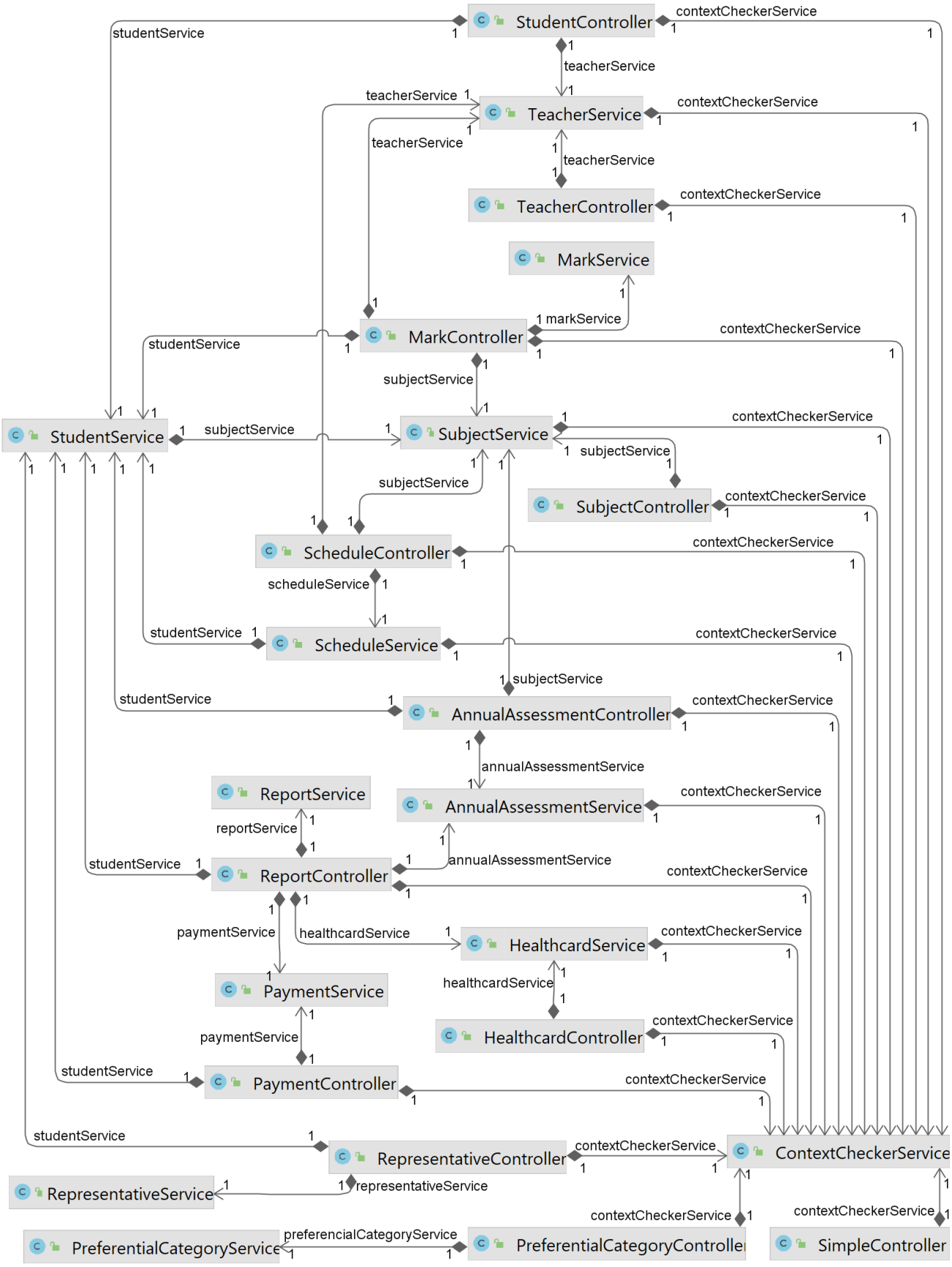


Рисунок 2.14 – Діаграма класів ПЗ “Автоматизація обліку навчального процесу приватної школи “CIA: свідомість, іновації, академічність””

Специфікації класів ПЗ “Автоматизація обліку навчального процесу приватної школи “CIA: свідомість, іновації, академічність””

Наведемо опис класів

Клас StudentController, використовується для обробки запитів з фронтенду, які пов'язані з таблицею “Учні” у БД.

Клас TeacherController, використовується для обробки запитів з фронтенду, які пов'язані з таблицею “Вчителі” у БД.

Клас MarkController, використовується для обробки запитів з фронтенду, які пов'язані з таблицею “Оцінки” у БД.

Клас SubjectController, використовується для обробки запитів з фронтенду, які пов'язані з таблицею “Предмети” у БД.

Клас ScheduleController, використовується для обробки запитів з фронтенду, які пов'язані з таблицею “Розклад” у БД.

Клас AnnualAssessmentController, використовується для обробки запитів з фронтенду, які пов'язані з таблицею “Підсумкові оцінки” у БД.

Клас HealthcardController, використовується для обробки запитів з фронтенду, які пов'язані з таблицею “Медичні картки” у БД.

Клас PaymentController, використовується для обробки запитів з фронтенду, які пов'язані з таблицею “Рахунки” у БД.

Клас RepresentativeController, використовується для обробки запитів з фронтенду, які пов'язані з таблицею “Представники” у БД.

Клас PreferencialCategoryController, використовується для обробки запитів з фронтенду, які пов'язані з таблицею “Пільгові категорії” у БД.

Клас SimpleController, використовується для обробки запитів з фронтенду що не потребують взаємодії з БД.

Клас ReportController, використовується для обробки запитів з фронтенду що відповідають за формування звітності.

Клас StudentService, використовується для бізнес логіки й запитів до таблиці “Учні” у БД.

Клас TeacherService, використовується для бізнес логіки й запитів до

таблиці “Вчителі” у БД.

Клас MarkService, використовується для бізнес логіки й запитів до таблиці “Оцінки” у БД.

Клас SubjectService, використовується для бізнес логіки й запитів до таблиці “Предмети” у БД.

Клас ScheduleService, використовується для бізнес логіки й запитів до таблиці “Розклад” у БД.

Клас AnnualAssessmentService, використовується для бізнес логіки й запитів до таблиці “Підсумкові оцінки” у БД.

Клас HealthcardService, використовується для бізнес логіки й запитів до таблиці “Медичні картки” у БД.

Клас PaymentService, використовується для бізнес логіки й запитів до таблиці “Рахунки” у БД.

Клас RepresentativeService, використовується для бізнес логіки й запитів до таблиці “Представники” у БД.

Клас PreferencialCategoryService, використовується для бізнес логіки й запитів до таблиці “Пільгові категорії” у БД.

Клас ContextCheckerService, використовується для логіки перевірки контексту, а саме: чи має користувач певну роль, який id має поточний користувач, який навчальний рік зараз.

Клас ReportService, використовується для логіки формування звітності.

Пакет controllers

У цьому пакеті розміщено класи, що вкінці назви мають “Controller”. Класи пакету можуть мати безліч методів. Методи відповідають за обробку сервером дій користувача, що пов'язані з ціллю класу.

Самі методи можуть мати будь-які назви, що дозволені правилами мови Java (але для більшої зрозумілості бажано використовувати назви, що описують за яку обробку відповідає метод), оскільки виклик цих методів відбувається не по їх назві, а через анотацію “@Mapping” (або альтернативні їй анотації). У класах

присутні методи для додавання/редагування/видалення записів з відповідних таблиць БД, для перегляду списків записів з БД, й для навігації по потрібним формам.

У кожного класу типу <назва таблиці>Controller є атрибут <назва таблиці>Service. Окрім таких атрибутів, також у класах, за потреби, присутні @Autowired атрибути. У всіх класів присутній @Autowired атрибут contextCheckerService – використовується для перевірки ролі користувача, чи він має право виконати запит.

Пакет services

У цьому пакеті розміщено класи, що вкінці назви мають “Service”. Класи пакету можуть мати безліч методів. Методи відповідають за логіку роботи програми й звертання до бази даних. Назви класів мають певну відповідність з назвами таблиць БД (наприклад StudentService оброблює запити, пов'язані з таблицею Учні), є виключення: ReportService для логіки, пов'язаної з генерацією звітів.

У класах присутні методи для додавання/редагування/видалення записів з відповідних таблиць БД, для перегляду списків записів з БД.

У кожного класу <назва таблиці>Service є атрибут <назва таблиці>Repository для звертання до відповідної таблиці БД. Окрім таких атрибутів, також у класах, за потреби, присутні @Autowired атрибути для інших репозитаріїв та інших сервісів.

Динамічна модель ПЗ “Автоматизація обліку навчального процесу приватної школи “CIA: свідомість, іновації, академічність””

Динамічна модель ПЗ “Автоматизація обліку навчального процесу приватної школи “CIA: свідомість, іновації, академічність”” представлена у вигляді діаграм послідовності.

Діаграма послідовності для варіанту використання “Додати підсумкову оцінку”, представлена на рисунку 2.15. Діаграма послідовності для варіанту використання “Додати учня”, представлена на рисунку 2.16.

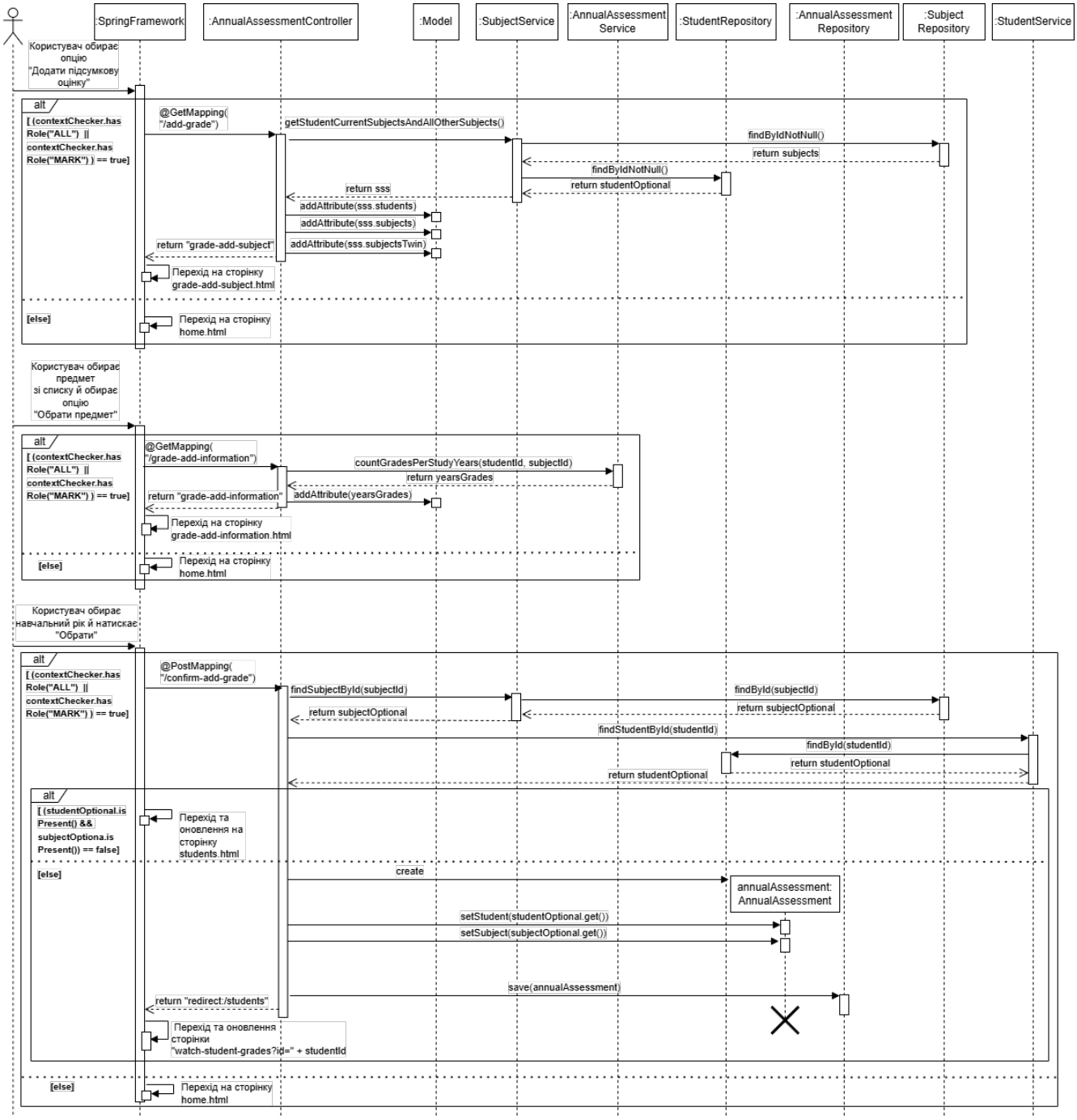


Рисунок 2.15 – Діаграма послідовності варіанту використання “Додати підсумкову оцінку”

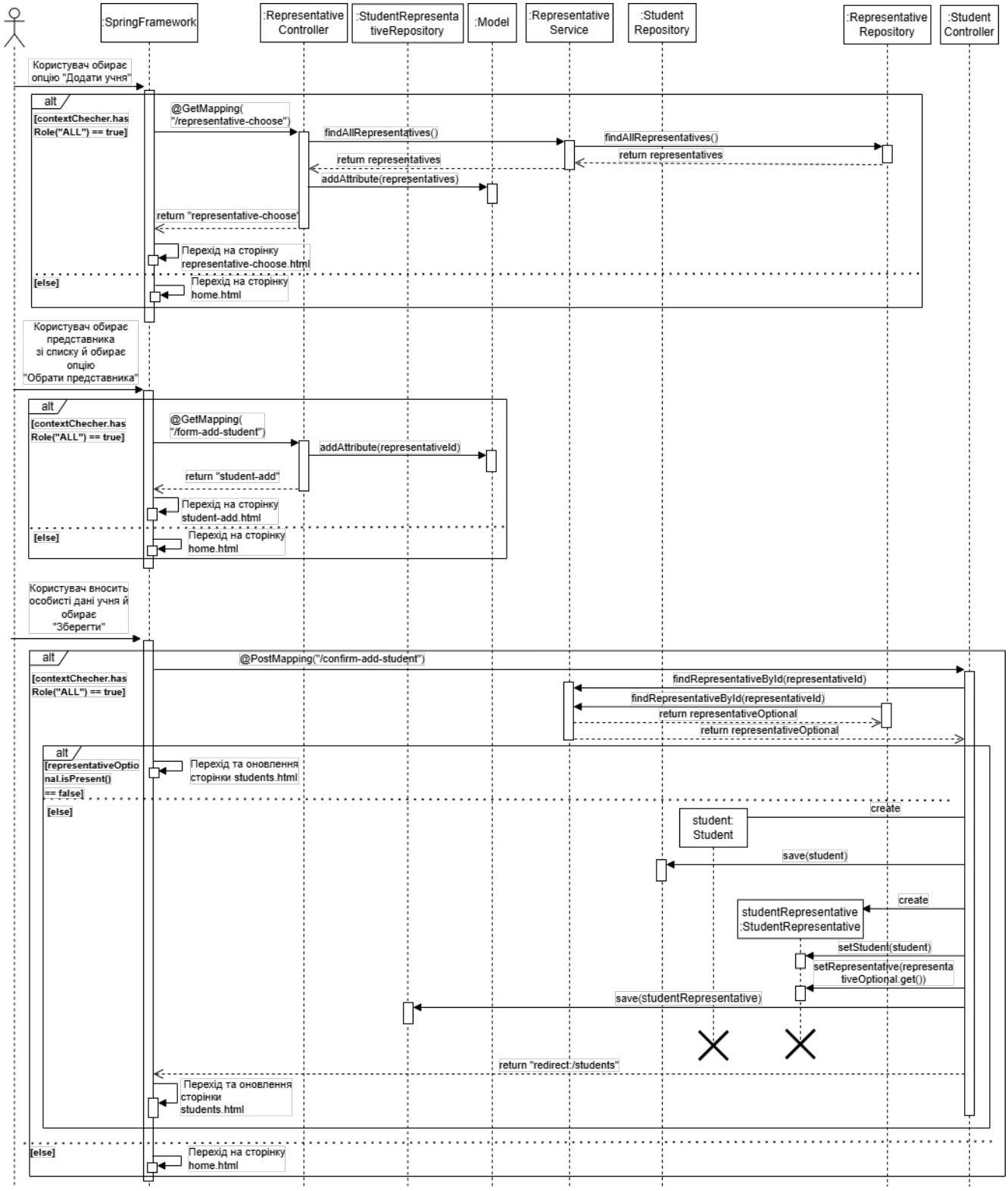


Рисунок 2.16 – Діаграма послідовності варіанту використання“Додати учня”

Логічна модель даних ПЗ “Автоматизація обліку навчального процесу приватної школи “CIA: свідомість, іновації, академічність””

Логічна модель даних представлена на рисунку 2.17.

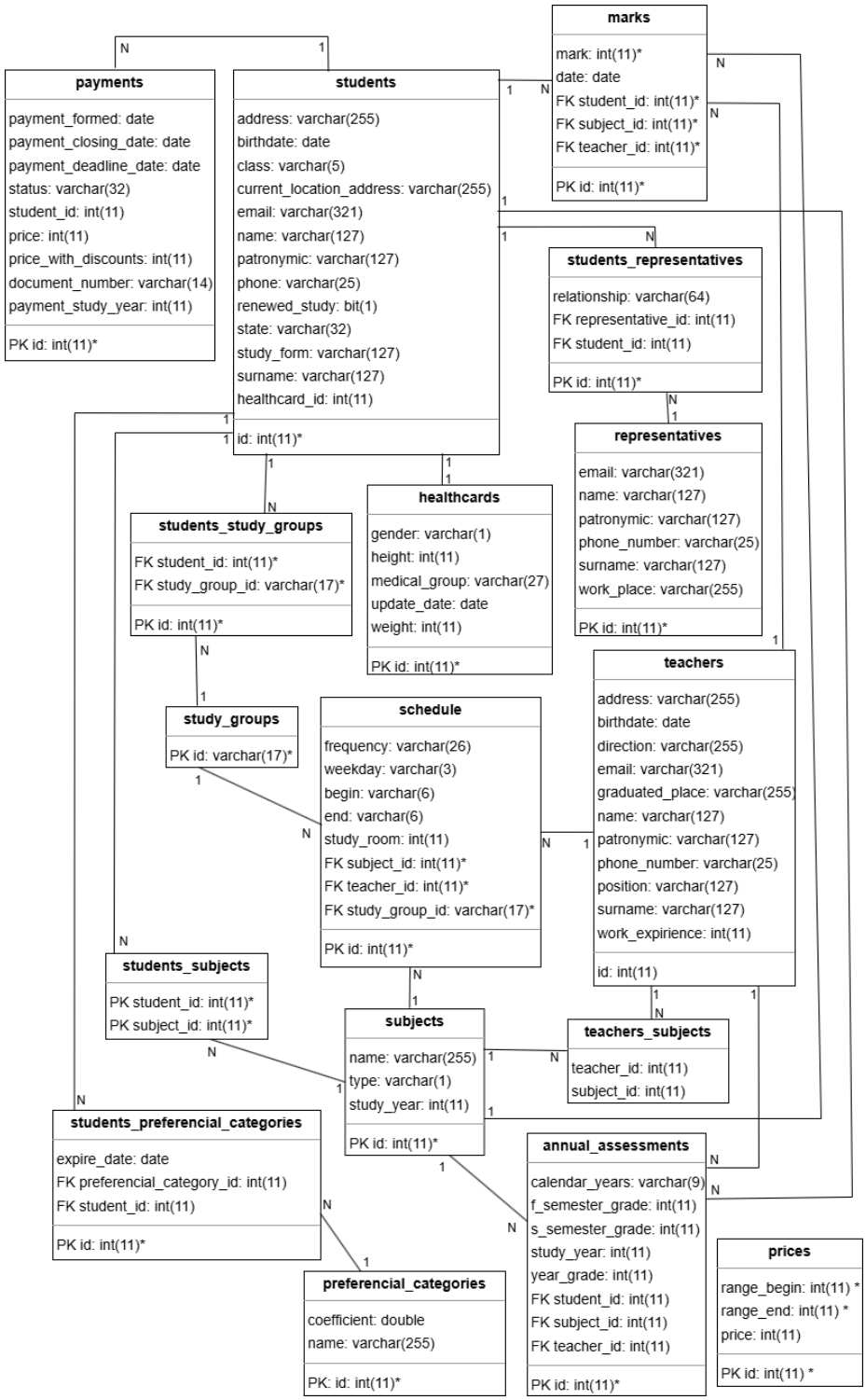


Рисунок 2.17 – Логічна модель даних ПЗ “Автоматизація обліку навчального процесу приватної школи “CIA: свідомість, іновації, академічність””

Розробивши логічну модель було визначено первинні й зовнішні ключі.

2.3.3 Робочий проєкт ПЗ

Обґрунтування вибори мови та системи програмування

Для розробки програмного забезпечення “Автоматизація обліку навчального процесу приватної школи “CIA: свідомість, іновації, академічність”” були обрані такі засоби розробки:

– Інтегроване середовище розробки **IntelliJ IDEA**

IntelliJ IDEA – це інтегроване середовище розробки (IDE), призначене для роботи з різними мовами програмування, зокрема Java, Kotlin, JavaScript, Python і багатьма іншими. Воно відоме своєю потужністю, ефективністю та широким спектром функцій, які полегшують процес розробки програмного забезпечення. IntelliJ IDEA надає розширену підтримку для автоматичного завершення коду, виявлення помилок, рефакторингу, аналізу коду та інших функцій, що допомагають розробникам писати якісний і надійний код. Воно також надає можливість роботи з системами контролю версій, засобами тестування та збірки проєктів. IntelliJ IDEA має інтуїтивний та дружній інтерфейс користувача, що сприяє зручності роботи з проєктами будь-якої складності. Воно підтримує також розширення плагінами, які дозволяють розширювати його функціональність та пристосовувати його до потреб розробника. IntelliJ IDEA використовується як професійними розробниками, так і початківцями, завдяки своїм продуктивним інструментам, які допомагають прискорити розробку та поліпшити якість коду. Воно є одним з найпопулярніших IDE для розробки програмного забезпечення і застосовується в різних галузях індустрії [8].

– Мова програмування **Java** 17 версії

Java – це потужна і популярна об'єктно-орієнтована мова програмування, яка використовується для розробки різноманітних програмних продуктів. Вона відома своєю платформовою незалежністю, що дозволяє виконувати програми на

різних операційних системах без необхідності змінювати їх код. Java забезпечує розширену функціональність, включаючи автоматичне управління пам'яттю, багатопотоковість, виключення, обробку вводу-виводу та багато інших вбудованих бібліотек і інструментів. Вона підтримує розробку як великих корпоративних застосунків, так і мобільних додатків та веб-сервісів. Java має широке співтовариство розробників, активну підтримку та велику кількість ресурсів для навчання і розвитку. Вона також є основною мовою для розробки програмного забезпечення для вбудованих систем, інтернету речей та інших технологій. Java просунута, стабільна і масштабована мова програмування, що забезпечує безпеку та надійність виконання коду. Вона знаходить застосування в багатьох сферах, включаючи фінанси, телекомунікації, медіа, ігрову індустрію та багато інших [9].

