МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами

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 застосунку для автоматизації формування розкладу занять в університеті «Розклад занять в НУК»

Виконав: студент групи 4151

_____ Іванов С.О.

(підпис, ПІБ)

Керівник роботи:

к.т.н., доцент_____

(посада, науковий ступень вчене звання)

_____ Фаріонова Т.А.

(підпис, ПІБ)

Миколаїв – 2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами
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 Іванову Станіславу Олеговичу
(Прізвище, ім'я, по батькові)

1. Тема роботи: Розробка програмного застосунку для автоматизації формування розкладу занять в університеті «Розклад занять в НУК»

Керівник роботи Фаріонова Тетяна Анатоліївна, к.т.н., доцент
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. Актуальність теми. 3. Мета, об'єкт роботи. 4. Задачі кваліфікаційної роботи; 5. Аналіз вимог до ПЗ; 6 Архітектура ПЗ; 7 Діаграма діяльності прецедентів ПЗ; 8. Модель бази даних; 9 Динамічна модель ПЗ; 10 Вибір засобів розробки програмного застосунку; 11-12 Результати розробки; 13-14. Висновки; 15 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.

КАЛЕНДАРНИЙ ПЛАН

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 проекту 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 _____ Іванов С.О.
(підпис) (ПІБ)

Керівник роботи _____ Фаріонова Т.А.
(підпис) (ПІБ)

АНОТАЦІЯ

Кваліфікаційна робота присвячена вирішенню практичної задачі інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов, а саме розробці програмного застосунку для автоматизації формування розкладу занять в університеті ”Розклад занять НУК”.

Метою роботи є розробка програмного застосунку для автоматизації формування розкладу занять в університеті ”Розклад занять НУК”, що дозволить підвищити ефективність роботи диспетчерської служби університету, зменшити витрат часу на адміністрування, мінімізувати імовірність виникнення помилок при введені та обробки даних. Об’єктом є процес розробки програмного забезпечення для автоматизації формування розкладу занять в університеті.

В ході дипломного проєктування були розглянути особливості автоматичного формування розкладу; на основі аналізу існуючих програмних продуктів для автоматизації складання розкладу обґрунтовано необхідність розробки програмного застосунку для автоматизації формування розкладу занять в університеті; сформульовані вимоги до ПЗ; виконані всі етапи проектування програмного забезпечення (ескізний, технічний, робочий); розглянути питання охорони праці у відділі розробки ПЗ; оформлена необхідна програмна документація.

Розробку програмного забезпечення було виконано за допомогою середовища розробки Visual Studio Code, мови стилізації CSS, мов програмування JavaScript та Python, фреймворку Flask.

Кваліфікаційна робота викладена на 108 сторінках друкованого тексту, містить 53 рисунок, 4таблиці, 5 додатків та список використаних джерел із 12 найменувань.

ABSTRACT

The qualification work is devoted to solving a practical problem of software engineering, characterised by complexity and uncertainty of conditions, namely the development of a software application for automating the formation of a class schedule at the university ‘NUK Class Schedule’.

The aim of the work is to develop a software application for automating the formation of the university class schedule ‘NUK Class Schedule’, which will increase the efficiency of the university dispatch service, reduce the time spent on administration, minimise the likelihood of errors in data entry and processing. The object is the process of developing software to automate the formation of class schedules at the university.

During the course of the diploma project, the features of automatic timetabling were considered; based on the analysis of existing software products for automating timetabling, the need to develop a software application for automating the formation of a university timetable was substantiated; software requirements were formulated; all stages of software design (draft, technical, working) were completed; occupational safety issues in the software development department were considered; the necessary software documentation was drawn up.

The software was developed using the Visual Studio Code development environment, CSS styling language, JavaScript and Python programming languages, and the Flask framework.

The qualification work is presented on 108 pages of printed text, contains 53 figures, 4 tables, 5 appendices, a list of references of 15 titles.