– Фреймворк **Spring MVC**

Spring MVC – це фреймворк для розробки веб-додатків на мові програмування Java, який базується на платформі Spring. Він надає розробникам потужні інструменти для побудови веб-додатків згідно з архітектурним шаблоном Model-View-Controller (MVC). Spring MVC дозволяє легко розділити бізнес-логіку, представлення та обробку запитів у веб-додатку. Він надає гнучкість у виборі шаблону представлення, підтримку валідації даних, керування сесіями та інтеграцію з іншими модулями Spring. Spring MVC є одним з найпопулярніших фреймворків для розробки веб-додатків у Java-середовищі, і його застосовують для створення різноманітних веб-проєктів різного розміру і складності [10].

– Парадигма **ООП**

ООП (**Об'єктно-орієнтоване програмування**) – це парадигма програмування, яка спрямована на створення програмних систем, орієнтованих на об'єкти. У ООП, програмний код організований у вигляді об'єктів, які взаємодіють між собою шляхом обміну повідомленнями. ООП базується на концепції класів і об'єктів. Класи визначають структуру та поведінку об'єктів, а об'єкти є конкретними екземплярами класів. Класи можуть мати властивості (поля) і методи (функції), які виконують певні дії. ООП надає переваги, такі як

модульність, перевикористання коду, поліморфізм і спадкування. Вона спрощує розробку програмних проєктів, полегшує їх обслуговування та розширення, а також забезпечує високу рівень абстракції і розбиття задач на менші компоненти. ООП є популярним підходом у багатьох мовах програмування, таких як Java, C++, Python і C#. Вона дозволяє розробникам ефективно створювати складні програми і працювати з великими обсягами даних [11].

– Збірка **Maven**

Maven – це інструмент для управління проєктами та збірки програмного забезпечення у Java-середовищі. Він надає структуровану систему для організації залежностей, компіляції, тестування та розгортання проєкту. Завдяки файлу конфігурації (pom.xml), Maven автоматизує процес залежностей, що дозволяє зручно керувати сторонніми бібліотеками та модулями. Він також надає засоби для виконання різних фаз розробки, таких як збірка, тестування, пакування і розгортання. Maven пропонує стандартизовану структуру проєкту, що дозволяє розробникам легко зорієнтуватись в коді і співпрацювати над проєктом. Він також забезпечує можливість автоматичного завантаження необхідних залежностей з централізованого репозиторію. Maven є популярним інструментом у спільноті Java-розробників завдяки своїй простоті використання, потужним функціям та можливостям інтеграції з іншими інструментами розробки. Він сприяє ефективній розробці програмного забезпечення та забезпечує стабільність і надійність проєктів [12].

– **HTML**, **CSS**, **JavaScript**, **Bootstrap**, **Thymeleaf**. Що використовуватимуться для написання фронтенту, що взаємодіятиме з користувачем.

HTML (**HyperText Markup Language**) – це стандартна мова розмітки, використовувана для створення веб-сторінок і веб-додатків. Вона визначає структуру і зовнішній вигляд веб-сторінки за допомогою тегів і елементів. HTML використовується для створення текстового контенту, заголовків, списків, таблиць, зображень, посилань та багатьох інших елементів, що дозволяють відображати інформацію на веб-сторінці. Вона також підтримує можливість

створення форм для введення даних і взаємодії з користувачем. HTML є основною мовою для створення веб-сторінок і використовується в поєднанні з CSS (Cascading Style Sheets) для задання стилів і JavaScript для додавання динамічного поведінки. Вона є стандартизованою і підтримується всіма сучасними веб-браузерами, що робить її універсальним і доступним інструментом для розробки веб-сайтів. HTML є важливою складовою для розуміння основ веб-розробки [13].

CSS (**Cascading Style Sheets**) – це мова, що використовується для оформлення сторінок веб-сайтів. Вона дозволяє відокремити зовнішній вигляд документу від його структури та змінювати відображення HTML-елементів на сторінці. CSS дозволяє задавати стилі для різних елементів документу, таких як шрифти, кольори, розміри, відступи, рамки та інші параметри. CSS можна застосовувати як вбудовано, так і зовнішньо до HTML-документів, що робить можливим використання однієї таблиці стилів для кількох сторінок сайту [14].

JavaScript (**JS**) – це високорівнева, об'єктно-орієнтована мова програмування, яка використовується для розробки динамічних веб-сайтів. Вона дозволяє створювати інтерактивні елементи, взаємодіяти з користувачем, маніпулювати вмістом сторінки та виконувати різноманітні операції в браузері. JavaScript може працювати як вбудований скрипт безпосередньо у HTML-коді сторінки або зовнішній файл, який підключається до сторінки. Вона має багатий набір вбудованих функцій та можливостей, що дозволяє розробникам створювати складні інтерфейси, валідувати дані, реалізовувати анімацію та багато іншого. JavaScript є однією з найпопулярніших мов програмування для веб-розробки та має широку підтримку серед сучасних браузерів [15].

Bootstrap – це фреймворк для створення адаптивних та стильних веб-інтерфейсів. Він надає розробникам набір готових компонентів і гнучку систему сіток, що дозволяє швидко створювати сучасні веб-додатки з привабливим дизайном. Bootstrap включає різноманітні елементи керування, такі як кнопки, форми, таблиці, модальні вікна, навігаційні панелі та багато іншого. Завдяки вбудованій підтримці CSS та JavaScript, а також адаптивному дизайну, сайти,

створені на Bootstrap, коректно відображаються на будь-яких пристроях [16].

Thymeleaf – це потужний шаблонізатор для Java, призначений для створення динамічних веб-інтерфейсів. Він легко інтегрується з Spring Boot і дозволяє розробникам будувати зручні та гнучкі веб-додатки з HTML. Thymeleaf підтримує різноманітні директиви для роботи зі змінними, умовами, циклами та фрагментами шаблонів, що спрощує розробку багаторазових компонентів. Завдяки серверному рендерингу, шаблонізатор забезпечує високу продуктивність і безпеку, дозволяючи ефективно працювати з даними без потреби в додатковому JavaScript. [17]

– СУБД **MariaDB**

MariaDB – це відкрита реляційна система управління базами даних (СУБД), яка є форком MySQL та пропонує вдосконалені функції та можливості. Вона зберігає сумісність з MySQL, що дозволяє легко перенести існуючі бази даних на MariaDB без змін у коді додатків. MariaDB пропонує покращену продуктивність, масштабованість та надійність, включаючи підтримку розподіленого виконання запитів та реплікації. Вона також підтримує широкий набір функцій, включаючи індекси, тригери, збережені процедури, транзакції та багато іншого. MariaDB активно розвивається та має активну спільноту розробників, яка забезпечує підтримку та вдосконалення цієї СУБД [18].

– Бібліотека **Lombok**.

Lombok – це бібліотека для мов програмування Java, яка спрощує розробку шляхом автоматичної генерації багатьох загальних методів і коду. Вона дозволяє розробникам зменшити кількість повторюваного коду, використовуючи анотації для автоматичного генерування гетерів, сетерів, конструкторів, методів equals() та hashCode(). Lombok також надає можливість автоматичної генерації методів toString(), @Builder для створення об'єктів за допомогою патерну "Builder" та інших корисних функцій. Використання Lombok дозволяє розробникам зосередитись на основній логіці додатка, зменшуючи кількість написаного коду і полегшуючи його читабельність. Це популярний інструмент у Java-розробці, який спрощує рутинну роботу і покращує продуктивність [19].

Spring MVC забезпечує архітектуру патерну Model – View - Controller (Модель–Відображення–Контролер) – структура для створення слабо пов'язаних веб-додатків, що розділяє основні аспекти їх розробки: об'єкти, бізнес-логіку та зовнішній вигляд програми. Основна перевага архітектури MVC — можливість міняти один із компонентів програми, суттєво не впливаючи на інші.

Розробка та тестування основних класів ПЗ “Автоматизація обліку навчального процесу приватної школи “СІА: свідомість, іновації, академічність””

Розглянемо бекенд частину функції “Додати розклад”

Призначення: додавання до БД новий розклад.

Учасники: завуч, адміністратор.

Передумови: авторизація у системі як завуч або адміністратор.

Сценарій:

1) Система відображає перелік предметів;

2) Користувач обирає основний предмет зі списку й натискає “Обрати”;

3) Система відображає перелік вчителів;

4) Користувач обирає вчителя зі списку й натискає “Обрати”;

5) Система відображає перелік класів;

6) Користувач обирає клас зі списку й натискає “Назначити клас”;

7) Система відображає форму вводу інформації про розклад;

8) Користувач заповнює поля форми й натискає “Зберегти”;

9) Система перевіряє поля форми на валідність. Якщо всі поля введено коректно, то система додасть розклад до бази даних. Якщо якесь поле було введено неправильно, то система відображає повідомлення про помилку. Якщо користувач виправить некоректно заповнені поля й ще раз натисне “Зберегти”, то система додасть розклад до бази даних.

Альтернативний сценарій:

1) Система відображає перелік предметів;

2) Користувач обирає додатковий предмет зі списку й натискає “Обрати”;

3) Система відображає перелік вчителів;

4) Користувач обирає вчителя зі списку й натискає “Обрати”;

5) Система відображає інформацію про кількість учнів що пов'язані з предметом;

6) Користувач обирає опцію “Назначити усім”;

7) Система відображає форму вводу інформації про розклад;

8) Користувач заповнює поля форми й натискає “Зберегти”;

9) Система перевіряє поля форми на валідність. Якщо всі поля введено коректно, то система додасть розклад до бази даних. Якщо якесь поле було введено неправильно, то система відображає повідомлення про помилку. Якщо користувач виправить некоректно заповнені поля й ще раз натисне “Зберегти”, то система додасть розклад до бази даних.

Частини коду контролеру та сервісу, що відповідають за функцію “Додавання розкладу”, наведено нижче. Повний код наведений у Додатку Б.

ScheduleController.java

```
package com.project.javaproject.controllers;
import com.project.javaproject.services.ContextCheckerService;
import com.project.javaproject.services.ScheduleService;
import lombok.AllArgsConstructor;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;
import java.time.LocalTime;
import java.util.List;
import java.util.Optional;
@Controller
@AllArgsConstructor
public class ScheduleController {
   @Autowired
   private ContextCheckerService contextCheckerService;
   private ScheduleService scheduleService;

@PostMapping("/add-schedule")
      public String addSchedule(@RequestParam int subjectId, @RequestParam int
teacherId, @RequestParam String classField,@RequestParam String frequency,
@RequestParam String weekday,@RequestParam LocalTime begin, @RequestParam
```

```
LocalTime end, @RequestParam int studyRoom){
   String url="home";
   if(contextCheckerService.hasRole("ALL") ||
contextCheckerService.hasRole("MARK")) {
       url="redirect:/schedule";
       scheduleService.addSchedule(subjectId, teacherId, classField, frequency,
weekday, begin, end, studyRoom);
   }
   return url;
}
```

ScheduleService.java

```
package com.project.javaproject.services;

import com.project.javaproject.dao.*;
import com.project.javaproject.dto.StudentScheduleDto;
import com.project.javaproject.dto.SubjectTeacherClassesShowtableDto;
import com.project.javaproject.entities.*;
import lombok.AllArgsConstructor;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.time.LocalTime;
import java.util.*;

@Service
@AllArgsConstructor
public class ScheduleService {

   private ScheduleRepository scheduleRepository;

   @Autowired
   private StudentRepository studentRepository;

   @Autowired
   private SubjectRepository subjectRepository;

   @Autowired
   private TeacherRepository teacherRepository;

   @Autowired
   private ContextCheckerService contextCheckerService;
```

```
    private StudentsStudyGroupRepository studentsStudyGroupRepository;

    private StudyGroupRepository studyGroupRepository;

    @Autowired
    private StudentService studentService;
       public void addSchedule(int subjectId, int teacherId, String classField,
String frequency, String weekday, LocalTime begin, LocalTime end, int studyRoom) {
    boolean isDataValid = validateScheduleInfo(frequency, weekday, begin, end,
studyRoom);
    Optional<Subject> optionalSubject = subjectRepository.findById(subjectId);
    Optional<Teacher> optionalTeacher = teacherRepository.findById(teacherId);
    if (optionalSubject.isPresent() && optionalTeacher.isPresent() && isDataValid)
{
        List<Student> students =
studentRepository.findByStudiedSubjects_Id(subjectId);
        Optional<StudyGroup> studyGroup;
        StudyGroup rStudyGroup;
        if(optionalSubject.get().getType().equals("д")) {
            studyGroup = studyGroupRepository.findById("Д" + subjectId);
        }
        else {
            studyGroup = studyGroupRepository.findById(classField);
            List<Student> students1 = new ArrayList<>();
            for(Student student:students) {
                if(!student.getClassField().equals(classField)) {
                    students1.add(student);
                }
            }
            students.removeAll(students1);
        }
        if(!studyGroup.isPresent()) {
            StudyGroup newStudyGroup = new StudyGroup();
            if(optionalSubject.get().getType().equals("о")) {
                newStudyGroup.setId(classField);
            }
            else {
                newStudyGroup.setId("Д" + subjectId);
            }
            StudyGroup savedGroup = studyGroupRepository.save(newStudyGroup);
            Set<StudentsStudyGroup> studyGroups = new HashSet<>();
            for(Student elem:students) {
                StudentsStudyGroup tmpSSG= new StudentsStudyGroup();
```

```
                tmpSSG.setStudent(elem);
                tmpSSG.setStudyGroup(savedGroup);
                StudentsStudyGroup savedTmpSSG =
studentsStudyGroupRepository.save(tmpSSG);
                studyGroups.add(savedTmpSSG);
            }
            savedGroup.setStudentsStudyGroups(studyGroups);
            rStudyGroup = studyGroupRepository.save(savedGroup);
        }
        else {
            rStudyGroup=studyGroup.get();
        }
        Schedule schedule = new Schedule();
        schedule.setFrequency(frequency);
        schedule.setWeekday(weekday);
        schedule.setBegin(begin.toString());
        schedule.setEnd(end.toString());
        schedule.setStudyRoom(studyRoom);
        schedule.setTeacher(optionalTeacher.get());
        schedule.setSubject(optionalSubject.get());
        schedule.setStudyGroup(rStudyGroup);
        Schedule schedule1 = scheduleRepository.save(schedule);
        Teacher teacher = optionalTeacher.get();
        teacher.getSchedule().add(schedule1);
        teacherRepository.save(teacher);
        Subject subject = optionalSubject.get();
        subject.getSchedule().add(schedule1);
        subjectRepository.save(subject);
        rStudyGroup.getSchedules().add(schedule1);
        studyGroupRepository.save(rStudyGroup);
    }
}

public boolean validateScheduleInfo(String frequency, String weekday, LocalTime
begin, LocalTime end, int studyRoom) {
    boolean isScheduleValid=true;
    if(studyRoom<0) isScheduleValid=false;
    if(!(frequency.equals("щотижня") || frequency.equals("парні тижні") ||
frequency.equals("непарні тижні"))) isScheduleValid=false;
    if(!(weekday.equals("Пн") || weekday.equals("Вт") || weekday.equals("Ср") ||
weekday.equals("Чт") || weekday.equals("Пт") || weekday.equals("Сб") ||
weekday.equals("Нд"))) isScheduleValid=false;
    return isScheduleValid;
```

```
}
```

Наведений код відповідає за додавання нового розкладу й виконується після того як користувач виконає 8-мий пункт сценарію.

Коли користувач натискає “Зберегти”, заповнивши форму вводу, то надсилається post-mapping запит “/add-schedule”, котрий Spring перенаправить на метод addSchedule в ScheduleController. Контролер викликає contextCheckerService.hasRole() для перевірки чи користувач, що надіслав запит, має право на виконання цього запиту (у конкретному випадку це перевірка чи має роль адміністратора або завуча). Якщо немає ролі, то користувача перенаправить на стартову сторінку. А якщо є, то перенаправить на сторінку з переліком розкладу. Але спочатку, контролер звернеться до ScheduleService.addSchedule() для того щоб додати розклад.

Перевірка на правильність вводу даних здійснюється ще на вебсторінці, тому можливість що користувач випадково надішле запит з неправильними даними зведена до мінімуму. Однак, є шляхи обходу цих перевірок, що є на html сторінках (якщо напряму відправити запит на сервіс), тому важливо мати перевірку даних й в бекенді. addSchedule() викликає функцію ScheduleService.validateScheduleInfo() котра перевіряє чи дані введено правильно.

Також робиться запит до SubjectRepository та TeacherRepository для перевірки чи є записи з наданими id. Якщо дані введено правильно й є такі записи, то продовжується алгоритм додавання розкладу.

При ініціалізації список учнів students, має записи учнів у котрі вивчають обраний предмет. Далі йде перевірка на існування навчальної групи по цьому предмету. Якщо optionalSubject.get().getType().equals("д"), тобто, обраний предмет є додатковим, то id навчальних груп формуються по принципу “Д”+ідентифікатор предмету, інакше, якщо предмет є основним, то id навчальної група шукається по наданому класу classField. Здійснюється пошук навчальної групи у репозиторії studyGroupRepository.findById(). У випадку якщо це предмет основний, то зі списку students виключаються всі учні, що не належать заданому classField.

Далі перевіряється чи вже існує навчальна група. Якщо групи не існує if(! studyGroup.isPresent()), то створюється нова група, котра матиме id номер класу, якщо ця група для основного предмету, або “Д”+ідентифікатор предмету, якщо ця група для додаткового предмету. Далі до цієї групи додаються учні. Й нова навчальна група й зв'язок з учнями зберігаються у БД.

Далі у програмі навчальна група використовується при додаванні нового розкладу. Створюється об'єкт schedule, котрому сетерами виставляються задані параметри. Цей об'єкт зберігається у БД у таблицю розклад. Після збереження нового запису, його пов'язують з таблицями предмети, вчителі та навчальні групи, й зберігають зміни БД.

Тестування було виконано методом білого ящику (структурного тестування). Під час тестування не було виявлено помилок. Тестування знаходиться у додатку Д.

3 РЕЗУЛЬТАТИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

В поданій кваліфікаційній роботі було розв'язано практичну задачу інженерії програмного забезпечення. Розв'язана задача характеризується комплексністю та невизначеністю умов. Для вирішення задачі було розроблено ПЗ “Автоматизація обліку навчального процесу приватної школи “CIA: свідомість, іновації, академічність””. Програмне забезпечення призначено для спрощення роботи працівників приватної школи “CIA: свідомість, іновації, академічність” й полегшення ведення обліку навчального процесу. На рисунку 3.1 наведена головна сторінка адміністратора.

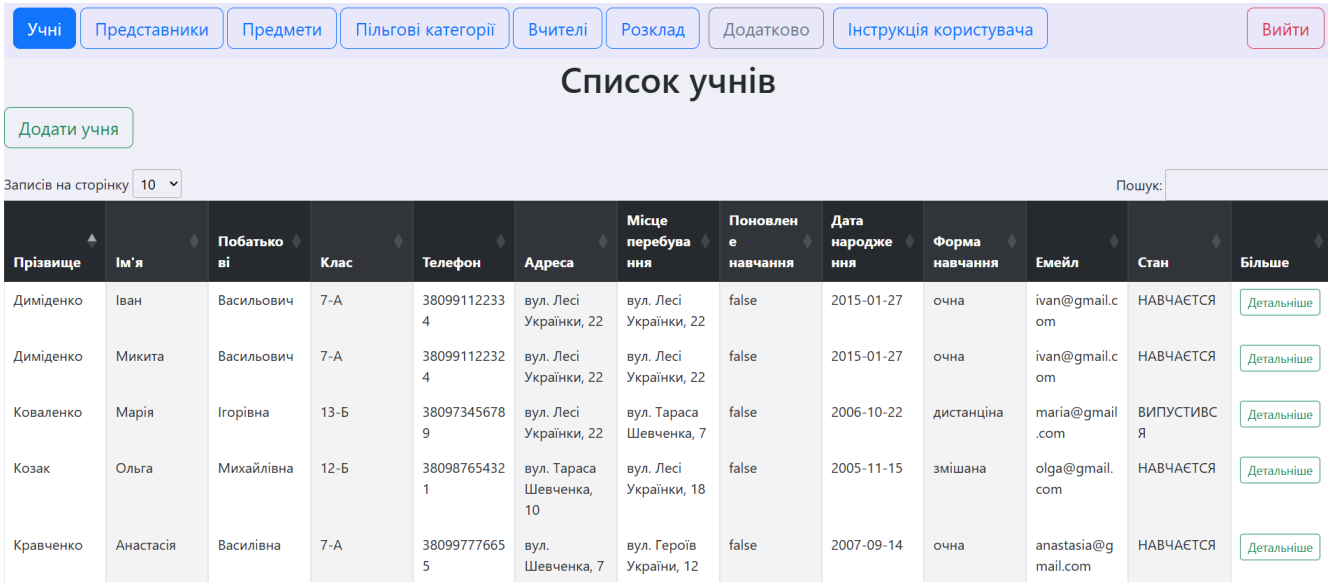


Рисунок 3.1 – Головна сторінка адміністратора при роботі з ПЗ “Автоматизація обліку навчального процесу приватної школи “CIA”

Під час виконання роботи було зроблено аналіз організації обліку навчального процесу у приватній школі “CIA: свідомість, іновації, академічність” й проведено аналіз існуючих аналогів. Розглянувши існуючі аналоги було знайдено, що для вирішення поставленої задачі, вони не є ефективними. Також зроблена постановка задачі на розробку програмного забезпечення, під час якої

визначено експлуатаційне та функціональне призначення розробки, також розроблено вимоги до функціональний характеристик, до надійності, до умов експлуатації, до складу і параметрів технічних засобів, до інформаційної та програмної сумісності, до програмної документації, а також узгоджено порядок контролю й приймання.

Далі зроблено аналіз вимог до програмного забезпечення шляхом розроблення засобами UML діаграми варіантів використання і написання специфікацій до цих варіантів використання. Визначено 6 користувачів системи: адміністратор, завуч, медсестра, учень, бухгалтер, вчитель.

Розглянуто архітектурні стилі, й обрано найбільш слушний для розроблювального ПЗ. Було обрано MVC (Model-View-Controller) архітектурний стиль і по ньому спроєктовано та описано архітектуру розроблювального ПЗ у вигляді діаграми компонентів та розгортання.

Розроблено ескізний, технічний і робочий проєкти.

У ескізному проєкті було побудовано концептуальну модель програмного застосунку, проєкт користувальницького інтерфейсу та змодельовано сценарії варіантів використання за допомогою діаграм діяльності.

У технічному проєкті було розроблено статичну модель програмного забезпечення у вигляді діаграми класів, й зроблено специфікацію класів. Також розроблено логічну модель даних й створено динамічну модель програмного забезпечення за допомогою діаграм послідовності.

У робочому проєкті було обрано й обґрунтовано засоби розробки: середовище розробки IntelliJ IDEA, мова програмування Java 17 версії, фреймворк Spring MVC, збірка Maven, СУБД MariaDB, HTML, CSS, JavaScript, Bootstrap, Thymeleaf, Lombok. Також розроблено програмне забезпечення і проведено тестування білої скриньки.

Також розроблено усю необхідну документацію: Додаток А Технічне завдання, Додаток Б Текст програми, Додаток В Опис програмного забезпечення, Додаток Г Інструкція користувача, Додаток Д Програма та методика випробування програмного забезпечення.

4 ОХОРОНА ПРАЦІ

4.1 Вступ

Охорона праці є одним із фундаментальних елементів соціальної політики держави та складовою частиною управління будь-якою організацією, установою чи підприємством. Її головною метою є збереження життя, здоров'я, працездатності та добробуту працівників у процесі їхньої професійної діяльності. Охорона праці – це система правових, соціально-економічних, технічних, санітарно-гігієнічних і профілактичних заходів, яка спрямована на створення безпечних умов праці та попередження виробничого травматизму. В Україні питання охорони праці врегульовані на законодавчому рівні. Основним нормативно-правовим актом є Закон України «Про охорону праці» від 14.10.1992 р. № 2694-XII, який визначає основи державної політики в цій сфері. Закон встановлює обов'язки роботодавців щодо створення безпечних умов праці, права працівників на безпечну працю, порядок проведення інструктажів, навчання та атестації, а також питання фінансування заходів з охорони праці [20].

До законодавчої бази також входять Кодекс законів про працю України (КЗпП), Закон України «Про освіту», типове положення про порядок проведення навчання і перевірки знань з питань охорони праці, галузеві накази та інструкції [21].

Для закладів освіти, зокрема приватних шкіл, охорона праці має не менш важливе значення, ніж для підприємств промислового сектора. У навчальному процесі беруть участь діти, педагоги, адміністративний та технічний персонал, кожен з яких має право на безпечні умови праці та навчання. Оскільки діти є особливо вразливою категорією, питання безпеки в освітньому середовищі виходять на перший план. Наявність безпечних умов у навчальних кабінетах, майстернях, спортивних залах, на території школи – обов'язкова вимога, встановлена як загальнодержавними, так і галузевими стандартами.

Правова охорона праці в системі освіти передбачає, що керівник закладу освіти несе персональну відповідальність за створення та підтримання належного рівня безпеки праці й навчання. Це включає: призначення відповідальних осіб за охорону праці, організацію регулярних інструктажів (вступного, первинного, повторного), забезпечення засобами індивідуального захисту, розробку локальних інструкцій з охорони праці, планування евакуаційних заходів та навчальних тривог, а також створення умов для надання першої домедичної допомоги.

Охорона праці у закладі освіти – це не лише формальність, а комплексна та відповідальна діяльність, що має забезпечити безпеку всіх учасників освітнього процесу. Реалізація норм охорони праці базується на дотриманні законодавства, галузевих стандартів та принципів превентивної безпеки. Впровадження системного підходу до охорони праці дозволяє мінімізувати ризики, забезпечити відповідність законодавчим вимогам і створити безпечне освітнє середовище.

4.2 Санітарно-гігієнічні вимоги до середовища закладу освіти

Забезпечення належного санітарно-гігієнічного стану в закладі загальної середньої освіти є обов’язковим елементом системи охорони праці та важливою умовою збереження здоров’я всіх учасників освітнього процесу. До санітарно-гігієнічних показників належать такі чинники, як якість повітря, температура та вологість у приміщеннях, освітлення, рівень шуму, забезпечення чистоти, дотримання режимів провітрювання та санітарно-побутове облаштування.

Основні вимоги до санітарно-гігієнічних умов визначені Державними санітарними правилами і нормами влаштування, утримання загальноосвітніх навчальних закладів та організації навчально-виховного процесу, а також санітарними регламентами, затвердженими Міністерством охорони здоров’я України.

Температурний режим в навчальних приміщеннях має відповідати нормам:

– в класних кімнатах 17-20 °С,

– в майстернях по обробці металу і дерева 16-18 °С,

– в спортивному залі 15-17 °С,

– в роздягальнях при спортивному залі 19-23 °С,

– в актовому залі 17-20 °С,

– в бібліотеці 16-18 °С,

– в медичних кабінетах 21-23 °С,

– в рекреаціях 16-18 °С,

– в спальних приміщеннях 18-20 °С;

– у вестибюлі, гардеробі 16-19 °С;

– в санітарних вузлах 17-21 °С;

– в душових не нижче 25 °С.

Вологість повітря має підтримуватись на рівні 40–60% [22].

Для досягнення цього необхідно проводити регулярне вологе прибирання та провітрювання приміщень, а за потреби – використовувати зволожувачі або осушувачі повітря.

Освітлення має відповідати віковим та функціональним особливостям користувачів. Всі навчальні приміщення повинні мати природне і штучне освітлення.

Усі навчальні приміщення закладів освіти повинні мати природне освітлення, рівні якого мають відповідати вимогам ДБН В.2.5-28:2018 «Природне і штучне освітлення» [23], затверджених наказом Міністерства регіонального розвитку, будівництва та житлово-комунального господарства України від 03 жовтня 2018 року № 264. Природне освітлення повинно бути рівномірним і не створювати блиску.

Коефіцієнт природного освітлення (далі - КПО) в навчальних приміщеннях повинен дорівнювати 2,5 % на робочих місцях 3-го ряду робочих столів учнів (1 м від внутрішньої стіни). При двобічному освітленні мінімальне значення КПО визначається на другому ряді робочих столів учнів.

Рівномірність освітлення на робочому місці (відношення мінімального рівня освітлення до максимального) повинна складати не більше 0,3.

Достатність і рівномірність освітлення можна оцінити за світловим коефіцієнтом (СК) (відношення загальної площі вікон до площі підлоги), величина якого має становити 1 : 4 - 1 : 5.

Для захисту від прямих променів сонця, запобігання перегріву навчальних приміщень вікна повинні бути облаштовані сонцезахисними засобами (підйомно-поворотні жалюзі, козирки, ролети тощо), які легко очищаються від пилу та миються.

Забороняється розміщувати на підвіконні навчальних приміщень рослини, які перешкоджають доступу прямого сонячного світла.

Для забезпечення оптимального природного освітлення навчальних приміщень необхідно мити вікна не менше 2-х разів протягом навчального року.

Рівень шуму в навчальних приміщеннях не повинен перевищувати встановлені гранично допустимі рівні. Особливої уваги потребують класи, розташовані поблизу вулиць із інтенсивним рухом або приміщень з гучним обладнанням.

Чистота та гігієна приміщень мають забезпечуватись шляхом щоденного вологого прибирання з використанням дозволених дезінфекційних засобів, провітрювання, дотримання графіків генеральних прибирань. У санітарних вузлах повинна бути постійно наявна проточна вода, мило, паперові рушники або сушарки для рук.

Питний режим забезпечується шляхом встановлення питних фонтанчиків або автоматів із очищеною водою. За необхідності використовується фасована питна вода з сертифікатом якості.

Меблі та обладнання повинні відповідати зросту учнів та бути виготовлені з безпечних для здоров’я матеріалів. Обов’язковим є дотримання ергономічних вимог до робочих місць, що дозволяє запобігати порушенням постави та зору.

Контроль санітарно-гігієнічного стану має здійснюватися адміністрацією закладу на постійній основі. Доцільно вести журнал контролю температурного режиму та якості прибирання, здійснювати моніторинг скарг учасників освітнього процесу, співпрацювати з органами Держпродспоживслужби, медичними

установами та інспекторами з охорони праці.

Належне дотримання санітарно-гігієнічних показників не лише забезпечує виконання вимог законодавства, а й створює комфортні, безпечні та сприятливі умови для навчання й праці, що особливо важливо в умовах сучасного освітнього середовища.

4.3 Ергономічні вимоги до облаштування середовища закладу освіти

Ергономіка – це наука про пристосування умов праці до фізіологічних, психологічних та анатомічних особливостей людини з метою зниження втоми, підвищення продуктивності та збереження здоров'я. В умовах приватного закладу загальної середньої освіти ергономічні вимоги стосуються не лише персоналу (вчителів, адміністрації, технічних працівників), а й учнів, робоче місце яких має відповідати віковим та фізичним особливостям.

Основні принципи ергономіки в освітньому середовищі:

– Стільці та парти мають відповідати зростовим групам (за ДСТУ), щоб уникнути порушення постави, розвитку сколіозу та зорової втоми. Кожне робоче місце має бути промарковане (наприклад, «група 3» – для дітей зростом 130–145 см).

– Бажано використовувати меблі з можливістю регулювання висоти сидіння, нахилу спинки, кута нахилу стільниці – особливо в комп'ютерних класах.

– Робоче місце має дозволяти вільно рухатися, змінювати позу, без перешкод для проходу між партами. Рекомендований простір на одне робоче місце – не менше 1,25 м².

– Джерело світла повинно розміщуватися ліворуч (для правшів) та не створювати тіней або відблисків на поверхні стола або екрана монітора. У комп'ютерних класах екрани слід розташовувати під кутом 90° до вікна.

– Стіл вчителя повинен бути розміщений так, щоб забезпечувати візуальний

контроль за класом, мати зручне розташування для документів, засобів зв’язку, комп’ютерного обладнання.

– Екран монітора має знаходитись на відстані 50–70 см від очей учня, а його верхній край – на рівні очей або трохи нижче. Робоче місце повинне бути обладнане стільцем з опорою для спини, підлокітниками, а клавіатура – розташована на одному рівні з ліктями.

– З метою зменшення втоми учнів, у розкладі передбачаються фізкультхвилинки, перерви після занять за комп’ютером (рекомендовано – 15-хвилинна перерва після кожних 30 хвилин роботи), чергування розумової та фізичної активності.

– Робочий простір повинен мати зручну температуру, сприятливу кольорову гаму, акустичні характеристики, зони для відпочинку, що важливо для збереження ментального здоров’я.

– Усі елементи робочого місця мають бути виготовлені з екологічно безпечних матеріалів без гострих кутів і з міцними кріпленнями [24].

Упровадження ергономічних принципів у шкільне середовище сприяє не лише безпеці праці та навчання, а й створює умови для зниження рівня захворюваності, підвищення концентрації уваги, загального комфорту та якості освітнього процесу.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було проаналізовано практичну задачу інженерії програмного забезпечення та здійснено постановку задачі на розробку програмного забезпечення для автоматизації обліку навчального процесу приватної школи “СІА: свідомість, іновації, академічність”. Здійснено розробку програмного забезпечення, здійснивши аналіз вимог, проєктування архітектури та розроблення проєкту програмного забезпечення. Також було досліджено охорону праці у школах.

Були вирішені наступні задачі:

1) Проведено аналіз практичної задачі інженерії програмного забезпечення.

2) Здійснено аналіз існуючих аналогів.

3) Виконано постановку задачі на розробку ПЗ для автоматизації обліку навчального процесу приватної школи.

4) Проведено аналіз вимог.

5) Розроблено архітектуру.

7) Розроблено ескізний, технічний і робочий проєкти.

8) Розроблено розділ “Охорона праці”.

9) Розроблено документацію.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Про затвердження Порядку надання пільг окремим категоріям громадян з урахуванням середньомісячного сукупного доходу сім'ї. Постанова, Порядок від 04.06.2015 № 389. Редакція: 16.09.2022, Діє з 23.09.2022. URL:https://ips.ligazakon.net/document/KP150389?an=1 (дата звернення 19.04.2025).

2. PowerSchool SIS | PowerSchool. URL: https://www.powerschool.com/student-information-cloud/powerschool-sis/ (дата звернення 31.03.2025).

3. PowerSchool hack: missed basic security step resulted in data breach. URL: https://www.nbcnews.com/tech/security/powerschool-hack-data-breach-protect-student-school-teacher-safe-rcna189029 (data звернення 19.04.2025).

4. Про ЄДЕБО | ЄДИНА ДЕРЖАВНА ЕЛЕКТРОННА БАЗА з питань ОСВІТИ. URL:https://info.edbo.gov.ua/about/ (дата звернення 31.03.2025).

5. Інститут модернізації змісту освіти. Єдина державна електрона база з питань освіти. Презентація. URL:https://www.imzo.gov.ua/wp-content/uploads/2015/11/Upravlinska-informatsiya.pdf (дата звернення 06.04.2025).

6. SchoolMate | Програмне забезпечення для управління школою. URL: https://www.schoolmate.com.ua/uk/ (дата звернення 31.03.2025).

7. Students | SchoolMate language school management system. URL:https://www.schoolmate.eu/en/tutorial/menu-students/ (дата звернення 06.04.2025).

8. IntelliJ IDEA – the IDE for Pro Java and Kotlin Development. URL: https://www.jetbrains.com/idea (дата звернення 31.03.2025).

9. Java | Oracle. URL: https://www.java.com/en (дата звернення 31.03.2025).

10. Spring Web MVC :: Spring Framework. URL: https://docs.spring.io/spring-framework/reference/web/webmvc.html (дата звернення 31.03.2025).

11. Introduction of Object Oriented Programming | GeeksforGeeks. URL:https://www.geeksforgeeks.org/introduction-of-object-oriented-programming (дата звернення 31.03.2025).

12. Maven – Welcome to Apache Maven. URL:https://maven.apache.org (дата звернення 31.03.2025).

13. HTML For Beginners The Easy Way: Start Learning HTML & CSS Today. URL:https://html.com (дата звернення 31.03.2025).

14. Cascading Style Sheets. URL: https://www.w3.org/Style/CSS/Overview.en.html (дата звернення 31.03.2025).

15. Learn JavaScript Online - Courses for Beginners – javascript.com. URL:https://www.javascript.com (дата звернення 31.03.2025).

16. Bootstrap · The most popular HTML, CSS, and JS library in the world. URL:https://getbootstrap.com (дата звернення 31.03.2025).

17. Thymeleaf. URL: https://www.thymeleaf.org (дата звернення 31.03.2025)

18. MariaDB Enterprise Open Source Database & SkySQL MariaDB Cloud | MariaDB. URL:https://mariadb.com (дата звернення 31.03.2025).

19. Project Lombok. URL:https://projectlombok.org (дата звернення 31.03.2025).

20. Про охорону праці | від 14.10.1992 № 2694-XII. URL: https://zakon.rada.gov.ua/laws/show/2694-12#Text (дата звернення 13.04.2025).

21. Про освіту | від 05.09.2017 № 2145-VIII. URL: https://zakon.rada.gov.ua/laws/show/2145-19#Text (дата звернення 13.04.2025).

22. Про затвердження Санітарного ре... | від 25.09.2020 № 2205. URL: https://zakon.rada.gov.ua/rada/show/z1111-20#n15 (дата звернення 13.04.2025).

23. ДБН В.2.5-28-2018 "Природне і штучне освітлення" №ДБН В.2.5-28-2018. URL: https://e-construction.gov.ua/laws_detail/3074958732556240833?doc_type=2 (дата звернення 13.04.2025).

24. ДСТУ ENV 1729-2:2004, Український інститут меблів, ДСТУ (Державний Стандарт України). URL: https://online.budstandart.com/ua/catalog/doc-page?id_doc=79859 (дата звернення 13.04.2025).

ДОДАТОК А Технічне завдання

Вступ

Повна назва продукту: програмне забезпечення “Автоматизація обліку навчального процесу приватної школи “СІА: свідомість, іновації, академічність””.

Сфера застосування

Сферою застосування програмної системи є приватна школа “СІА: свідомість, іновації, академічність”.

1 Підстави для розробки

Підставою для розробки даного програмного забезпечення є завдання на кваліфікаційну роботу для здобуття ступеню вищої освіти “бакалавр”.

2 Призначення розробки

2.1 Експлуатаційне призначення

Експлуатаційне призначення програмного забезпечення “Автоматизація обліку навчального процесу приватної школи “СІА: свідомість, іновації, академічність”” полягає в автоматизації та оптимізації процесів: зберігання, обробки інформації та формування звітів, що сприяє підвищенню ефективності адміністрування обліку учнів і полегшує взаємодію між учасниками цього процесу.

2.2 Функціональне призначення

Функціональне призначення полягає у автоматизації дій, котрі роблять робітники, що займаються обліком навчального процесу, та перегляду деякої інформації учнем/батьками учня.

Адміністратор повинен мати можливість: додавати, переглядати, редагувати, видаляти будь-яку інформацію що зберігається; формувати повний звіт стосовно учня, формувати медичний звіт, фінансову звітність, табель

успішності учня.

Завуч повинен мати можливість: додавати нові предмети, оцінки, підсумкові оцінки; переглядати особову інформацію про учнів, представників учнів; переглядати інформацію про предмети, розклад уроків, оцінки учнів, підсумкові оцінки учнів; редагувати та видаляти підсумкові оцінки, розклад уроків, оцінки; формувати табель успішності учня.

Працівник бухгалтерії повинен мати можливість: переглядати особову інформацію про учнів, представників учнів; переглядати інформацію про пільгові категорії учнів, рахунки учнів, предмети учнів; виставити учню рахунок на сплату навчання, змінити стан рахунку на сплату навчання; формувати фінансову звітність учня.

Шкільна медсестра повинна мати можливість: переглядати особову інформації про учнів, представників учнів; переглядати медичні картки учнів, редагувати медичну картку учня, формувати медичний звіт стосовно учня.

Учень/Представник учня повинні мати можливість: переглядати розклад уроків, оцінки, підсумкові оцінки.

Вчитель повинен мати можливість: переглядати особову інформації про учнів, переглядати розклад уроків, предмети, оцінки, підсумкові оцінки.

3 Вимоги до програмного забезпечення

3.1 Вимоги до функціональних характеристик

3.1.1 Вимоги до складу виконуваних функцій

Функції адміністратора:

– додавати інформацію (а саме про: вчителів, предмети, розклад занять, оцінки, підсумкові оцінки, рахунки, особові дані учнів, медичні картки, представників учнів, пільгові категорії);

– переглядати інформацію (а саме про: вчителів, предмети, розклад занять, оцінки, підсумкові оцінки, рахунки, особові дані учнів, медичні картки, представників учнів, пільгові категорії, ціну за навчання);

– редагувати інформацію (а саме про: вчителів, предмети, розклад занять,

оцінки, підсумкові оцінки, особові дані учнів, медичні картки, представників учнів, пільгові категорії, ціну за навчання);

– змінювати статус рахунків;

– видаляти інформацію (а саме про: оцінки, підсумкові оцінки, розклад).

– формувати звітність:

-повний звіт про учня (особові дані учня, медичну картку учня, представників учня, пільгові категорії учня, підсумкові оцінки учня, рахунки учня),

-фінансовий звіт про учня (ПІБ учня, клас, електрона пошта, рахунки – усі поля),

-медичний звіт про учня (ПІБ учня, клас, медична картка – усі поля),

-табель успішності учня (ПІБ учня, клас, електрона пошта, предмети – назва, підсумкові оцінки – усі поля).

Функції завуча:

– додавати нові предмети/оцінки/підсумкові оцінки/розклад;

– переглядати особову інформацію про учнів (ПІБ, клас, електрона пошта, форма навчання, стан у базі даних);

– переглядати інформацію про представників учнів (ПІБ, електрона пошта, номер телефону);

– переглядати інформацію про предмети (усі поля);

– переглядати інформацію про оцінки учнів (усі поля);

– переглядати інформацію про підсумкові оцінки учнів (усі поля);

– переглядати інформацію про розклад (усі поля);

– редагувати та видаляти розклад/оцінки/підсумкові оцінки (усі поля);

– формувати табель успішності учня (ПІБ учня, клас, електрона пошта, предмети – назва, підсумкові оцінки – усі поля).

Функції працівника бухгалтерії:

– переглядати особову інформацію про учнів (ПІБ, клас, електрона пошта, форма навчання, стан у базі даних);

– переглядати інформацію про представників учнів (ПІБ, електрона пошта,

номер телефону);

– переглядати інформації про пільгові категорії учнів (усі поля);

– переглядати інформації про рахунки учнів (усі поля);

– переглядати інформацію про предмети учнів (усі поля);

– виставити учню рахунок на сплату навчання, тобто створити новий рахунок;

– змінювати стан рахунка;

– редагувати вартість предметів (вартість у таблиці предмети);

– формувати фінансову звітність учня (ПІБ учня, клас, електрона пошта, рахунки – усі поля).

Функції шкільної медсестри:

– переглядати особову інформацію про учнів (ПІБ, клас, електрона пошта, форма навчання, стан у базі даних);

– переглядати інформацію про представників учнів (ПІБ, електрона пошта, номер телефону);

– переглядати медичні картки учнів (усі поля);

– редагувати медичну картку учня (усі поля);

– формувати медичний звіт стосовно учня (медична картка – усі поля; особові дані учня – ПІБ, клас, електрона адреса).

Функції учня/представника учня:

– переглядати розклад занять учня(усі поля);

– переглядати оцінки учня (усі поля);

– переглядати підсумкові оцінки учня(усі поля).

Функції вчителя:

– переглядати інформацію про власних учнів (ПІБ, клас, електрона пошта, форма навчання, стан у базі даних);

– переглядати власні предмети (всі поля);

– переглядати власні оцінки(всі поля);

– переглядати власний розклад(всі поля);

– переглядати власні підсумкові оцінки (всі поля);

– додавати/редагувати/видаляти свої оцінки (всі поля).

3.1.2 Вимоги до організації вхідних та вихідних даних

Вхідною інформацією є дані, які вводяться вручну (в залежності від типу поля введення, дані можуть бути: числові цілі, числові десяткові, текстові, формату дати, булеві) або обираються із представлених списків (для уникнення можливих помилок, у випадку коли є певна фіксована й невелика кількість опцій у заповненні інформації, у випадних списках представлені готові варіанти вибору), або отримуються з бази даних MariaDB.

Вихідною інформацією будуть результати операцій по взаємодії з базою даних, повідомлення, таблиці та сформовані звіти.

Вхідними даними будуть наступні:

– учень (ПІБ, клас, номер телефону, електрона пошта, зареєстрована та фактична адреси перебування, чи поновлене навчання, дата народження, форма навчання, стан у базі даних);

– медична картка (стать, медична група, вага, зріст, дата оновлення, учень);

– представник учня (ПІБ, місце роботи, номер телефону, електрона пошта);

– учень-представник учня (представник учня, стосунки з учнем, учень);

– учень-пільгова категорія (пільгова категорія, термін дії для учня, учень);

– підсумкова оцінка (за який рік навчання, оцінка за перший семестр, оцінка за другий семестр, річна оцінка, навчальний рік, учень, вчитель, предмет);

– рахунок (номер рахунку, дата формування рахунку, дата до котрої треба сплатити рахунок, дата закриття рахунку, статус рахунку, вартість до сплати, вартість до сплати з врахуванням пільг, за який рік навчання, учень);

– предмет (назва предмету, тип предмету, рік навчання для котрого цей предмет, вчителя що викладають);

– вчитель (ПІБ, номер телефону, електрона пошта, адреса проживання, дата народження, посада, напрям навчання, досвід роботи, навчальний заклад);

– розклад (періодичність заняття, день тижня для заняття, час початку заняття, час закінчення заняття, номер кабінету, вчитель, предмет, навчальна

група);

– навчальна група(ідентифікатор групи);

– учень-навчальна група(учень, навчальна група);

– оцінка (дата оцінки, значення оцінки, учень, вчитель, предмет);

– розцінки (проміжок навчальних років, для котрого розцінка, вартість навчання).