ЗМІСТ

ВСТУП 7
1 АНАЛІЗ ПРАКТИЧНОЇ ЗАДАЧІ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ З РОЗРОБКИ ПРОГРАМНОГО ЗАСТОСУНКУ ДЛЯ АВТОМАТИЗАЦІЇ ФОРМУВАННЯ РОЗКЛАДУ ЗАНЯТЬ В УНІВЕРСИТЕТІ «РОЗКЛАД ЗАНЯТЬ НУК» 9
1.1 Аналіз практичної задачі інженерії програмного забезпечення 9
1.2 Аналіз існуючих програмних продуктів для планування розкладу та обґрунтування необхідності розробки програмного застосунку для автоматизації формування розкладу занять в університеті «Розклад занять в НУК» 10
1.3 Постановка задачі на розробку програмного застосунку для автоматизації формування розкладу занять в університеті «Розклад занять в НУК» 13
2 РОЗРОБКА ПРОГРАМНОГО ЗАСТОСУНКУ ДЛЯ АВТОМАТИЗАЦІЇ ФОРМУВАННЯ РОЗКЛАДУ ЗАНЯТЬ В УНІВЕРСИТЕТІ «РОЗКЛАД ЗАНЯТЬ В НУК» 19
2.1 Аналіз вимог до програмного забезпечення 19
2.1.1 Побудова моделі варіантів використання програмного застосунку для автоматизації формування розкладу занять в університеті «Розклад занять в НУК» 19
2.1.2 Специфікації варіантів використання 23
2.2 Розробка архітектури програмного застосунку для автоматизації формування розкладу занять в університеті «Розклад занять в НУК» 24
2.3 Проєкт програмного забезпечення 27
2.3.1 Ескізний проєкт програмного забезпечення 27
2.3.1.1 Розробка діаграм діяльності прецедентів програмного забезпечення 27

2.3.1.2 Проєкт користувацького інтерфейсу програмного забезпечення 29
2.3.2 Технічний проєкт програмного забезпечення 31
2.3.2.1 Структура бази даних ПЗ 31
2.3.2.2 Динамічна модель програмного забезпечення 34
2.4 Робочий проєкт. Реалізація програмного застосунку для автоматизації формування розкладу занять в університеті «Розклад занять в НУК» 35
2.4.1 Вибір та обґрунтування засобів розробки програмного забезпечення 35
2.4.2 Опис та реалізація ключових функцій 36
2.4.3 Тестування та випробування програмного застосунку для автоматизації формування розкладу занять в університеті «Розклад занять в НУК» 45
3 РЕЗУЛЬТАТИ РОЗРОБКИ ПРОГРАМНОГО ЗАСТОСУНКУ ДЛЯ АВТОМАТИЗАЦІЇ ФОРМУВАННЯ РОЗКЛАДУ ЗАНЯТЬ В УНІВЕРСИТЕТІ «РОЗКЛАД ЗАНЯТЬ В НУК» 48
4 ОХОРОНА ПРАЦІ У ВІДІЛЛІ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 57
4.1 Аналіз небезпечних і шкідливих факторів у офісному приміщенні з персональними комп'ютерами 57
4.2 Розрахунок системи штучного освітлення у офісному приміщені з персональними комп'ютерами 62
ВИСНОВКИ 65
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 66
ДОДАТОК А. ТЕХНІЧНЕ ЗАВДАННЯ 68
ДОДАТОК Б – Вихідні тексти програмних модулів 73
ДОДАТОК В – Опис програмного забезпечення 87
ДОДАТОК Г – Інструкція користувача 93
ДОДАТОК Д – Програма та методика випробувань програмного забезпечення 105

ВСТУП

У сучасному світі цифрова трансформація охоплює всі аспекти нашого життя, зокрема й сферу освіти. Ефективне управління навчальним процесом вимагає автоматизації численних процедур, однією з яких є складання розкладу занять. Розклад є важливою частиною організації навчального процесу, що визначає, коли і де будуть проходити заняття, хто їх проводитиме, які ресурси використовуються [1].

У більшості навчальних закладів розклад формується вручну або з використанням застарілих інструментів, що призводить до значних витрат часу, виникнення конфліктів між парами, перевантаження викладачів і неефективного використання ресурсів. Крім того, зміни в розкладі часто до помилок, тому викає необхідність автоматизувати цей процес за рахунок розробки спеціалізованого програмного застосунку що обумовлює актуальність теми роботи.

Кваліфікаційна робота присвячена розв’язанню практичної задачі інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов, а саме розробці програмного застосунку для автоматизації формування розкладу занять в університеті ”Розклад занять НУК”

Метою кваліфікаційної роботи є розробка програмного застосунку для автоматизації формування розкладу занять в університеті ”Розклад занять НУК”, що дозволить підвищити ефективність роботи диспетчерської служби університету, зменшити витрат часу на адміністрування, мінімізувати імовірність виникнення помилок при введені та обробки даних. Використання ПЗ сприятиме підвищенню задоволеності користувачів за рахунок зручного інтерфейсу та широких можливостей керування.