3.2 Вимоги до надійності

3.2.1 Вимоги до забезпечення надійного (сталого) функціонування програми.

Надійне функціонування програми має бути забезпечене виконанням сукупності організаційно-технічних заходів, перелік яких подано нижче:

– моніторинг та логування: встановлення системи моніторингу, яка відстежує стан програми та обладнання серверів, а також системи логування, яка реєструє інформацію про події та помилки для швидкого виявлення проблем.

– резервне копіювання даних: час від часу створення резервних копій бази даних і конфігураційних файлів, а також забезпечення їх зберігання на віддалених серверах чи в безпечних об'єктах для збереження цілісності даних у випадку аварійної ситуації.

– резервне живлення та джерела живлення: забезпечення надійного та безперебійного живлення серверів та інших інфраструктурних компонентів.

3.2.2 Відмови через некоректні дії оператора

Немає

3.3 Умови експлуатації

3.3.1 Кліматичні умови експлуатації

Кліматичні умови експлуатації, при яких повинні забезпечуватись задані характеристики, повинні відповідати вимогам, що пред'являються до технічних засобів в частині умов їх експлуатації.

3.3.2 Вимоги до кваліфікації користувачів

Користувачі повинні мати базові навички роботи з ноутбуком або комп'ютером.

3.4 Вимоги до складу і параметрів технічних засобів

Для коректної роботи програмного забезпечення потрібно мати сервер наступної мінімальної конфігурації:

– Intel Core i3–10100 3,6 ГГц або аналогічний за характеристиками AMD Ryzen 3 3300X;

– оперативна пам'ять 4 ГБ;

– 20 ГБ вільного місця на диску;

– з'єднання по локальній мережі.

Також маємо клієнтську машину наступної мінімальної конфігурації:

– Intel Core i3–10100 3,6 ГГц або аналогічний за характеристиками AMD Ryzen 3 3300X;

– оперативна пам'ять 2 ГБ;

– з'єднання по локальній мережі.

3.5 Вимоги до інформаційної та програмної сумісності

Вимогли до програмного забезпечення серверу:

– система управління базами даних Mariadb;

– операційна система не нижче Windows 10;

Вимоги до програмного забезпечення клієнтської машини:

– операційна система не нижче Windows 10;

– браузер Google Chrome версії не нижче 93.0.4577.82 або браузер Mozilla Firefox версії не нижче 91.0.

3.6 Спеціальні вимоги

Програма повинна забезпечувати взаємодію з користувачем за допомогою

графічного інтерфейсу користувача, розробленого відповідно до рекомендацій компанії виробника операційної системи.

4 Вимоги до програмної документації

Склад програмної документації повинен включати в себе:

– технічне завдання;

– програму та методику випробувань;

– керівництво користувача;

– опис програми;

– код програми.

5 Техніко-економічні показники

У даній роботі техніко-економічні показники не розраховуються.

6 Етапи роботи

Етапи роботи представлені у таблиці А.1.

Таблиця А.1 – Етапи роботи

Номер	Назва етапу роботи	Термін виконання
1.	Аналіз практичної задачі інженерії ПЗ	19.02.2025
2.	Аналіз вимог до ПЗ	28.02.2025
3.	Розробка архітектури ПЗ	08.03.2025
4.	Розробка проєкту ПЗ	26.03.2025
5.	Реалізація та тестування ПЗ	30.04.2025

7 Порядок контролю та приймання

Контроль за аналізом та проєктуванням кожної окремої частини програмного забезпечення проводиться на кожному етапі з урахуванням вимог, визначених у технічному завданні. Кожна стадія розробки повинна бути представлена в зазначені строки та узгоджена із замовником.

Приймання проводиться відповідно до програми і методики випробування і

документують за допомогою протоколу проведення випробувань. У разі знаходження помилок під час приймання програмного продукту складається акт про знайдені помилки, який підписується представниками замовника і розробника і затверджується керівником організації – замовника та організації – розробника. Розробник повинен протягом не більше ніж двох тижнів виправити зазначені помилки і оповістити замовника про повторне проходження перевірки.

ДОДАТОК Б Текст програми

```
package com.project.javaproject;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.scheduling.annotation.EnableScheduling;

@EnableScheduling
@SpringBootApplication
public class JavaProjectApplication {
    public static void main(String[] args) {
        SpringApplication.run(JavaProjectApplication.class, args);
    }
}
```

```
package com.project.javaproject.beans;
import com.project.javaproject.services.StudentService;
import jakarta.annotation.PostConstruct;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.scheduling.annotation.Scheduled;
import org.springframework.stereotype.Component;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.time.LocalDate;
import java.util.Scanner;

@Component
public class Scheduler {
    @Autowired
    private StudentService studentService;

    private final String promotionDateFilePath = "src/main/resources/last-promotion.txt";

    @PostConstruct
    public void init() {
        checkAndUpdateClasses();
    }

    public void checkAndUpdateClasses() {
        LocalDate today = LocalDate.now();
        LocalDate promotionDate = LocalDate.of(today.getYear(), 8, 1);
        try {
            LocalDate lastUpdate = readLastPromotionDate();

            if (lastUpdate.isBefore(promotionDate)) {
                if (today.isEqual(promotionDate) || today.isAfter(promotionDate)) {
                    studentService.promoteStudyYear();
                    writeLastPromotionDate(today);
                }
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

```
    @Scheduled(cron = "0 0 1 * * *")
    public void dailyCheck() {
        checkAndUpdateClasses();
    }
    private LocalDate readLastPromotionDate() throws IOException {
        File file = new File(promotionDateFilePath);
        Scanner sc = new Scanner(file);
        return LocalDate.parse(sc.nextLine());
    }

    private void writeLastPromotionDate(LocalDate date) throws IOException {
        try {
            FileWriter f2 = new FileWriter(promotionDateFilePath);
            f2.write(date.toString());
            f2.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

```
package com.project.javaproject.controllers;
import com.project.javaproject.dto.StudentSubjectsSubjectsDto;
import com.project.javaproject.entities.AnnualAssessment;
import com.project.javaproject.entities.Student;
import com.project.javaproject.entities.Subject;
import com.project.javaproject.services.AnnualAssessmentService;
import com.project.javaproject.services.ContextCheckerService;
import com.project.javaproject.services.StudentService;
import com.project.javaproject.services.SubjectService;
import lombok.AllArgsConstructor;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;
import java.util.List;
import java.util.Optional;

@Controller
@AllArgsConstructor
public class AnnualAssessmentController {
    private AnnualAssessmentService annualAssessmentService;

    @Autowired
    private ContextCheckerService contextCheckerService;

    @Autowired
    private StudentService studentService;

    @Autowired
    private SubjectService subjectService;

    @PostMapping("/grade-delete")
    public String gradeDelete(@RequestParam int id) {
        String url = "home";
        if(contextCheckerService.hasRole("ALL") || contextCheckerService.hasRole("MARK")) {
            Optional<AnnualAssessment> annualAssessmentOptional =
annualAssessmentService.findAnnualAssessmentById(id);
            if (annualAssessmentOptional.isPresent()) {
```

```
                int studentId = annualAssessmentOptional.get().getStudent().getId();
                annualAssessmentService.deleteAnnualAssessment(id);
                url="redirect:/watch-student-grades?id="+studentId;
            }
        }
        return url;
    }

    @GetMapping("/watch-student-grades")
    public String watchStudentGrades(@RequestParam(required = false) Integer id,
@RequestParam(required = false) Integer subjectId, Model model) {
        String url="home";
        if(contextCheckerService.hasRole("ALL") || contextCheckerService.hasRole("MARK") ||
contextCheckerService.hasRole("STUDY") || contextCheckerService.hasRole("TEACH")) {
            if(null == id) {
                if(contextCheckerService.hasRole("STUDY")) {
                    id=contextCheckerService.getCurrentUserId();
                }
                else return url;
            }
            Optional<Student> studentOptional = studentService.findStudentById(id);
            if(studentOptional.isPresent()) {
                boolean showGrades=false;
                if(contextCheckerService.hasRole("STUDY")) {
                    if(contextCheckerService.getCurrentUserId() ==
studentOptional.get().getId()) {
                        showGrades=true;
                    }
                }
                else {
                    showGrades=true;
                }
                if(showGrades) {
                    List<AnnualAssessment> annualAssessments;
                    if(contextCheckerService.hasRole("TEACH") && subjectId!=null) {
                        annualAssessments =
annualAssessmentService.findByStudent_IdAndSubject_Id(id, subjectId);
                    }
                    else {
                        annualAssessments =
annualAssessmentService.findAnnualAssessmentByStudentId(id);
                    }
                    model.addAttribute("grades", annualAssessments);
                    model.addAttribute("student", studentOptional.get());
                    url = "student-grades";
                }
            }
        }
        return url;
    }

    @GetMapping("/add-grade")
    public String addingGrade(@RequestParam int studentId, Model model) {
        String url="home";
        if(contextCheckerService.hasRole("ALL") || contextCheckerService.hasRole("MARK")) {
            url = "redirect:/students";
            StudentSubjectsSubjectsDto sssDto =
subjectService.getStudentCurrentSubjectsAndAllOtherSubjects(studentId);
            if(sssDto.isChecked()) {
                model.addAttribute("student", sssDto.getStudent());
                model.addAttribute("subjects", sssDto.getSubjects());
                model.addAttribute("subjectsAll", sssDto.getSubjects_twin());
```

```
                url = "grade-add-subject";
            }
        }
        return url;
    }

    @GetMapping("/grade-add-information")
    public String gradeAddInformation(@RequestParam int studentId, @RequestParam int
subjectId, Model model) {
        String url="home";
        if(contextCheckerService.hasRole("ALL") || contextCheckerService.hasRole("MARK")) {
            url = "redirect:/students";
            Optional<Student> studentOptional = studentService.findStudentById(studentId);
            Optional<Subject> subjectOptional = subjectService.findSubjectById(subjectId);
            if (studentOptional.isPresent() && subjectOptional.isPresent()) {
                model.addAttribute("student", studentOptional.get());
                model.addAttribute("subject", subjectOptional.get());
                model.addAttribute("yearsGrades",
annualAssessmentService.countGradesPerStudyYears(studentId, subjectId));
                model.addAttribute("studentYear",
studentService.getStudentStudyYear(studentId));
                model.addAttribute("studyYears",
contextCheckerService.getCurrentStudyYears());
                url = "grade-add-information";
            }
        }
        return url;
    }

    @PostMapping("/confirm-add-grade")
    public String confirmAddGrade(@RequestParam int studentId, @RequestParam int subjectId,
@RequestParam String calendarYears) {
        String url="home";
        if(contextCheckerService.hasRole("ALL") || contextCheckerService.hasRole("MARK")) {
            url = "redirect:/students";
            Optional<Student> studentOptional = studentService.findStudentById(studentId);
            Optional<Subject> subjectOptional = subjectService.findSubjectById(subjectId);
            if (studentOptional.isPresent() || subjectOptional.isPresent()) {
                annualAssessmentService.addAnnualAssessment(studentId, subjectId,
calendarYears);
                url = "redirect:/watch-student-grades?id=" + studentId;
            }
        }
        return url;
    }

    @PostMapping("/update-grades-all")
    public String addGradesForAll() {
        String url="home";
        if(contextCheckerService.hasRole("ALL") || contextCheckerService.hasRole("MARK")) {
            url="redirect:/students";
            annualAssessmentService.addAnnualAssessmentToAllStudents();
        }
        return url;
    }

    @GetMapping("/edit-assessment")
    public String gradeAddInformation(@RequestParam int id, Model model) {
        String url="home";
        if(contextCheckerService.hasRole("ALL") || contextCheckerService.hasRole("MARK")) {
            url = "redirect:/students";
            Optional<AnnualAssessment> annualAssessmentOptional =
```

```
annualAssessmentService.findAnnualAssessmentById(id);
            if (annualAssessmentOptional.isPresent()) {
                model.addAttribute("grade", annualAssessmentOptional.get());
                url = "edit-grade";
            }
        }
        return url;
    }

    @PostMapping("/update-grade")
    public String udpateGrade(@RequestParam int id, @RequestParam int
yearGrade ,@RequestParam int fSemesterGrade, @RequestParam int sSemesterGrade,
@RequestParam int studyYear, @RequestParam String calendarYears) {
        String url = "home";
        if (contextCheckerService.hasRole("ALL") || contextCheckerService.hasRole("MARK"))
{
            url = "redirect:/students";
            Optional<AnnualAssessment> annualAssessmentOptional =
annualAssessmentService.findAnnualAssessmentById(id);
            if(annualAssessmentOptional.isPresent()) {
                annualAssessmentService.updateAnnualAssessment(id, yearGrade,
fSemesterGrade,  sSemesterGrade, studyYear, calendarYears);
                url="redirect:/watch-student-grades?
id="+annualAssessmentOptional.get().getStudent().getId();
            }
        }
        return url;
    }
}


package com.project.javaproject.services;
import com.project.javaproject.dao.*;
import com.project.javaproject.entities.*;
import lombok.AllArgsConstructor;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import java.util.*;

@Service
@AllArgsConstructor
public class AnnualAssessmentService {
    private AnnualAssessmentRepository annualAssessmentRepository;

    @Autowired
    private StudentRepository studentRepository;

    @Autowired
    private MarkRepository markRepository;

    @Autowired
    private TeacherRepository teacherRepository;

    @Autowired
    private SubjectRepository subjectRepository;

    @Autowired
    private ContextCheckerService contextCheckerService;

    public boolean deleteAnnualAssessment(int id) {
        annualAssessmentRepository.deleteById(id);
        return true;
```

```
    }

    public List<AnnualAssessment> findAnnualAssessmentByStudentId(int id) {
        return annualAssessmentRepository.findByStudent_Id(id);
    }

    public Optional<AnnualAssessment> findAnnualAssessmentById(int id) {
        return annualAssessmentRepository.findById(id);
    }

    public List<AnnualAssessment> studentStudyYearAnnualAssessments(int studentId, int
studyYear) { return annualAssessmentRepository.findByStudent_IdAndStudyYear(studentId,
studyYear); }
    public List<AnnualAssessment> studentOldStudyYearAnnualAssessments(int studentId, int
studyYear) { return
annualAssessmentRepository.findByStudent_IdAndStudyYearNotOrderByStudyYearDesc(studentId,
studyYear); }

    public List<AnnualAssessment> findByStudent_IdOrderByStudyYearDesc(int id) { return
annualAssessmentRepository.findByStudent_IdOrderByStudyYearDesc(id); }

    public List<AnnualAssessment> findByStudent_IdAndSubject_Id(int studentId, int
subjectId) { return annualAssessmentRepository.findByStudent_IdAndSubject_Id(studentId,
subjectId); }

    public List<List<String>> countGradesPerStudyYears(int studentId, int subjectId) {
        List<Mark> marks = markRepository.findByStudent_IdAndSubject_Id(studentId,
subjectId);
        Map<String, Integer> yearsGrades=new HashMap<>();
        List<List<String>> yearsGradesF = new ArrayList<>();
        for(Mark mark:marks) {
            String studyYears;
            if(mark.getDate().getMonthValue()<8) {
                studyYears=(mark.getDate().getYear()-1)+"-"+(mark.getDate().getYear());
            }
            else {
                studyYears=mark.getDate().getYear()+"-"+(mark.getDate().getYear()+1);
            }
            boolean oldStudyYears=false;
            for(var entry:yearsGrades.entrySet()) {
                if(entry.getKey().equals(studyYears)) {
                    entry.setValue(entry.getValue()+1);
                    oldStudyYears=true;
                }
            }
            if(!oldStudyYears) {
                yearsGrades.put(studyYears, 1);
            }
        }
        if(!yearsGrades.isEmpty()) {
            for (var entry:yearsGrades.entrySet()) {
                List<String> tmp=new ArrayList<>();
                tmp.add(entry.getKey());
                tmp.add(String.valueOf(entry.getValue()));
                yearsGradesF.add(tmp);
            }
        }
        return yearsGradesF;
    }

    public void addAnnualAssessmentToAllStudents() {
        String calendarYears = contextCheckerService.getCurrentStudyYears();
```

```
        List<Student> students = studentRepository.findByIdNotNullAndState("НАВЧАЄТСЯ");
        for (Student student : students) {
            Set<Subject> subjects = student.getStudiedSubjects();
            for (Subject subject : new HashSet<>(subjects)) {
                addAnnualAssessment(student.getId(), subject.getId(), calendarYears);
            }
        }
    }

    public boolean addAnnualAssessment(int studentId, int subjectId, String calendarYears)
{
        Optional<Student> studentOptional = studentRepository.findById(studentId);
        Optional<Subject> subjectOptional = subjectRepository.findById(subjectId);
        List<Mark> marks = markRepository.findByStudent_IdAndSubject_Id(studentId,
subjectId);
        String []yearsStr = calendarYears.split("-");
        int beginYear = Integer.parseInt(yearsStr[0]);
        int endYear = Integer.parseInt(yearsStr[1]);
        if (endYear - beginYear == 1) {
            if (studentOptional.isPresent() && subjectOptional.isPresent() && !
marks.isEmpty()) {
                Integer teacherId = MarkService.findMostFrequentValue(marks);
                Optional<Teacher> teacherOptional = teacherRepository.findById(teacherId);
                if(teacherOptional.isPresent()) {
                    List<Mark> fsMark = new ArrayList<>();
                    List<Mark> ssMark = new ArrayList<>();
                    for (Mark mark : marks) {
                        if (mark.getDate().getYear() == beginYear &&
mark.getDate().getMonthValue() > 7) {
                            fsMark.add(mark);
                        } else if (mark.getDate().getYear() == endYear &&
mark.getDate().getMonthValue() < 7) {
                            ssMark.add(mark);
                        }
                    }
                    AnnualAssessment annualAssessmenttmp = new AnnualAssessment();
                    Optional<AnnualAssessment> oldAssessment =
annualAssessmentRepository.findByStudent_IdAndSubject_IdAndStudyYearAndCalendarYears(stude
ntId, subjectId, subjectOptional.get().getStudyYear(), calendarYears);
                    if (oldAssessment.isPresent()) {
                        annualAssessmenttmp = oldAssessment.get();
                    }
                    if (!fsMark.isEmpty() && !ssMark.isEmpty()) {

annualAssessmenttmp.setStudyYear(subjectOptional.get().getStudyYear());
                        double tmpV = 0;
                        double tmpN = 0;
                        for (Mark mark : fsMark) {
                            tmpV += mark.getMark();
                            tmpN++;
                        }
                        if (tmpN != 0) {
                            tmpV /= tmpN;
                            annualAssessmenttmp.setFSemesterGrade((int) Math.round(tmpV));
                        }
                        tmpV = 0;
                        tmpN = 0;
                        for (Mark mark : ssMark) {
                            tmpV += mark.getMark();
                            tmpN++;
                        }
                        if (tmpN != 0) {
```

```
                            tmpV /= tmpN;
                            annualAssessmenttmp.setSSemesterGrade((int) Math.round(tmpV));
                        }
                        if (annualAssessmenttmp.getFSemesterGrade() > 0 &&
annualAssessmenttmp.getFSemesterGrade() < 13 && annualAssessmenttmp.getSSemesterGrade() >
0 && annualAssessmenttmp.getSSemesterGrade() < 12) {
                            annualAssessmenttmp.setYearGrade((int)
Math.round((annualAssessmenttmp.getFSemesterGrade() +
annualAssessmenttmp.getSSemesterGrade()) / 2.0));
                        }
                        annualAssessmenttmp.setCalendarYears(calendarYears);
                        AnnualAssessment annualAssessment =
annualAssessmentRepository.save(annualAssessmenttmp);
                        annualAssessment.setStudent(studentOptional.get());
                        annualAssessment.setSubject(subjectOptional.get());
                        annualAssessment.setTeacher(teacherOptional.get());
                        Set<AnnualAssessment> studentGrades =
studentOptional.get().getAnnualAssessments();
                        Set<AnnualAssessment> teacherGrades =
teacherOptional.get().getAnnualAssessments();
                        Set<AnnualAssessment> subjectGrades =
subjectOptional.get().getAnnualAssessments();
                        subjectGrades.add(annualAssessment);
                        subjectOptional.get().setAnnualAssessments(subjectGrades);
                        teacherGrades.add(annualAssessment);
                        teacherOptional.get().setAnnualAssessments(teacherGrades);
                        studentGrades.add(annualAssessment);
                        studentOptional.get().setAnnualAssessments(studentGrades);
                        annualAssessmentRepository.save(annualAssessment);
                        studentRepository.save(studentOptional.get());
                        subjectRepository.save(subjectOptional.get());
                        teacherRepository.save(teacherOptional.get());
                        return true;
                    }
                }
            }
        }
        return false;
    }

    public boolean updateAnnualAssessment(int id, int yearGrade , int fSemesterGrade, int
sSemesterGrade, int studyYear, String calendarYears) {
        Optional<AnnualAssessment> annualAssessmentOptional =
annualAssessmentRepository.findById(id);
        if(annualAssessmentOptional.isPresent()){
            annualAssessmentOptional.get().setYearGrade(yearGrade);
            annualAssessmentOptional.get().setFSemesterGrade(fSemesterGrade);
            annualAssessmentOptional.get().setSSemesterGrade(sSemesterGrade);
            annualAssessmentOptional.get().setStudyYear(studyYear);
            annualAssessmentOptional.get().setCalendarYears(calendarYears);
            annualAssessmentRepository.save(annualAssessmentOptional.get());
            return true;
        }
        return false;
    }

}
package com.project.javaproject.entities;
import jakarta.persistence.*;
import lombok.Getter;
import lombok.Setter;
```

```
@Getter
@Setter
@Entity
@Table(name = "annual_assessments")
public class AnnualAssessment {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id", nullable = false)
    private Integer id;

    @Column(name = "year_grade")
    private Integer yearGrade;

    @Column(name = "f_semester_grade")
    private Integer fSemesterGrade;

    @Column(name = "s_semester_grade")
    private Integer sSemesterGrade;

    @Column(name = "study_year")
    private Integer studyYear;

    @Column(name = "calendar_years", length = 9)
    private String calendarYears;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "student_id")
    private Student student;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "subject_id")
    private Subject subject;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "teacher_id")
    private Teacher teacher;

}
```

```
package com.project.javaproject.dao;

import com.project.javaproject.entities.AnnualAssessment;
import org.springframework.data.jpa.repository.JpaRepository;

import java.util.List;
import java.util.Optional;

public interface AnnualAssessmentRepository extends JpaRepository<AnnualAssessment,
Integer> {
    List<AnnualAssessment> findByIdNotNull();

    List<AnnualAssessment> findByStudent_Id(Integer id);

    List<AnnualAssessment> findByStudent_IdOrderByStudyYearDesc(Integer id);

    Optional<AnnualAssessment>
findByStudent_IdAndSubject_IdAndStudyYearAndCalendarYears(Integer id, Integer id1, Integer
studyYear, String calendarYears);

    List<AnnualAssessment> findByStudent_IdAndSubject_Id(Integer id, Integer id1);
```

```
    List<AnnualAssessment> findByStudent_IdAndStudyYear(Integer id, Integer studyYear);

    List<AnnualAssessment> findByStudent_IdAndStudyYearNotOrderByStudyYearDesc(Integer id,
Integer studyYear);

}
```

```
package com.project.javaproject.entities;
import jakarta.persistence.*;
import lombok.Getter;
import lombok.Setter;
import java.time.LocalDate;
import java.util.LinkedHashSet;
import java.util.Set;

@Getter
@Setter
@Entity
@Table(name = "students")
public class Student {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id", nullable = false)
    private Integer id;

    @Column(name = "name", length = 127)
    private String name;

    @Column(name = "surname", length = 127)
    private String surname;

    @Column(name = "patronymic", length = 127)
    private String patronymic;

    @Column(name = "class", length = 5)
    private String classField;

    @Column(name = "phone", length = 25)
    private String phone;

    @Column(name = "address")
    private String address;

    @Column(name = "current_location_address")
    private String currentLocationAddress;

    @Column(name = "renewed_study")
    private Boolean renewedStudy;

    @Column(name = "birthdate")
    private LocalDate birthdate;

    @Column(name = "study_form", length = 127)
    private String studyForm;

    @Column(name = "state", length = 32)
    private String state;

    @Column(name = "email", length = 321)
    private String email;
```

```
    @OneToMany(mappedBy = "student")
    private Set<AnnualAssessment> annualAssessments = new LinkedHashSet<>();

    @OneToMany(mappedBy = "student")
    private Set<Mark> marks = new LinkedHashSet<>();

    @OneToMany(mappedBy = "student")
    private Set<Payment> payments = new LinkedHashSet<>();

    @ManyToMany
    @JoinTable(
            name = "students_subjects",
            joinColumns = @JoinColumn(name = "student_id"),
            inverseJoinColumns = @JoinColumn(name = "subject_id"))
    Set<Subject> studiedSubjects;


    @OneToMany(mappedBy = "student")
    Set<StudentsRepresentative> representatives;

    @OneToMany(mappedBy = "student")
    Set<StudentsPreferencialCategory> preferencialCategories;

    @OneToOne(cascade = CascadeType.ALL)
    @JoinColumn(name = "healthcard_id", referencedColumnName = "id")
    private Healthcard healthcard;

    @OneToMany(mappedBy = "student")
    private Set<StudentsStudyGroup> studentsStudyGroups = new LinkedHashSet<>();

}
```

```
package com.project.javaproject.dao;
import com.project.javaproject.entities.Student;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import java.util.List;

public interface StudentRepository extends JpaRepository<Student, Integer> {
    @Query("""
            select s from Student s inner join s.studiedSubjects studiedSubjects inner join
s.studiedSubjects.teachers teachers
            where studiedSubjects.id = ?1 and teachers.id = ?2""")
    List<Student> findByStudiedSubjects_IdAndStudiedSubjects_Teachers_Id(Integer subjectId,
Integer teacherId);

    List<Student> findByIdNotNullOrderByIdAsc();

    List<Student> findByIdNotNullOrderByNameAsc();

    List<Student> findByStudiedSubjects_Id(Integer id);

    List<Student> findByIdNotNullAndState(String state);

}
```

```
package com.project.javaproject.controllers;
import com.project.javaproject.entities.Student;
import com.project.javaproject.entities.Teacher;
import com.project.javaproject.services.ContextCheckerService;
```

```
import com.project.javaproject.services.StudentService;
import com.project.javaproject.services.TeacherService;
import lombok.AllArgsConstructor;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.multipart.MultipartFile;
import java.time.LocalDate;
import java.util.List;
import java.util.Optional;

@Controller
@AllArgsConstructor
public class StudentController {

   @Autowired
   private ContextCheckerService contextCheckerService;

   private StudentService studentService;

   @Autowired
   private TeacherService teacherService;

   @GetMapping("/students")
   public String showAllStudents(Model model){
       String url = "home";
       List<Student> students = studentService.findAllStudents();
       model.addAttribute("students", students);
       if(contextCheckerService.hasRole("ALL")) url = "students";
       else if(contextCheckerService.hasRole("MONEY")) url = "money";
       else if(contextCheckerService.hasRole("MARK")) url = "assessments";
       else if(contextCheckerService.hasRole("HEALTH")) url = "healthcards";
       else if(contextCheckerService.hasRole("STUDY")) {
           int id = contextCheckerService.getCurrentUserId();
           Optional<Student> optionalStudent = studentService.findStudentById(id);
           if(optionalStudent.isPresent()){
               model.addAttribute("student", optionalStudent.get());
               url = "redirect:/see_student_details?id="+id;
           }
       }
       else if(contextCheckerService.hasRole("TEACH")) {
           int id = contextCheckerService.getCurrentUserId();
           Optional<Teacher> optionalTeacher = teacherService.findTeacherById(id);
           if(optionalTeacher.isPresent()){
               url = "redirect:/subjects";
           }
       }
       return url;
   }


   @GetMapping("/see_student_details")
   public String seeStudentDetails(@RequestParam(required = false) Integer id, Model
model){
       String url = "home";
       if(contextCheckerService.hasRole("ALL") || contextCheckerService.hasRole("STUDY"))
{
           if(null == id) {
               if(contextCheckerService.hasRole("STUDY")) {
```

```
                    id=contextCheckerService.getCurrentUserId();
                }
                else return url;
            }
            Optional<Student> student = studentService.findStudentById(id);
            if (student.isPresent()) {
                boolean loadDetails = false;
                if (contextCheckerService.hasRole("STUDY")) {
                    if (student.get().getId() == contextCheckerService.getCurrentUserId())
{
                        loadDetails = true;
                    }
                } else {
                    loadDetails = true;
                }
                if (loadDetails) {
                    model.addAttribute("student", student.get());
                    url = "student-details";
                }
            }
        }
        return url;
    }

    @GetMapping("/edit-student")
    public String editStudent(@RequestParam int id, Model model){
        String url="home";
        if(contextCheckerService.hasRole("ALL")) {
            Optional<Student> optionalStudent = studentService.findStudentById(id);
            if(optionalStudent.isPresent()){
                model.addAttribute("student", optionalStudent.get());
                url = "edit-student";
            }
            else url="redirect:/students";
        }
        return url;
    }

    @PostMapping("/update-student")
    public String updateStudent(@RequestParam int id, @RequestParam String name,
@RequestParam String surname, @RequestParam String patronymic, @RequestParam String
classField, @RequestParam String phone, @RequestParam String address, @RequestParam String
currentLocationAddress, @RequestParam String renewedStudy, @RequestParam LocalDate
birthdate, @RequestParam String studyForm, @RequestParam String email, @RequestParam
String state){
        String url="home";
        if(contextCheckerService.hasRole("ALL")) {
            if(studentService.updateStudent(id, name, surname, patronymic, classField,
phone, address, currentLocationAddress,
                    renewedStudy, birthdate, studyForm, email, state)) {
                url="redirect:/see_student_details?id="+id;
            }
        }
        return url;
    }

    @PostMapping("add-student")
    public String addStudent(@RequestParam String name, @RequestParam String surname,
@RequestParam String patronymic, @RequestParam String classField, @RequestParam String
currentLocation, @RequestParam String phone, @RequestParam String address, @RequestParam
int representativeId, @RequestParam String relationship, @RequestParam String
renewedStudy, @RequestParam String birthdate, @RequestParam String email, @RequestParam
```

```
String studyForm){
        String url="home";
        if(contextCheckerService.hasRole("ALL")) {
            if(studentService.addStudent(name, surname, patronymic, classField,
currentLocation, phone, address, representativeId, relationship, renewedStudy, birthdate,
email, studyForm)) {
                url = "redirect:/students";
            }
        }
        return url;
    }

    @PostMapping("/upload-avatar")
    public String handleAvatarUpload(@RequestParam MultipartFile image, @RequestParam int
studentId) {
        String url = "home";
        if(contextCheckerService.hasRole("ALL")) {
            url="redirect:/students";
            if (!image.isEmpty()) {
                try {
                    if(studentService.updateStudentPhoto(image, studentId)) {
                        url = "redirect:/see_student_details?id="+studentId;
                    }
                } catch (Exception e) {
                    System.out.println(e);
                }
            }
        }
        return url;
    }
}
```

```
package com.project.javaproject.services;
import com.project.javaproject.dao.HealthcardRepository;
import com.project.javaproject.dao.RepresentativeRepository;
import com.project.javaproject.dao.StudentRepository;
import com.project.javaproject.dao.StudentsRepresentativeRepository;
import com.project.javaproject.entities.Healthcard;
import com.project.javaproject.entities.Representative;
import com.project.javaproject.entities.Student;
import com.project.javaproject.entities.StudentsRepresentative;
import lombok.AllArgsConstructor;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.userdetails.User;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.provisioning.JdbcUserDetailsManager;
import org.springframework.stereotype.Service;
import org.springframework.web.multipart.MultipartFile;
import javax.imageio.ImageIO;
import java.awt.image.BufferedImage;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.LocalDate;
import java.util.List;
import java.util.Optional;

@Service
@AllArgsConstructor
public class StudentService {
```

```
    private StudentRepository studentRepository;

    @Autowired
    private HealthcardRepository healthcardRepository;

    @Autowired
    private RepresentativeRepository representativeRepository;

    @Autowired
    private StudentsRepresentativeRepository studentsRepresentativeRepository;

    @Autowired
    private SubjectService subjectService;

    @Autowired
    private JdbcUserDetailsManager jdbcUserDetailsManager;

    public void promoteStudyYear() {
        List<Student> studentList = studentRepository.findByIdNotNullAndState("НАВЧАЄТСЯ");
        for(int i=0; i<studentList.size(); i++) {
            String[] classField = studentList.get(i).getClassField().split("-");
            int year=Integer.parseInt(classField[0]);
            if(year<13 && year>0) {
                studentList.get(i).setClassField(String.valueOf(year+1)+"-"+classField[1]);
                if(year==12) {
                    studentList.get(i).setState("ВИПУСТИВСЯ");
                }
            }
        }
        List<Student> changedStudents = studentRepository.saveAll(studentList);
        for(Student student:changedStudents) {
            subjectService.studentUpdateMainSubjects(student.getId());
        }
    }

    public Optional<Student> findStudentById(int id) {
        return studentRepository.findById(id);
    }

    public List<Student> findAllStudents() {
        return studentRepository.findByIdNotNullOrderByNameAsc();
    }

    public boolean updateStudent(int id, String name, String surname, String patronymic,
String classField, String phone, String address, String currentLocationAddress,  String
renewedStudy, LocalDate birthdate, String studyForm, String email, String state) {
        Optional<Student> optionalStudent = studentRepository.findById(id);
        boolean isValid = validateStudentInfo(classField, studyForm);
        if(optionalStudent.isPresent() || isValid) {
            Student student = optionalStudent.get();
            student.setId(id);
            student.setName(name);
            student.setSurname(surname);
            student.setPatronymic(patronymic);
            student.setClassField(classField);
            student.setPhone(phone);
            student.setAddress(address);
            student.setCurrentLocationAddress(currentLocationAddress);
            if (renewedStudy.equalsIgnoreCase("true") ||
renewedStudy.equalsIgnoreCase("так"))
                student.setRenewedStudy(Boolean.TRUE);
            else if (renewedStudy.equalsIgnoreCase("false") ||
```

```
renewedStudy.equalsIgnoreCase("ні"))
                student.setRenewedStudy(Boolean.FALSE);
            student.setBirthdate(birthdate);
            student.setStudyForm(studyForm);
            student.setState(state);
            student.setEmail(email);
            studentRepository.save(student);
            if(!optionalStudent.get().getClassField().equals(classField)) {
                subjectService.studentUpdateMainSubjects(id);
            }
            return true;
        }
        else return false;
    }

    public boolean addStudent(String name, String surname, String patronymic, String
classField, String currentLocation, String phone, String address, int representativeId,
String relationship, String renewedStudy, String birthdate, String email, String
studyForm) {
        Optional<Representative> representativeOptional =
representativeRepository.findById(representativeId);
        boolean isValid = validateStudentInfo(classField, studyForm);
        if(representativeOptional.isPresent() || isValid) {
            Student student = new Student();
            student.setName(name);
            student.setSurname(surname);
            student.setPatronymic(patronymic);
            student.setClassField(classField);
            student.setPhone(phone);
            student.setAddress(address);
            student.setCurrentLocationAddress(currentLocation);
            if(renewedStudy.equals("true") || renewedStudy.equals("так"))
student.setRenewedStudy(Boolean.TRUE);
            else if(renewedStudy.equals("false") || renewedStudy.equals("ні"))
student.setRenewedStudy(Boolean.FALSE);
            student.setEmail(email);
            student.setStudyForm(studyForm);
            student.setState("НАВЧАЄТСЯ");
            student.setBirthdate(LocalDate.parse(birthdate));
            Healthcard healthcard = new Healthcard();
            healthcard.setUpdateDate(LocalDate.now());
            healthcardRepository.save(healthcard);
            healthcard.setStudent(student);
            student.setHealthcard(healthcard);
            Student savedStudent = studentRepository.save(student);
            Student savedStudent1 = studentRepository.save(savedStudent);
            StudentsRepresentative studentsRepresentative = new StudentsRepresentative();
            studentsRepresentative.setStudent(savedStudent1);
            studentsRepresentative.setRepresentative(representativeOptional.get());
            studentsRepresentative.setRelationship(relationship);
            studentsRepresentativeRepository.save(studentsRepresentative);
            healthcardRepository.save(healthcard);
            UserDetails studentAcc = User.withDefaultPasswordEncoder()
                    .username("учень"+savedStudent1.getId())
                    .password(savedStudent1.getPhone())
                    .roles("STUDY")
                    .build();
            jdbcUserDetailsManager.createUser(studentAcc);
            return true;
        }
        return false;
    }
```

```
    public int getStudentStudyYear(int studentId) {
        Optional<Student> studentOptional = studentRepository.findById(studentId);
        if(studentOptional.isPresent()) {
            return Integer.parseInt(studentOptional.get().getClassField().split("-")[0]);
        }
        return -1;
    }

    public boolean updateStudentPhoto(MultipartFile image, int studentId) throws
IOException {
        if(studentRepository.findById(studentId).isPresent()) {
            if (image.isEmpty()) {
                throw new IllegalArgumentException("Файл порожній!");
            }
            String contentType = image.getContentType();
            if (contentType == null || !contentType.startsWith("image/")) {
                throw new IllegalArgumentException("Файл не є зображенням!");
            }
            BufferedImage bufferedImage = ImageIO.read(image.getInputStream());
            if (bufferedImage == null) {
                throw new IllegalArgumentException("Не вдалося зчитати зображення!");
            }
            Path uploadPath = Paths.get("uploads/photo/");
            if (!Files.exists(uploadPath)) {
                Files.createDirectories(uploadPath);
            }
            String fileName = studentId + ".png";
            Path filePath = uploadPath.resolve(fileName);
            ImageIO.write(bufferedImage, "png", filePath.toFile());
            return true;
        }
        return false;
    }

    public boolean validateStudentInfo(String classField, String studyForm) {
        boolean isScheduleValid=true;
        if(!(studyForm.equals("очна") || studyForm.equals("дистанціна") ||
studyForm.equals("змішана") || studyForm.equals("екстернатна")
                || studyForm.equals("індивідуальна") || studyForm.equals("сімейна")))
isScheduleValid=false;
        if(!classField.matches("^[0-9]{1,2}-[А-Я]{1,2}$")) isScheduleValid=false;
        return isScheduleValid;
    }
}


package com.project.javaproject.controllers;
import com.project.javaproject.entities.AnnualAssessment;
import com.project.javaproject.entities.Healthcard;
import com.project.javaproject.entities.Payment;
import com.project.javaproject.entities.Student;
import com.project.javaproject.services.*;
import jakarta.servlet.http.HttpSession;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpHeaders;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;
```

```
import java.net.URLEncoder;
import java.nio.charset.StandardCharsets;
import java.util.*;

@RestController
@RequestMapping("/report")
public class ReportController {
   private final ReportService reportService;

   @Autowired
   private ContextCheckerService contextCheckerService;

   @Autowired
   private StudentService studentService;

   @Autowired
   private HealthcardService healthcardService;

   @Autowired
   private AnnualAssessmentService annualAssessmentService;

   @Autowired
   private PaymentService paymentService;

   public ReportController(ReportService reportService) { this.reportService =
reportService; }

   @GetMapping("/pdf")
   public ResponseEntity<byte[]> generateReport(@RequestParam int id) {
       if(contextCheckerService.hasRole("ALL")) {
           try {
               List<Student> items = new ArrayList<>();
               Optional<Student> studentOptional = studentService.findStudentById(id);
               if (studentOptional.isPresent()) {
                   items.add(studentOptional.get());
                   Map<String, Object> data = new HashMap<>();
                   data.put("items", items);

                   byte[] pdfContent = reportService.generatePdf("report", data);

                   return ResponseEntity.ok()
                           .header(HttpHeaders.CONTENT_DISPOSITION, "inline;
filename=report.pdf")
                           .contentType(MediaType.APPLICATION_PDF)
                           .body(pdfContent);
               }
           } catch (Exception e) {
               return ResponseEntity.internalServerError().build();
           }
       }
       return ResponseEntity.internalServerError().build();
   }

   @GetMapping("/medical/pdf")
   public ResponseEntity<byte[]> generateMedReport(@RequestParam int id) {
       if(contextCheckerService.hasRole("ALL") || contextCheckerService.hasRole("HEALTH"))
{
           try {
               Optional<Healthcard> healthcardOptional =
healthcardService.findHealthcardByStudentId(id);
               Optional<Student> studentOptional = studentService.findStudentById(id);
               if (studentOptional.isPresent() && healthcardOptional.isPresent()) {
```

```
                    Student student = studentOptional.get();
                    Healthcard healthcard = healthcardOptional.get();
                    Map<String, Object> data = new HashMap<>();
                    data.put("student", student);
                    data.put("healthcard", healthcard);

                    byte[] pdfContent = reportService.generatePdf("report-medical", data);

                    String filename = "Медичний_Звіт_" + student.getSurname() + "_" +
student.getClassField() + ".pdf";
                    String encodedFilename = URLEncoder.encode(filename,
StandardCharsets.UTF_8).replace("+", "%20");

                    return ResponseEntity.ok()
                            .header(HttpHeaders.CONTENT_DISPOSITION, "inline;
filename*=UTF-8''" + encodedFilename)
                            .contentType(MediaType.APPLICATION_PDF)
                            .body(pdfContent);
                }
            } catch (Exception e) {
                return ResponseEntity.internalServerError().build();
            }
        }
        return ResponseEntity.internalServerError().build();
    }

    @GetMapping("/study/pdf")
    public ResponseEntity<byte[]> generateStudyReport(@RequestParam int id) {
        if(contextCheckerService.hasRole("ALL") || contextCheckerService.hasRole("MARK")) {
            try {
                Optional<Student> studentOptional = studentService.findStudentById(id);
                if (studentOptional.isPresent()) {
                    int studyYear = studentService.getStudentStudyYear(id);
                    List<AnnualAssessment> gradesCurrent =
annualAssessmentService.studentStudyYearAnnualAssessments(id, studyYear);
                    List<AnnualAssessment> gradesOld =
annualAssessmentService.studentOldStudyYearAnnualAssessments(id, studyYear);
                    Student student = studentOptional.get();
                    Map<String, Object> data = new HashMap<>();
                    data.put("student", student);
                    data.put("gradesCurrent", gradesCurrent);
                    data.put("gradesOld", gradesOld);

                    byte[] pdfContent = reportService.generatePdf("report-study", data);

                    String filename = "Табель_Успішності_" + student.getSurname() + "_" +
student.getClassField() + ".pdf";
                    String encodedFilename = URLEncoder.encode(filename,
StandardCharsets.UTF_8).replace("+", "%20");

                    return ResponseEntity.ok()
                            .header(HttpHeaders.CONTENT_DISPOSITION, "inline;
filename*=UTF-8''" + encodedFilename)
                            .contentType(MediaType.APPLICATION_PDF)
                            .body(pdfContent);
                }
            } catch (Exception e) {
                return ResponseEntity.internalServerError().build();
            }
        }
        return ResponseEntity.internalServerError().build();
    }
```

```
    @GetMapping("/money/pdf")
    public ResponseEntity<byte[]> generatMoneyReport(@RequestParam int id) {
        if(contextCheckerService.hasRole("ALL") || contextCheckerService.hasRole("MONEY"))
{
            try {
                List<Payment> payments =
paymentService.getStudentPayments(id).getPayments();
                Optional<Student> studentOptional = studentService.findStudentById(id);
                if (studentOptional.isPresent()) {
                    Student student = studentOptional.get();
                    Map<String, Object> data = new HashMap<>();
                    data.put("student", student);
                    data.put("payments", payments);

                    byte[] pdfContent = reportService.generatePdf("report-money", data);

                    String filename = "Фінансова_Звітність_" + student.getSurname() + "_" +
student.getClassField() + ".pdf";
                    String encodedFilename = URLEncoder.encode(filename,
StandardCharsets.UTF_8).replace("+", "%20");

                    return ResponseEntity.ok()
                            .header(HttpHeaders.CONTENT_DISPOSITION, "inline;
filename*=UTF-8''" + encodedFilename)
                            .contentType(MediaType.APPLICATION_PDF)
                            .body(pdfContent);
                }
            } catch (Exception e) {
                return ResponseEntity.internalServerError().build();
            }
        }
        return ResponseEntity.internalServerError().build();
    }

    @GetMapping("/full/pdf")
    public ResponseEntity<byte[]> generateFullReport(@RequestParam int id) {
        if(contextCheckerService.hasRole("ALL")) {
            try {
                Optional<Student> studentOptional = studentService.findStudentById(id);
                if (studentOptional.isPresent()) {
                    int studyYear = studentService.getStudentStudyYear(id);
                    List<AnnualAssessment> gradesCurrent =
annualAssessmentService.studentStudyYearAnnualAssessments(id, studyYear);
                    List<AnnualAssessment> gradesOld =
annualAssessmentService.studentOldStudyYearAnnualAssessments(id, studyYear);
                    Student student = studentOptional.get();
                    Map<String, Object> data = new HashMap<>();
                    data.put("student", student);
                    data.put("gradesCurrent", gradesCurrent);
                    data.put("gradesOld", gradesOld);

                    byte[] pdfContent = reportService.generatePdf("report-full", data);

                    String filename = "Повний_Звіт_" + student.getSurname() + "_" +
student.getClassField() + ".pdf";
                    String encodedFilename = URLEncoder.encode(filename,
StandardCharsets.UTF_8).replace("+", "%20");

                    return ResponseEntity.ok()
                            .header(HttpHeaders.CONTENT_DISPOSITION, "inline;
filename*=UTF-8''" + encodedFilename)
                            .contentType(MediaType.APPLICATION_PDF)
```

```
                        .body(pdfContent);
                }
            } catch (Exception e) {
                return ResponseEntity.internalServerError().build();
            }
        }
        return ResponseEntity.internalServerError().build();
    }
    @GetMapping("/cancel-transaction")
    public String cancelTransaction(HttpSession session, @RequestParam String
transactionId) {
        String startUrl = (String) session.getAttribute("transactionStartUrl_" +
transactionId);
        return "redirect:" + (startUrl != null ? startUrl : "/defaultPage");
    }
}
```

```
package com.project.javaproject.services;
import com.openhtmltopdf.pdfboxout.PdfRendererBuilder;
import org.springframework.stereotype.Service;
import org.thymeleaf.context.Context;
import org.thymeleaf.spring6.SpringTemplateEngine;
import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.OutputStream;
import java.util.Map;

@Service
public class ReportService {

    private final SpringTemplateEngine templateEngine;

    public ReportService(SpringTemplateEngine templateEngine) {
        this.templateEngine = templateEngine;
    }

    public byte[] generatePdf(String templateName, Map<String, Object> data) throws
Exception {
        Context context = new Context();
        context.setVariables(data);
        String html = templateEngine.process(templateName, context);

        try (ByteArrayOutputStream outputStream = new ByteArrayOutputStream()) {
            OutputStream os = outputStream;
            PdfRendererBuilder builder = new PdfRendererBuilder();
            builder.useFastMode();
            builder.withHtmlContent(html, null);
            builder.toStream(os);
            builder.useFont(new
File("src\\main\\resources\\static\\fonts\\DejaVuSans.ttf"), "DejaVuSans");
            builder.run();
            return outputStream.toByteArray();
        }
    }
}
```

ДОДАТОК В Опис програмного забезпечення

1. Загальні відомості

1.1 Призначення і найменування розробки

Розроблене програмне забезпечення “Автоматизація обліку навчального процесу приватної школи “CIA: свідомість, іновації, академічність”” є програмою для адміністратора, завуча, вчителів, медсестри, бухгалтера, учнів приватної школи, які займаються обліком та переглядом інформації.

Програмне рішення передбачає створення вебзастосунку що дозволить зменшити навантаження, покращити якість комунікації, забезпечити зручний доступ до освітньої інформації.

1.2 Програмне забезпечення, яке необхідне для функціонування програми

Для функціонування програми потрібно наступне програмне забезпечення:

– Java Development Kit (JDK) v. 17;

– XAMPP (з СУБД MariaDB v. 10.4.32);

– браузер Google Chrome версії не нижче 93.0.4577.82 або браузер Mozilla Firefox версії не нижче 91.0 (для доступу клієнтів до застосунку).

Додаткові утиліти:

– Git + Git Extensions (контроль версій коду);

– IntelliJ IDEA Ultimate (IDE для написання додатку, Ultimate версія надає поліпшені можливості, й дозволяє легко маніпулювати речами пов'язаними з БД).

1.3 Мови програмування, на яких написана програма

Програма написана на таких мовах програмування:

– бекенд: Java;

– фронтенд: HTML+CSS+JavaSript.

2 Функціональне призначення

Функціональне призначення полягає в реалізації наступних функцій:

Функції адміністратора:

– додавати інформацію (а саме про: вчителів, предмети, розклад занять, оцінки, підсумкові оцінки, рахунки, особові дані учнів, медичні картки, представників учнів, пільгові категорії);

– переглядати інформацію (а саме про: вчителів, предмети, розклад занять, оцінки, підсумкові оцінки, рахунки, особові дані учнів, медичні картки, представників учнів, пільгові категорії, ціну за навчання);

– редагувати інформацію (а саме про: вчителів, предмети, розклад занять, оцінки, підсумкові оцінки, особові дані учнів, медичні картки, представників учнів, пільгові категорії, ціну за навчання);

– змінювати статус рахунків;

– видаляти інформацію (а саме про: оцінки, підсумкові оцінки, розклад).

– формувати звітність:

-повний звіт про учня (особові дані учня, медичну картку учня, представників учня, пільгові категорії учня, підсумкові оцінки учня, рахунки учня),

-фінансовий звіт про учня (ПІБ учня, клас, електрона пошта, рахунки – усі поля),

-медичний звіт про учня (ПІБ учня, клас, медична картка – усі поля),

-табель успішності учня (ПІБ учня, клас, електрона пошта, предмети – назва, підсумкові оцінки – усі поля).

Функції завуча:

– додавати нові предмети/оцінки/підсумкові оцінки/розклад;

– переглядати особову інформацію про учнів (ПІБ, клас, електрона пошта, форма навчання, стан у базі даних);

– переглядати інформацію про представників учнів (ПІБ, електрона пошта, номер телефону);

– переглядати інформацію про предмети (усі поля);

– переглядати інформацію про оцінки учнів (усі поля);

– переглядати інформацію про підсумкові оцінки учнів (усі поля);

– переглядати інформацію про розклад (усі поля);

– редагувати та видаляти розклад/оцінки/підсумкові оцінки (усі поля);

– формувати табель успішності учня (ПІБ учня, клас, електрона пошта, предмети – назва, підсумкові оцінки – усі поля).

Функції працівника бухгалтерії:

– переглядати особову інформацію про учнів (ПІБ, клас, електрона пошта, форма навчання, стан у базі даних);

– переглядати інформацію про представників учнів (ПІБ, електрона пошта, номер телефону);

– переглядати інформації про пільгові категорії учнів (усі поля);

– переглядати інформації про рахунки учнів (усі поля);

– переглядати інформацію про предмети учнів (усі поля);

– виставити учню рахунок на сплату навчання, тобто створити новий рахунок;

– змінювати стан рахунка;

– редагувати вартість предметів (вартість у таблиці предмети);

– формувати фінансову звітність учня (ПІБ учня, клас, електрона пошта, рахунки – усі поля).

Функції шкільної медсестри:

– переглядати особову інформацію про учнів (ПІБ, клас, електрона пошта, форма навчання, стан у базі даних);

– переглядати інформацію про представників учнів (ПІБ, електрона пошта, номер телефону);

– переглядати медичні картки учнів (усі поля);

– редагувати медичну картку учня (усі поля);

– формувати медичний звіт стосовно учня (медична картка – усі поля; особові дані учня – ПІБ, клас, електрона адреса).

Функції учня/представника учня:

– переглядати розклад занять учня(усі поля);

– переглядати оцінки учня (усі поля);

– переглядати підсумкові оцінки учня(усі поля).

Функції вчителя:

– переглядати інформацію про власних учнів (ПІБ, клас, електрона пошта, форма навчання, стан у базі даних);

– переглядати власні предмети (всі поля);

– переглядати власні оцінки(всі поля);

– переглядати власний розклад(всі поля);

– переглядати власні підсумкові оцінки (всі поля);

– додавати/редагувати/видаляти свої оцінки (всі поля).

3 Опис логічної структури

Розроблене програмне забезпечення для автоматизації обліку навчального процесу приватної школи “СІА: свідомість, іновації, академічність” базується на Java з використанням фреймворку Spring, завдяки котрому забезпечується розбивання програми на MVC архітектуру з наявними пакетами: entities (представлення таблиць БД у вигляді сутностей), dao (data access objects, інтерфейси для взаємодії з БД), controllers (для отримання й оброблення запитів від фронтенду), services (для обробки бізнес логіки додатку).

Для зберігання й обробки даних використовується база даних MariaDB, що забезпечує високу надійність й швидкий доступ до облікової інформації.

4 Необхідні технічні засоби

Для коректної роботи програмного забезпечення потрібно мати сервер наступної мінімальної конфігурації:

– Intel Core i3–10100 3,6 ГГц або аналогічний за характеристиками AMD Ryzen 3 3300X;

– оперативна пам'ять 4 ГБ;

– 20 ГБ вільного місця на диску;

– з'єднання по локальній мережі.

Також маємо клієнтську машину наступної мінімальної конфігурації:

– Intel Core i3–10100 3,6 ГГц або аналогічний за характеристиками AMD Ryzen 3 3300X;

– оперативна пам'ять 2 ГБ;

– з'єднання по локальній мережі.

5 Виклик і завантаження

Виклик вебзастосунку здійснюється по переході по URL. URL або прив'язаний до ярлика у файловій системі комп'ютера, або у закладках у браузері, або користувач вводить вручну у строку вводу, або відкриває посилання, надане йому кимось у середовищі інтернет-комунікації. Після цього користувача зустрічає вітальна сторінка, з якої він переходить на сторінку авторизації. Якщо користувач був вже авторизований та сесія все ще активна, то замість сторінки авторизації перейде до головної сторінки програми.

6 Вхідні та вихідні дані

Дані зберігаються у базі даних MariaDB. Вхідні дані вводяться вручну з клавіатури, обираються з випадаючих списків, обираються як опція за допомогою radiobutton, обираються з вспливаючого вікна з календарем. Виведення вихідних даних відбувається у режимі реального часу на екран монітору.

ДОДАТОК Г Керівництво користувача

1 Призначення програми

Програма призначена для поліпшення обліку навчального процесу у приватній школі “CIA: свідомість, іновації, академічність” (далі “CIA”). Виклик вебзастосунку здійснюється по переході по URL. URL або прив'язаний до ярлика у файловій системі комп'ютера, або у закладках у браузері, або користувач вводить вручну у строку вводу, або відкриває посилання надане йому кимось у середовищі інтернет-комунікації. Після цього користувача зустрічає вітальна сторінка, з якої він переходить на сторінку авторизації. Якщо користувач був вже авторизований та сесія все ще активна, то замість сторінки авторизації перейде до головної сторінки програми. Зручний інтерфейс забезпечує навігацію між розділами сайту. Дозволяє працівникам школи проводити облік навчального процесу: виставляти оцінки, обраховувати підсумкові оцінки, виставляти рахунки на сплату, назначати учням пільгові категорії, назначати вчителям та учням розклад, змінювати розцінки за навчання, формувати звітність. Зберігання даних у єдиній БД сприяє цілісності даних, а також забезпечує доступ у режимі реального часу.

2 Умови виконання програми

2.1 Апаратні засоби для виконання

Вимоги до мінімальних апаратних засобів сервера:

– Intel Core i3–10100 3,6 ГГц або аналогічний за характеристиками AMD Ryzen 3 3300X;

– оперативна пам'ять 4 ГБ;

– 20 ГБ вільного місця на диску;

– з'єднання по локальній мережі.

Вимоги до мінімальних апаратних засобів клієнта:

– Intel Core i3–10100 3,6 ГГц або аналогічний за характеристиками AMD

Ryzen 3 3300X;

– оперативна пам'ять 2 ГБ;

– з'єднання по локальній мережі.

2.2 Програмні засоби для виконання

Вимогли до мінімального програмного забезпечення серверу:

– Java Development Kit (JDK);

– система управління базами даних Mariadb;

– операційна система не нижче Windows 10.

Вимоги до мінімального програмного забезпечення клієнтської машини:

– операційна система не нижче Windows 10;

– браузер Google Chrome версії не нижче 93.0.4577.82 або браузер Mozilla Firefox версії не нижче 91.0.

2.3 Додаткові умови

Додаткові умови для виконання програми:

– Постійний доступ до Інтернету (у випадку, якщо немає локального підключення користувача до локальної мережі школи);

– Права доступу до файлової системи (для збереження звітів та завантаження фотографії учня).

3 Виконання програми

При переході по URL на сторінку програмного забезпечення “Автоматизація обліку навчального процесу приватної школи “CIA” у браузері відображається сторінка привітання (рис. Г.1), з котрої користувач приступає до взаємодії з сайтом.

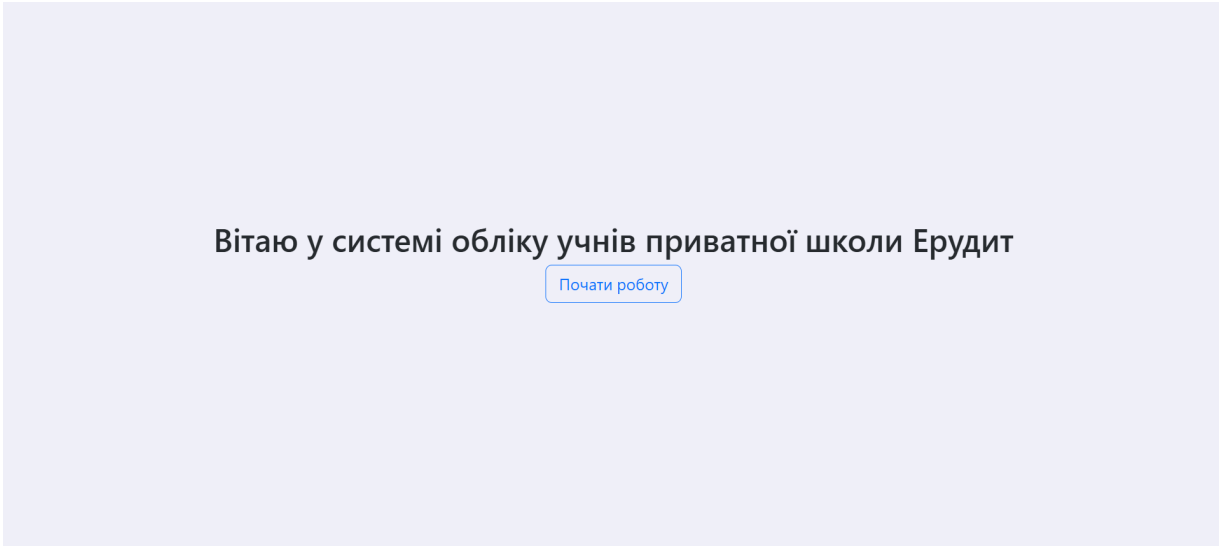


Рисунок Г.1 – Вітальна сторінка ПЗ для автоматизації обліку навчального процесу приватної школи “СІА”

При натисканні кнопки “Почати роботу”, програма перевіряє чи є активна сесія з користувачем, якщо немає, то відображається сторінка авторизації (рис. Г.2), інакше, відображається головна сторінка програми, що залежить від прав авторизованого користувача.

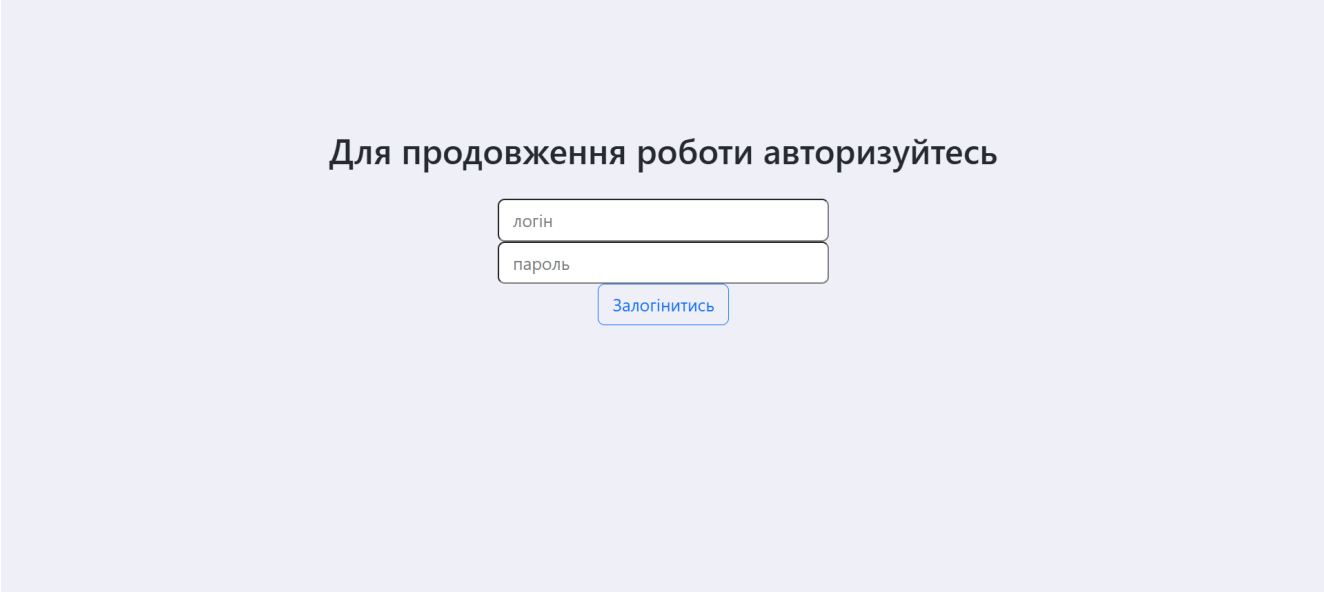


Рисунок Г.2 – Сторінка авторизації при роботі з ПЗ для автоматизації обліку навчального процесу приватної школи “СІА”

Якщо користувач авторизований як медсестра, завуч, адміністратор або

бухгалтер, то на головній сторінці програми він побачить перелік учнів; якщо ж користувач авторизований як вчитель, то на головній сторінці програми він побачить перелік предметів (рис. Г.3), а якщо авторизований як учень, то на головній сторінці побачить сторінку детальніше про себе (рис. Г.4).

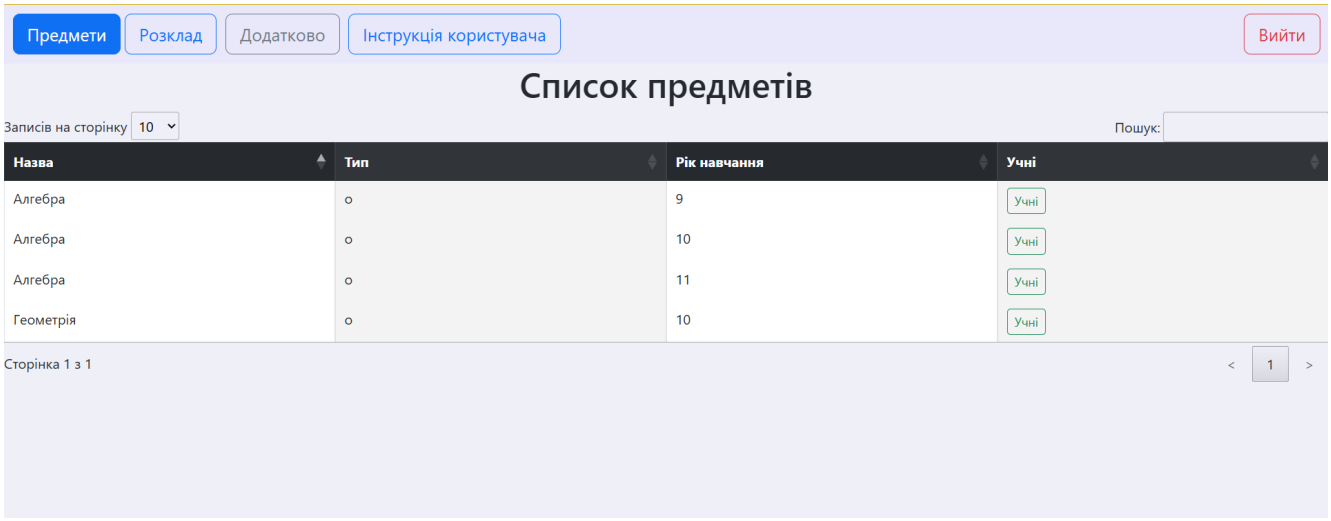


Рисунок Г.3 – Головна сторінка вчителя при роботі з ПЗ для автоматизації обліку навчального процесу приватної школи “СІА”

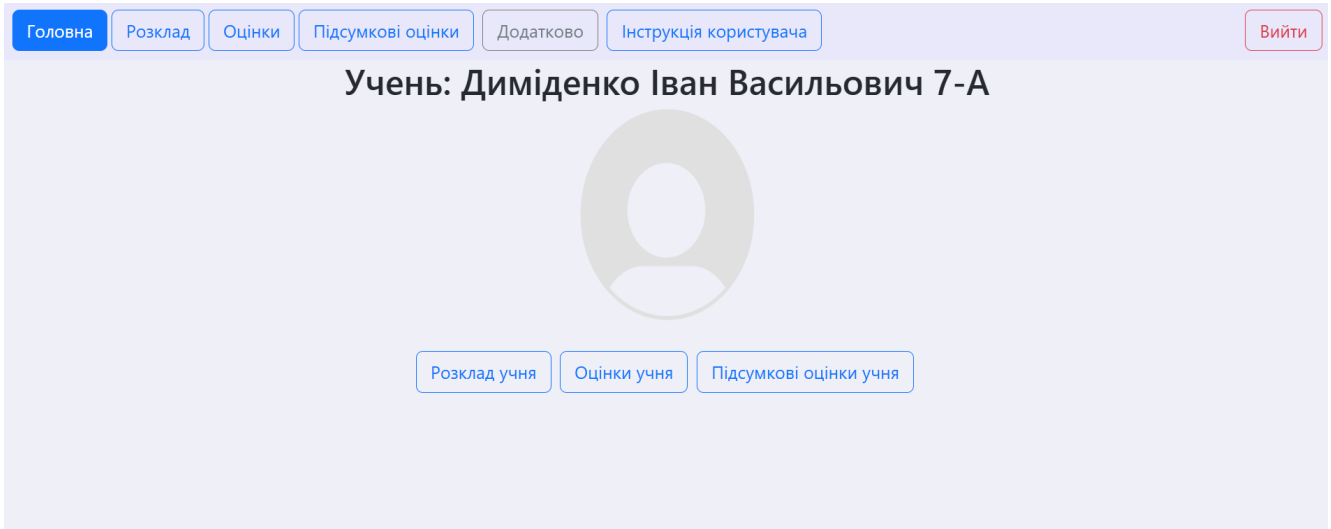


Рисунок Г.4 – Головна сторінка учня при роботі з ПЗ для автоматизації обліку навчального процесу приватної школи “СІА”

Адміністратор має більш широкі функціональні можливості при роботі з учнями (рис. Г.5), через що їх частина винесена у “Детальніше” (рис. Г.6).

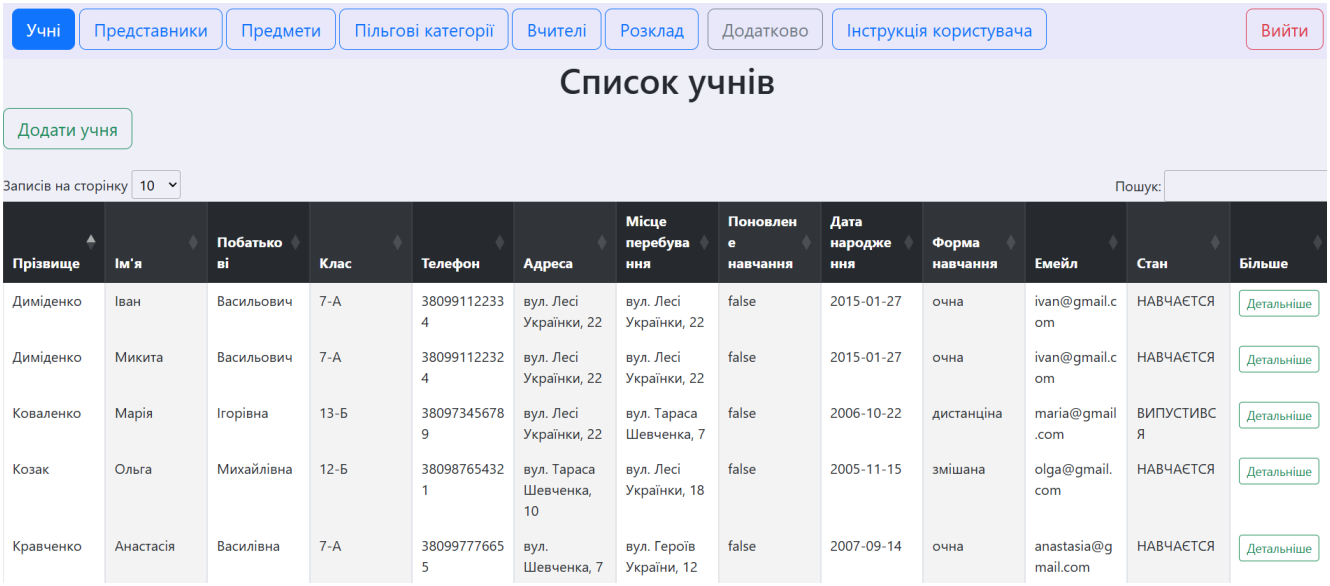


Рисунок Г.5 – Головна сторінка адміністратора при роботі з ПЗ для автоматизації обліку навчального процесу приватної школи “СІА”

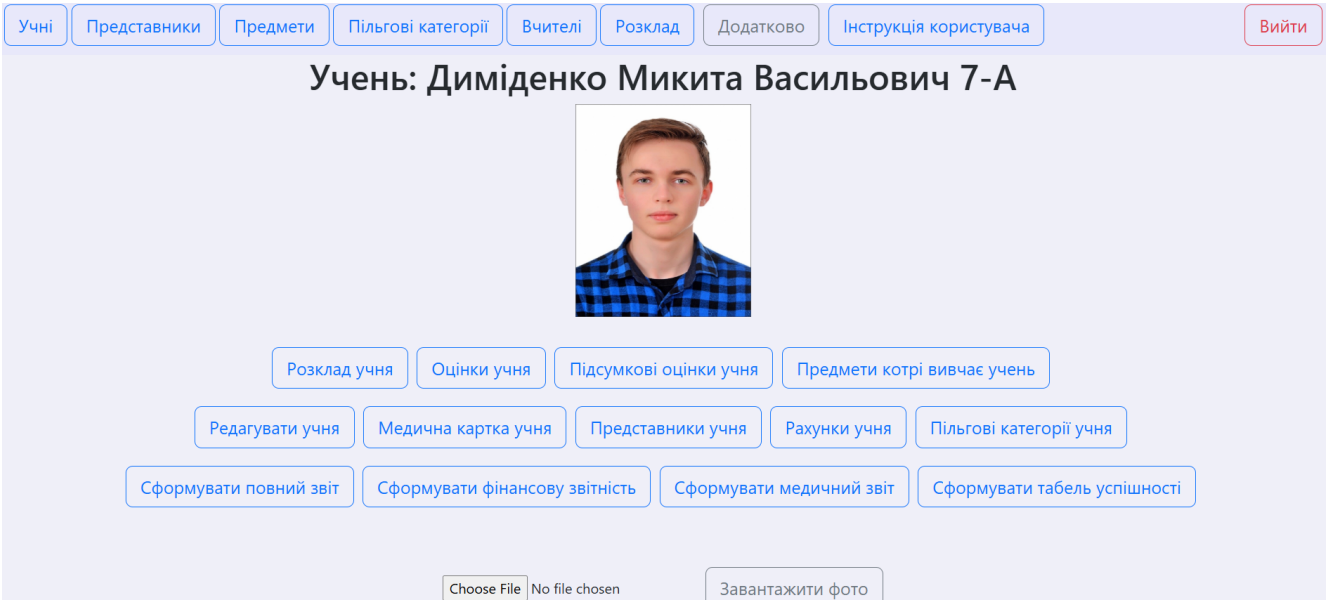


Рисунок Г.6 – Сторінка “Детальніше” при роботі адміністратора з учнями у ПЗ для автоматизації обліку навчального процесу приватної школи “СІА”

Завуч відповідає за організацію навчального процесу, тому на головній сторінці (рис. Г.7) він має кнопки, пов'язані з цим.

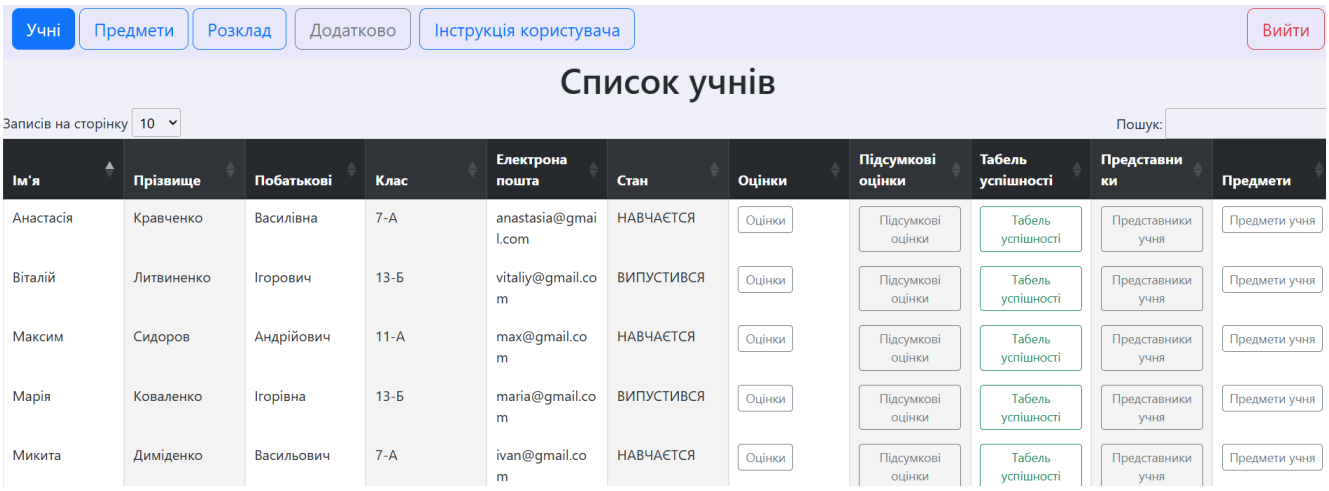


Рисунок Г.7 – Головна сторінка завуча при роботі з ПЗ для автоматизації обліку навчального процесу приватної школи “СІА”

Бухгалтер відповідає за рахунки та пільгові категорії учнів та має головну сторінку (рис. Г.8) відповідно з цим.

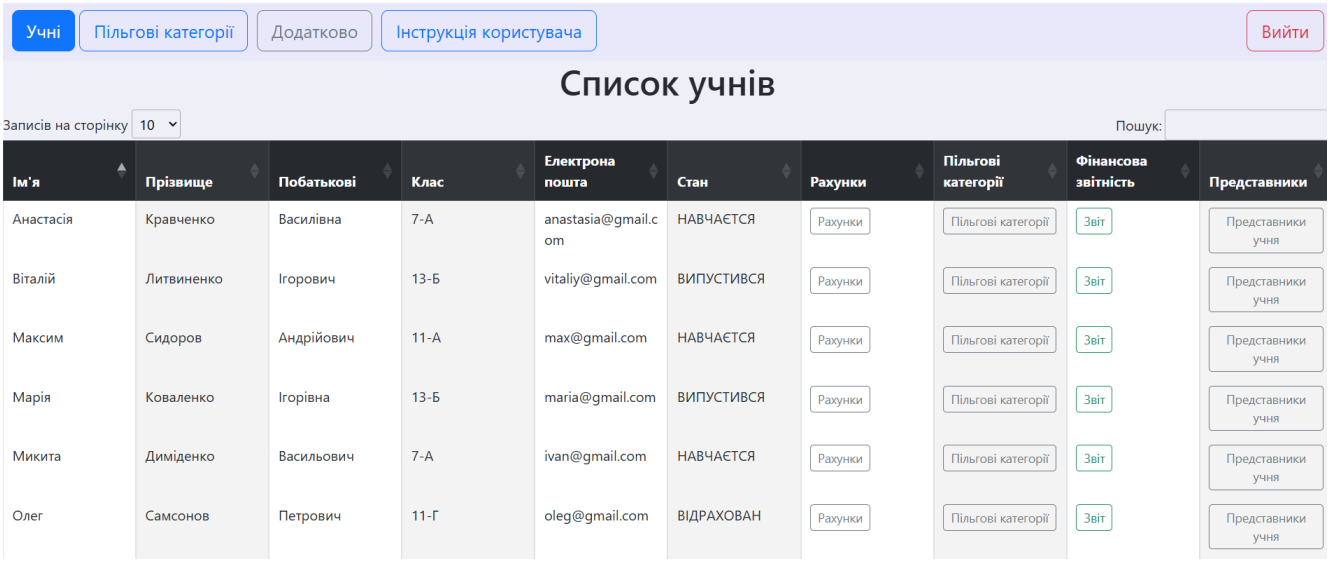


Рисунок Г.8 – Головна сторінка бухгалтера при роботі з ПЗ для автоматизації обліку навчального процесу приватної школи “СІА”

Медсестра відповідає виключно за ведення медичних карток учнів, тому вона має головну сторінку (рис. Г.9) з відповідними елементами.

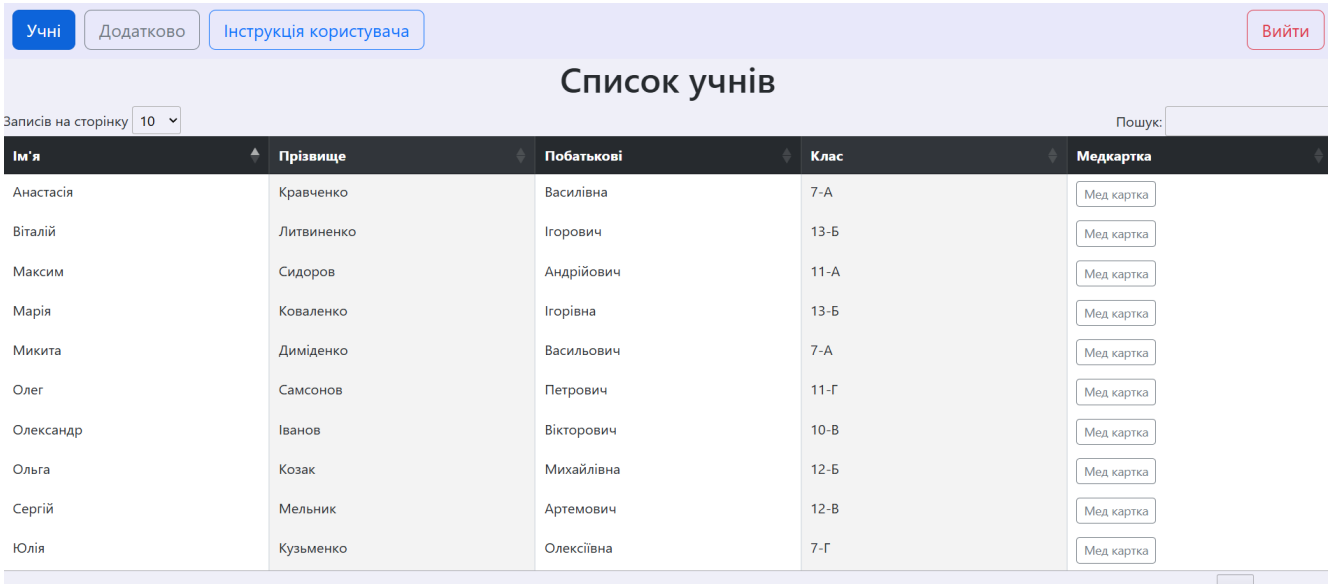


Рисунок Г.9 – Головна сторінка медсестри при роботі з ПЗ для автоматизації обліку навчального процесу приватної школи “СІА”

Щоб вийти з сесії, достатньо натиснути “Вийти” у головному меню. Якщо натиснути “Інструкція користувача”, то у новій вкладці відкриється pdf файл з інструкцією користувача. Для зміни паролю треба натиснути “Додатково” у верхньому меню. Після цього програма перенаправить на сторінку, на котрій є кнопка “Змінити пароль”. При натисканні на цю кнопку програма перенаправить на форму зміни паролю (рис. Г.10). На цій сторінці можна натиснути на емоджі ока щоб побачити пароль що вводиш.

Всі інші функції, виконуються по різному в залежності від користувача. Головне меню (меню що знаходиться у верхній частині екрану), відображається на кожній сторінці. Вміст цього меню залежить від ролі авторизованого користувача. Кнопки, які має головне меню в залежності від ролі, можна побачити у наведених вище рисунках Г.3 – Г.5, Г.7 – Г.9. В незалежності від ролі авторизованого користувача, перша кнопка у головному меню, веде на головну сторінку.

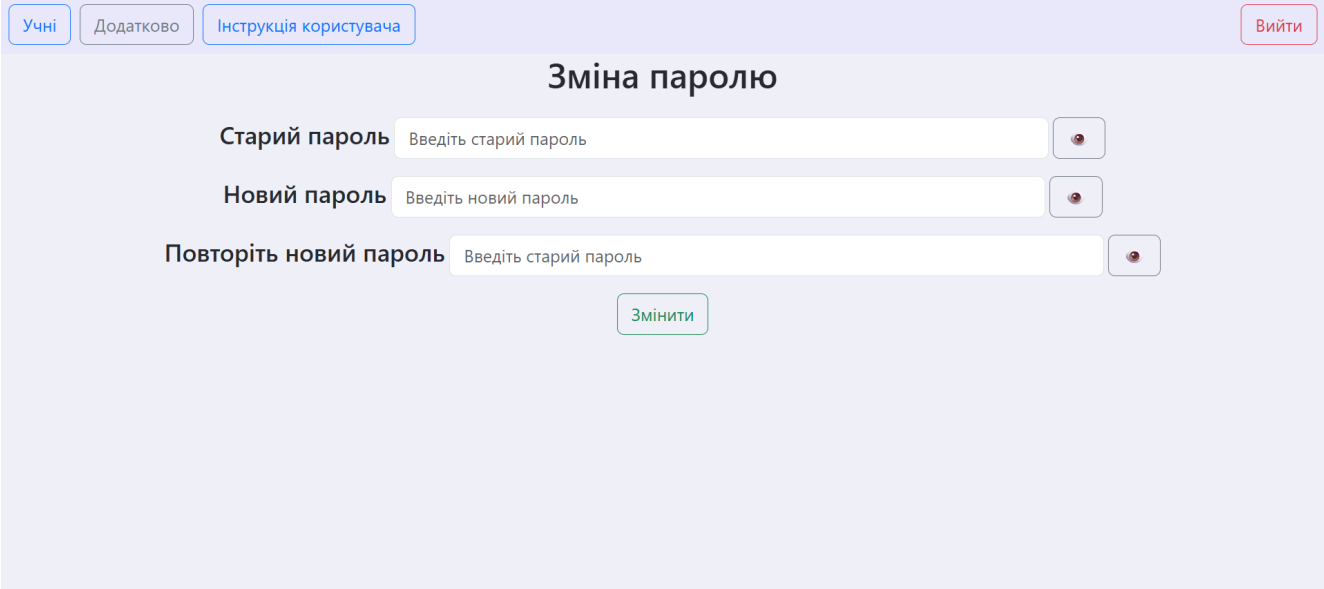


Рисунок Г.10 – Сторінка зміни паролю у ПЗ для автоматизації обліку навчального процесу приватної школи “CIA”

Нижче наведені дії користувачів для виконання різних функцій. Для усіх користувачів вказаний алгоритм працює з початкової точки – головної сторінки.

Адміністратор:

Додавання учня: натиснути “Додати учня” → обрати представника зі списку та натиснути “Обрати” → заповнити усі поля вводу → натиснути “Зберегти”.

Редагування учня: обрати учня зі списку та натиснути “Детальніше” → натиснути “Редагувати учня” → внести зміни у потрібні поля → натиснути “Оновити”.

Додати представника: натиснути “Представники” → заповнити усі поля форми → натиснути “Додати представника”.

Редагувати представника: натиснути “Представники” → обрати представника зі списку та натиснути “Редагувати” → внести зміни у потрібні поля → натиснути “Оновити”.

Додати зв'язок учня з представником: обрати учня зі списку й натиснути “Детальніше” → натиснути “Представники учня” → натиснути “Додати представника” → обрати представника зі списку й заповнити поле вводу “Стосунки” й натиснути “Обрати”.

Видалити зв'язок учня з представником: обрати учня зі списку та натиснути “Детальніше” → натиснути “Представники учня” → обрати представника зі списку й натиснути “Видалити зв'язок”.

Додати предмет: натиснути “Предмети” → заповнити усі поля вводу → натиснути “Додати предмет”.

Редагування предмету: натиснути “Предмети” → обрати предмет зі списку та натиснути “Редагувати” → внести зміни у потрібні поля вводу → натиснути “Оновити”.

Додавання вчителя: натиснути “Вчителі” → заповнити усі поля вводу → натиснути “Додати вчителя”.

Редагувати вчителя: натиснути “Вчителі” → обрати вчителя зі списку та натиснути “Редагувати” → змінити необхідні поля → натиснути “Оновити”.

Призначити вчителя на викладання предмету: натиснути “Предмети” → обрати предмет зі списку та натиснути “Вчителі” → натиснути “Додати вчителя” → обрати вчителя зі списку та натиснути “Обрати”.

Зняти вчителя з викладання предмету: натиснути “Предмети” → обрати предмет зі списку та натиснути “Вчителі” → обрати вчителя зі списку та натиснути “Видалити зв'язок”.

Оновити всім учням предмети відповідно до класу навчання: натиснути “Додатково” → натиснути “Оновити всім учням предмети відповідно до класу навчання”.

Призначити учню додатковий предмет: обрати учня зі списку та натиснути “Детальніше” → натиснути “Предмети, котрі вивчає учень” → натиснути “Поставити додатковий предмет” → обрати предмет зі списку й натиснути “Обрати”.

Зняти учня з предмету: обрати учня зі списку та натиснути “Детальніше” → натиснути “Предмети, котрі вивчає учень” → обрати предмет зі списку й натиснути “Видалити зв'язок”.

Редагувати медичну картку учня: обрати учня зі списку та натиснути “Детальніше” → натиснути “Медична картка учня” → натиснути “Редагувати” →

змінити необхідні поля → натиснути “Оновити”.

Додати пільгову категорію: натиснути “Пільгові категорії” → заповнити усі поля вводу → натиснути “Додати пільгову категорію”.

Редагувати пільгову категорію: натиснути “Пільгові категорії” → обрати пільгову категорію зі списку та натиснути “Редагувати” → змінити необхідні поля → натиснути “Оновити”.

Назначити учню пільгову категорію: обрати учня зі списку та натиснути “Детальніше” → натиснути “Пільгові категорії учня” → натиснути “Додати пільгову категорію” → обрати пільгову категорію зі списку, обрати/ввести дату у поле вводу та натиснути “Обрати”.

Зняти пільгову категорію з учня: обрати учня зі списку та натиснути “Детальніше” → натиснути “Пільгові категорії учня” → обрати пільгову категорію зі списку та натиснути “Видалити зв'язок”.

Змінити термін пільгової категорії для учня: обрати учня зі списку та натиснути “Детальніше” → натиснути “Пільгові категорії учня” → обрати пільгову категорію зі списку та натиснути “Змінити термін” → обрати/ввести нову дату → натиснути “Оновити”.

Додати розклад: натиснути “Розклад” → натиснути “Додати розклад” → обрати предмет зі списку та натиснути “Обрати” → обрати вчителя зі списку та натиснути “Обрати” → обрати клас зі списку та натиснути “Назначити клас” або натиснути “Назначити усім” (тільки 1 опція доступна одночасно) → заповнити усі поля вводу → натиснути “Зберегти”.

Редагувати розклад: натиснути “Розклад” → обрати розклад зі списку та натиснути “Редагувати” → змінити необхідні поля вводу → натиснути “Оновити”.

Видалити розклад: натиснути “Розклад” → обрати розклад зі списку та натиснути “Видалити”.

Редагувати ціни за навчання: натиснути “Додатково” → натиснути “Розцінки” → обрати розцінку зі списку та натиснути “Редагувати” → змінити поле вводу на необхідне → натиснути “Оновити”.

Додати учню оцінку: обрати учня зі списку та натиснути “Детальніше” → натиснути “Оцінки учня” → натиснути “Додати оцінку” → в одному з двох списків обрати предмет та натиснути “Обрати” → обрати вчителя та натиснути “Обрати” → заповнити усі поля вводу → натиснути “Зберегти”.

Редагувати оцінку учня: обрати учня зі списку та натиснути “Детальніше” → натиснути “Оцінки учня” → обрати оцінку зі списку та натиснути редагувати → змінити необхідні поля → натиснути “Оновити”.

Видалити оцінку учню: обрати учня зі списку та натиснути “Детальніше” → натиснути “Оцінки учня” → обрати оцінку зі списку та натиснути “Видалити”.

Виставити всім учням підсумкові оцінки: натиснути “Додатково” → натиснути “Виставити всім учням підсумкові оцінки”.

Додати учню підсумкову оцінку: обрати учня зі списку та натиснути “Детальніше” → натиснути “Підсумкові оцінки учня” → натиснути “Додати оцінку” → в одному з двох списків обрати предмет та натиснути “Обрати” → обрати навчальний рік зі списку та натиснути “Зберегти”.

Редагувати підсумкову оцінку учня: обрати учня зі списку та натиснути “Детальніше” → натиснути “Підсумкові оцінки учня” → обрати підсумкову оцінку зі списку та натиснути “Редагувати” → змінити необхідні поля та натиснути “Оновити”.

Видалити підсумкову оцінку учня: обрати учня зі списку та натиснути “Детальніше” → натиснути “Підсумкові оцінки учня” → обрати підсумкову оцінку зі списку та натиснути “Видалити”.

Додати рахунок учню: обрати учня зі списку та натиснути “Детальніше” → натиснути “Рахунки учня” → натиснути “Додати рахунок” → ввести поле вводу → натиснути “Зберегти”.

Відмінити рахунок: обрати учня зі списку та натиснути “Детальніше” → натиснути “Рахунки учня” → обрати рахунок зі списку та натиснути “Відмінено”.

Сплатити рахунок: обрати учня зі списку та натиснути “Детальніше” → натиснути “Рахунки учня” → обрати рахунок зі списку та натиснути “Сплачено”.

Завантажити фотографію учня: обрати учня зі списку та натиснути

“Детальніше” → у нижній частині екрану обрати завантаження файлу → обрати у потрібний файл у вікні завантаження файлу → натиснути “Завантажити” у вікні завантаження файлу → натиснути “Завантажити фото”.

Сформувати повний звіт щодо учня: обрати учня зі списку та натиснути “Детальніше” → натиснути “Сформувати повний звіт”.

Сформувати фінансову звітність учня: обрати учня зі списку та натиснути “Детальніше” → натиснути “Сформувати фінансову звітність”.

Сформувати медичний звіт учня: обрати учня зі списку та натиснути “Детальніше” → натиснути “Сформувати медичний звіт”.

Сформувати табель успішності учня: обрати учня зі списку та натиснути “Детальніше” → натиснути “Сформувати табель успішності”.

Завуч:

Додати предмет: натиснути “Предмети” → заповнити усі поля вводу → натиснути “Додати предмет”.

Редагування предмету: натиснути “Предмети” → обрати предмет зі списку та натиснути “Редагувати” → внести зміни у потрібні поля вводу → натиснути “Оновити”.

Призначити вчителя на викладання предмету: натиснути “Предмети” → обрати предмет зі списку та натиснути “Вчителі” → натиснути “Додати вчителя” → обрати вчителя зі списку та натиснути “Обрати”.

Зняти вчителя з викладання предмету: натиснути “Предмети” → обрати предмет зі списку та натиснути “Вчителі” → обрати вчителя зі списку та натиснути “Видалити зв'язок”.

Оновити всім учням предмети відповідно до класу навчання: натиснути “Додатково” → натиснути “Оновити всім учням предмети відповідно до класу навчання”.

Призначити учню додатковий предмет: обрати учня зі списку та натиснути “Предмети учня” → натиснути “Поставити додатковий предмет” → обрати предмет зі списку й натиснути “Обрати”.

Зняти учня з предмету: обрати учня зі списку та натиснути “Предмети учня” → обрати предмет зі списку й натиснути “Видалити зв'язок”.

Додати учню оцінку: обрати учня зі списку та натиснути “Оцінки учня” → натиснути “Додати оцінку” → в одному з двох списків обрати предмет та натиснути “Обрати” → обрати вчителя та натиснути “Обрати” → заповнити усі поля вводу → натиснути “Зберегти”.

Редагувати оцінку учня: обрати учня зі списку та натиснути “Оцінки учня” → обрати оцінку зі списку та натиснути редагувати → змінити необхідні поля → натиснути “Оновити”.

Видалити оцінку учню: обрати учня зі списку та натиснути “Оцінки учня” → обрати оцінку зі списку та натиснути “Видалити”.

Виставити всім учням підсумкові оцінки: натиснути “Додатково” → натиснути “Виставити всім учням підсумкові оцінки”.

Додати учню підсумкову оцінку: обрати учня зі списку та натиснути “Підсумкові оцінки” → натиснути “Додати оцінку” → в одному з двох списків обрати предмет та натиснути “Обрати” → обрати навчальний рік зі списку та натиснути “Зберегти”.

Редагувати підсумкову оцінку учня: обрати учня зі списку та натиснути “Підсумкові оцінки” → обрати підсумкову оцінку зі списку та натиснути “Редагувати” → змінити необхідні поля та натиснути “Оновити”.

Видалити підсумкову оцінку учня: обрати учня зі списку та натиснути “Підсумкові оцінки” → обрати підсумкову оцінку зі списку та натиснути “Видалити”.

Додати розклад: натиснути “Розклад” → натиснути “Додати розклад” → обрати предмет зі списку та натиснути “Обрати” → обрати вчителя зі списку та натиснути “Обрати” → обрати клас зі списку та натиснути “Назначити клас” або натиснути “Назначити усім” (тільки 1 опція доступна одночасно) → заповнити усі поля вводу → натиснути “Зберегти”.

Редагувати розклад: натиснути “Розклад” → обрати розклад зі списку та натиснути “Редагувати” → змінити необхідні поля вводу → натиснути

“Оновити”.

Видалити розклад: натиснути “Розклад” → обрати розклад зі списку та натиснути “Видалити”.

Сформувати табель успішності учня: обрати учня зі списку та натиснути “Табель успішності”.

Медсестра:

Редагувати медичну картку учня: брати учня зі списку та натиснути “Мед картка” → натиснути “Редагувати” → змінити необхідні поля → натиснути “Оновити”.

Сформувати медичний звіт учня: обрати учня зі списку та натиснути “Мед Картка” → натиснути “Звіт”.

Бухгалтер:

Додати пільгову категорію: натиснути “Пільгові категорії” → заповнити усі поля вводу → натиснути “Додати пільгову категорію”.

Редагувати пільгову категорію: натиснути “Пільгові категорії” → обрати пільгову категорію зі списку та натиснути “Редагувати” → змінити необхідні поля → натиснути “Оновити”.

Назначити учню пільгову категорію: обрати учня зі списку та натиснути “Пільгові категорії” → натиснути “Додати пільгову категорію” → обрати пільгову категорію зі списку, обрати/ввести дату у поле вводу та натиснути “Обрати”.

Зняти пільгову категорію з учня: обрати учня зі списку та натиснути “Пільгові категорії” → обрати пільгову категорію зі списку та натиснути “Видалити зв'язок”.

Змінити термін пільгової категорії для учня: обрати учня зі списку та натиснути “Пільгові категорії” → обрати пільгову категорію зі списку та натиснути “Змінити термін” → обрати/ввести нову дату → натиснути “Оновити”.

Додати рахунок учню: обрати учня зі списку та натиснути “Рахунки” → натиснути “Додати рахунок” → ввести поле вводу → натиснути “Зберегти”.

Відмінити рахунок: обрати учня зі списку та натиснути “Рахунки” → обрати рахунок зі списку та натиснути “Відмінено”.

Сплатити рахунок: обрати учня зі списку та натиснути “Рахунки” → обрати рахунок зі списку та натиснути “Сплачено”.

Сформувати фінансову звітність учня: обрати учня зі списку та натиснути “Звіт”.

Вчитель:

Додати учню оцінку: обрати предмет зі списку та натиснути “Учні” → обрати учня зі списку та натиснути “Поставити” → заповнити усі поля вводу → натиснути “Зберегти”.

Редагувати оцінку учня: обрати предмет зі списку та натиснути “Учні” → обрати учня зі списку та натиснути “оцінки” → обрати оцінку зі списку та натиснути редагувати → змінити необхідні поля → натиснути “Оновити”.

Видалити оцінку учню: обрати предмет зі списку та натиснути “Учні” → обрати учня зі списку та натиснути “оцінки” → обрати оцінку зі списку та натиснути “Видалити”.

Учень:

Переглянути розклад занять: натиснути “Розклад учня”.

Переглянути оцінки: натиснути “Оцінки учня”.

Переглянути підсумкові оцінки: натиснути “Підсумкові оцінки учня”.

ДОДАТОК Д Програма та методика випробовування програмного забезпечення

1 Об'єкт випробувань

1.1 Найменування випробуваної програми

Об'єктом до випробовування є програмне забезпечення для автоматизації обліку навчального процесу приватної школи “CIA: свідомість, іновації, академічність” (далі – “CIA”).

1.2 Призначення програмного забезпечення

Програма призначена для поліпшення обліку навчального процесу у приватній школі “CIA”. Розроблене ПЗ повинно забезпечувати:

– авторизацію користувачів;

– зберігання та обробку інформації про учнів, вчителів, представників учнів, медичні картки учнів, пільгові категорії учнів, оцінки, підсумкові оцінки, предмети, рахунки, розцінки за навчання, розклад;

– формування звітності;

– надання інструкції користувача.

2 Мета випробувань

Метою проведення випробувань є перевірка відповідності програмного забезпечення вимогам, які викладені у технічному завданні, котре знаходиться у додатку А.

3 Вимоги до програми

Перевірка відповідності функціональних вимог, розробленого програмного забезпечення, функціональним вимогам сформульованим у технічному завданню, наведеному у додатку А.

4 Вимоги до програмної документації

Для проведення випробування повинна бути надана така програмна документація:

– технічне завдання;

– текст програми;

– опис програми;

– інструкція користувача;

– програма та методика випробувань ПЗ.

5 Засоби і порядок випробувань

5.1 Склад технічних засобів, що будуть використовуватись під час випробувань

Мінімальна комплектація, яка необхідна для проведення випробувань:

– CPU: Intel Core i3–10100 3,6 ГГц або аналогічний за характеристиками AMD;

– RAM: 4 ГБ;

– з'єднання по мережі.

5.2 Склад програмних засобів, використаних під час випробувань

Для проведення випробувань потрібні наступні програмні засоби:

– Java Development Kit (JDK);

– система управління базами даних Mariadb;

– операційна система не нижче Windows 10;

– браузер Google Chrome версії не нижче 93.0.4577.82 або браузер Mozilla Firefox версії не нижче 91.0.

5.3 Порядок проведення випробувань

Випробування проводяться у відповідності з наступним порядком:

– виконання інструкцій до кожної функції;

– аналіз отриманого результату і встановлення відповідності ПЗ до

документації.

При виявленні помилок у роботі програми, складається їх перелік й обговорюються терміни виправлення з розробником. Після цього, проводять повторне тестування.

6 Методи випробувань

Випробування буде проводитись за стратегією “білої скриньки”. Оскільки програма має велику кількість тестованих функцій, представлено результати випробувань декількох функцій програми.

6.1 Функціональна вимога “Додавання предмету”

Логічна структура функції додавання предмету представлена на рисунку Д.1, а у таблиці Д.1 представлені тести.

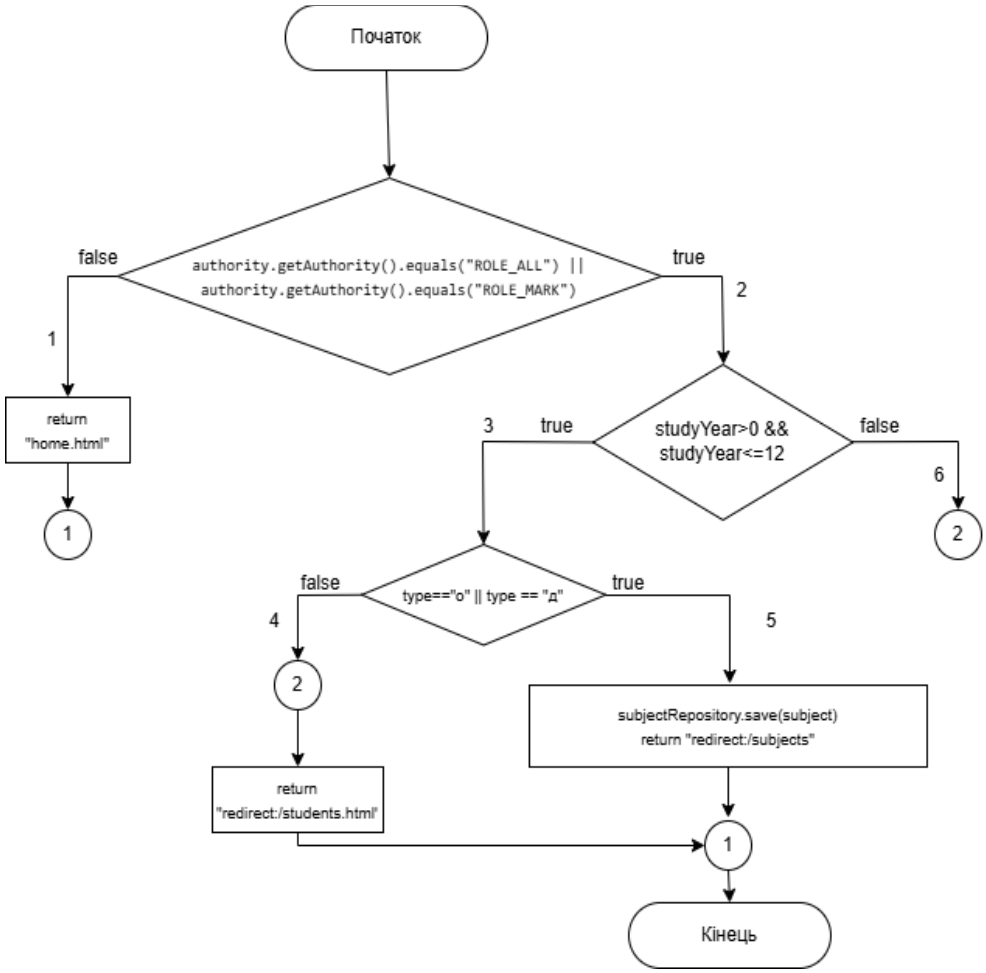


Рисунок Д.1 – Логічна структура функції додавання предмету

Тестові приклади за методом білої скриньки

Тести номер 1, 2, 3, 4 забезпечують покриття за критерієм тестування умов.

Тести номер 5, 6, 7, 8 забезпечують покриття за критерієм тестування гілок.

Таблиця Д.1 – Тести функції додавання предмету

N тесту	шлях	Вхідні умови	Тестові дані	Вірні вихідні умови
1	1	авторизація з правом “ROLE_TEACH”, name=рядок, type=”о” або “д”, 0<studyYear<=12	role = “ROLE_TEACH” name=”Геометрія”, type=”о”, studyYear=10	БД: без змін Перехід: home.html
2	1	авторизація з правом “ROLE_HEALTH”, name=рядок, type=”о” або “д”, 0<studyYear<=12	role = “ROLE_HEALTH” name=”Геометрія”, type=”о”, weekHours=3.5, studyYear=10	БД: без змін Перехід: home.html
3	1	авторизація з правом “ROLE_MONEY”, name=рядок, type=”о” або “д”, 0<studyYear<=12	role = “ROLE_MONEY” name=”Геометрія”, type=”о”, studyYear=10	БД: без змін Перехід: home.html
4	1	авторизація з правом “ROLE_STUDY”, name=рядок, type=”о” або “д”, 0<studyYear<=12	role = “ROLE_STUDY” name=”Геометрія”, type=”о”, studyYear=10	БД: без змін Перехід: home.html
5	2,3,4	авторизація з правом “ROLE_ALL”, name=рядок, type != ”о” або “д”, 0<studyYear<=12	role = “ROLE_ALL” name=”Геометрія”, type=”У”, studyYear=10	БД: без змін Перехід: students.html
6	2,3,5	авторизація з правом “ROLE_ALL”, name=рядок, type = ”о” або “д”, 0<studyYear<=12	role = “ROLE_ALL” name=”Геометрія”, type=”о”, studyYear=10	БД: додавання нового предмету з параметрами name=”Геометрія”, type=”о”, weekHours=3.5, studyYear=10 Перехід: subjects.html
7	2,6	авторизація з правом “ROLE_ALL”, name=рядок, type = ”о” або “д”, studyYear>12	role = “ROLE_ALL” name=”Геометрія”, type=”о”, studyYear=14	БД: без змін Перехід: students.html
8	2,6	авторизація з правом “ROLE_ALL”, name=рядок, type = ”о” або “д”, studyYear<=0	role = “ROLE_ALL” name=”Геометрія”, type=”о”, studyYear= -3	БД: без змін Перехід: students.html

6.2 Функціональна вимога “Додавання розкладу”

Логічна структура функції додавання розкладу представлена на рисунку Д.2, а у таблиці Д.2 представлені тести.

Рисунок Д.2 – Логічна структура функції додавання розкладу

Тестові приклади за методом білої скриньки

Тести номер 1, 2, 3, 4 забезпечують покриття за критерієм тестування умов.

Тести номер 5, 6, 7, 8, 9 забезпечують покриття за критерієм тестування гілок.

Таблиця Д.2 – Тести функції додавання розкладу

N тесту	шлях	Вхідні умови	Тестові дані	Вірні вихідні умови
1	2	3	4	5
1	1	авторизація з правом “ROLE_TEACH”, subjectId ідентифікатор існуючого у базі даних предмету, teacherId ідентифікатор існуючого у базі даних вчителя, classField = рядок frequency=рядок що має одне зі значень “щотижня”, “парні тижні”, “непарні тижні”; weekday = рядок що має одне зі значень “Пн”, “Вт”, “Ср”, “Чт”, “Пт”, “Сб”, “Нд”, begin=час, end=час, studyRoom>0	role = “ROLE_TEACH” subjectId=1, teacherId=1, classfield=”11-Б”, frequency=”щотижня”, weekday = “Вт”, begin=10:35, end = “11:20”, studyRoom=31	БД: без змін Перехід: home.html
2	1	авторизація з правом “ROLE_HEALTH”, subjectId ідентифікатор існуючого у базі даних предмету, teacherId ідентифікатор існуючого у базі даних вчителя, classField = рядок frequency=рядок що має одне зі значень “щотижня”, “парні тижні”, “непарні тижні”; weekday = рядок що має одне зі значень “Пн”, “Вт”, “Ср”, “Чт”, “Пт”, “Сб”, “Нд”, begin=час, end=час, studyRoom>0	role = “ROLE_HEALTH” subjectId=1, teacherId=1, classfield=”11-Б”, frequency=”щотижня”, weekday = “Вт”, begin=10:35, end = “11:20”, studyRoom=31	БД: без змін Перехід: home.html

Продовження таблиці Д.2

1	2	3	4	5
3	1	авторизація з правом “ROLE_MONEY”, subjectId ідентифікатор існуючого у базі даних предмету, teacherId ідентифікатор існуючого у базі даних вчителя, classField = рядок frequency=рядок що має одне зі значень “щотижня”, “парні тижні”, “непарні тижні”; weekday = рядок що має одне зі значень “Пн”, “Вт”, “Ср”, “Чт”, “Пт”, “Сб”, “Нд”, begin=час, end=час, studyRoom>0	role = “ROLE_MONEY” subjectId=1, teacherId=1, classfield=”11-Б”, frequency=”щотижня”, weekday = “Вт”, begin=10:35, end = “11:20”, studyRoom=31	БД: без змін Перехід: home.html
4	1	авторизація з правом “ROLE_STUDY”, subjectId ідентифікатор існуючого у базі даних предмету, teacherId ідентифікатор існуючого у базі даних вчителя, classField = рядок frequency=рядок що має одне зі значень “щотижня”, “парні тижні”, “непарні тижні”; weekday = рядок що має одне зі значень “Пн”, “Вт”, “Ср”, “Чт”, “Пт”, “Сб”, “Нд”, begin=час, end=час, studyRoom>0	role = “ROLE_STUDY” subjectId=1, teacherId=1, classfield=”11-Б”, frequency=”щотижня”, weekday = “Вт”, begin=10:35, end = “11:20”, studyRoom=31	БД: без змін Перехід: home.html
5	2,3,4, 5	авторизація з правом “ROLE_ALL”, subjectId ідентифікатор НЕ існуючого у базі даних предмету, teacherId ідентифікатор НЕ існуючого у базі даних вчителя, classField = рядок frequency=рядок що має одне зі значень “щотижня”, “парні тижні”, “непарні тижні”; weekday = рядок що має НЕ має жодного зі значень “Пн”, “Вт”, “Ср”, “Чт”, “Пт”, “Сб”, “Нд”, begin=час, end=час, studyRoom>0	role = “ROLE_ALL” subjectId=1, teacherId=1, classfield=”11-Б”, frequency=”щотижня”, weekday = “Гл”, begin=10:35, end = “11:20”, studyRoom=31	БД: без змін Перехід: students.html

Продовження таблиці Д.2

1	2	3	4	5
6	2,3,4,6,7	авторизація з правом “ROLE_ALL” subjectId ідентифікатор існуючого у базі даних предмету, teacherId ідентифікатор не існуючого у базі даних вчителя, classField = рядок frequency=рядок що має одне зі значень “щотижня”, “парні тижні”, “непарні тижні”; weekday = рядок що має одне зі значень “Пн”, “Вт”, “Ср”, “Чт”, “Пт”, “Сб”, “Нд”, begin=час, end=час, studyRoom>0	role = “ROLE_ALL” subjectId=1, teacherId=1, classfield=”11-Б”, frequency=”щотижня”, weekday = “Вт”, begin=10:35, end = “11:20”, studyRoom= - 8	БД: без змін Перехід: students.html
7	2,3,4,6,8	авторизація з правом “ROLE_ALL” subjectId ідентифікатор існуючого у базі даних предмету, teacherId ідентифікатор не існуючого у базі даних вчителя, classField = рядок frequency=рядок що має одне зі значень “щотижня”, “парні тижні”, “непарні тижні”; weekday = рядок що має одне зі значень “Пн”, “Вт”, “Ср”, “Чт”, “Пт”, “Сб”, “Нд”, begin=час, end=час, studyRoom<0	role = “ROLE_ALL” subjectId=1, teacherId=1, classfield=”11-Б”, frequency=”щотижня”, weekday = “Вт”, begin=10:35, end = “11:20”, studyRoom=31	БД: додавання новий розклад з параметрами subjectId=1, teacherId=1, classfield=”11-Б”, frequency=”щотижня”, weekday = “Вт”, begin=10:35, end = “11:20”, studyRoom=31 Перехід: schedule.html
8	2,3,9	авторизація з правом “ROLE_ALL” subjectId ідентифікатор існуючого у базі даних предмету, teacherId ідентифікатор не існуючого у базі даних вчителя, classField = рядок frequency=рядок що НЕМАЄ жодного зі значень “щотижня”, “парні тижні”, “непарні тижні”; weekday = рядок що має одне зі значень “Пн”, “Вт”, “Ср”, “Чт”, “Пт”, “Сб”, “Нд”, begin=час, end=час, studyRoom<0	role = “ROLE_ALL” subjectId=1, teacherId=1, classfield=”11-Б”, frequency=”час від часу”, weekday = “Вт”, begin=10:35, end = “11:20”, studyRoom=31	БД: без змін Перехід: students.html

Кінець таблиці Д.2

1	2	3	4	5
9	2,10	авторизація з правом “ROLE_ALL” subjectId ідентифікатор існуючого у базі даних предмету, teacherId ідентифікатор не існуючого у базі даних вчителя, classField = рядок frequency=рядок що має одне зі значень “щотижня”, “парні тижні”, “непарні тижні”; weekday = рядок що має одне зі значень “Пн”, “Вт”, “Ср”, “Чт”, “Пт”, “Сб”, “Нд”, begin=час, end=час, studyRoom<0	role = “ROLE_ALL” subjectId=99999, teacherId=8888, classfield=”11-Б”, frequency=”щотижня”, weekday = “Вт”, begin=10:35, end = “11:20”, studyRoom=31	БД: без змін Перехід: students.html