Для досягнення поставленої мети в ході дипломного проектування необхідно виконати наступні задачі:

- розглянути особливості автоматичного формування розкладу;

- проаналізувати існуючі програмні засоби для автоматизації складання розкладу;

- обґрунтувати необхідність розробки програмного застосунку для автоматизації формування розкладу занять в університеті ”Розклад занять НУК”;

- виконати постанову задачі на розробку ПЗ;

- сформулювати та задокументувати вимоги на розробку програмного застосунку для автоматизації формування розкладу занять в університеті ”Розклад занять НУК”;

- виконати всі етапи проектування програмного забезпечення (ескізний, технічний, робочий)

- розглянути питання охорони праці у відділі розробки програмного забезпечення;

- розробити всю необхідну програмну документацію.

Об’єктом є процес розробки програмного забезпечення для автоматизації формування розкладу занять в університеті.

1 АНАЛІЗ ПРАКТИЧНОЇ ЗАДАЧІ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ З РОЗРОБКИ ПРОГРАМНОГО ЗАСТОСУНКУ ДЛЯ АВТОМАТИЗАЦІЇ ФОРМУВАННЯ РОЗКЛАДУ ЗАНЯТЬ В УНІВЕРСИТЕТІ «РОЗКЛАД ЗАНЯТЬ НУК»

1.1 Аналіз практичної задачі інженерії програмного забезпечення

Складання навчального розкладу є складною задачею, яка вимагає урахування численних обмежень: наявність вільних аудиторій, завантаженість викладачів, особливості навчальних планів тощо. З метою автоматизації цього процесу широко використовуються спеціалізовані алгоритми, які дозволяють ефективно розподіляти ресурси та формувати оптимальні розклади.

Історично першими були застосовані методи математичного програмування, зокрема лінійне програмування, що дозволяло отримати рішення шляхом формалізації обмежень і цілей у вигляді системи рівнянь. Однак такі методи виявилися неефективними при великій кількості параметрів, через що надалі все ширше використовуються евристичні та метаевристичні підходи [1].

Серед найпоширеніших підходів можна виокремити:

- генетичний алгоритм, який базується на принципах природного добору. Його суть полягає у створенні популяції можливих рішень, селекції найкращих, їх схрещенні та мутації для отримання нових, більш ефективних варіантів. Генетичні алгоритми здатні знаходити глобальні оптимуми, однак вимагають значних обчислювальних ресурсів і складної настройки [1-3];

- алгоритм табу-пошуку працює як покращений локальний пошук, уникаючи повторного відвідування вже розглянутих рішень за рахунок ведення табу-списку. Такий підхід добре справляється з локальними мінімумами, однак не завжди гарантує отримання найкращого результату [2-3];

- жадібний алгоритм (greedy algorithm), який ґрунтується на прийнятті локально оптимальних рішень на кожному кроці, що спрощує його реалізацію та пришвидшує роботу [4]. Незважаючи на те, що такий підхід не гарантує досягнення глобального оптимуму, він є надзвичайно корисним у випадках, коли необхідна швидка генерація прийнятного рішення.

У межах цієї роботи для реалізації функціоналу автоматичного складання розкладу було обрано саме жадібний алгоритм, що дозволяє швидко й ефективно призначати заняття у доступні часові слоти. Основною перевагою цього підходу є його простота, швидкодія та зручність у реалізації, що робить його придатним для інтеграції у веб-застосунок, орієнтований на щоденне використання.

Таким чином, жадібна стратегія формування розкладу дозволить побудувати базову функціональність системи, з можливістю подальшого розширення за рахунок адаптації більш складних алгоритмів оптимізації.

1.2 Аналіз існуючих програмних продуктів для планування розкладу та обґрунтування необхідності розробки програмного застосунку для автоматизації формування розкладу занять в університеті «Розклад занять в НУК»

Для успішного проєктування та розробки застосунку, який дозволить автоматизувати формування розкладу, важливо провести аналіз існуючих програмних продуктів, що вже реалізують подібний функціонал. Таке дослідження дозволяє оцінити сучасний стан розробок у цій галузі, виявити сильні та слабкі сторони реалізацій, а також врахувати користувацький досвід при формуванні власних вимог до програмного забезпечення.

Однією з найвідоміших в Україні комерційних систем є «1С: Підприємство» [5], яка має модуль для автоматизації діяльності навчальних закладів. До її можливостей належать управління навчальними планами,

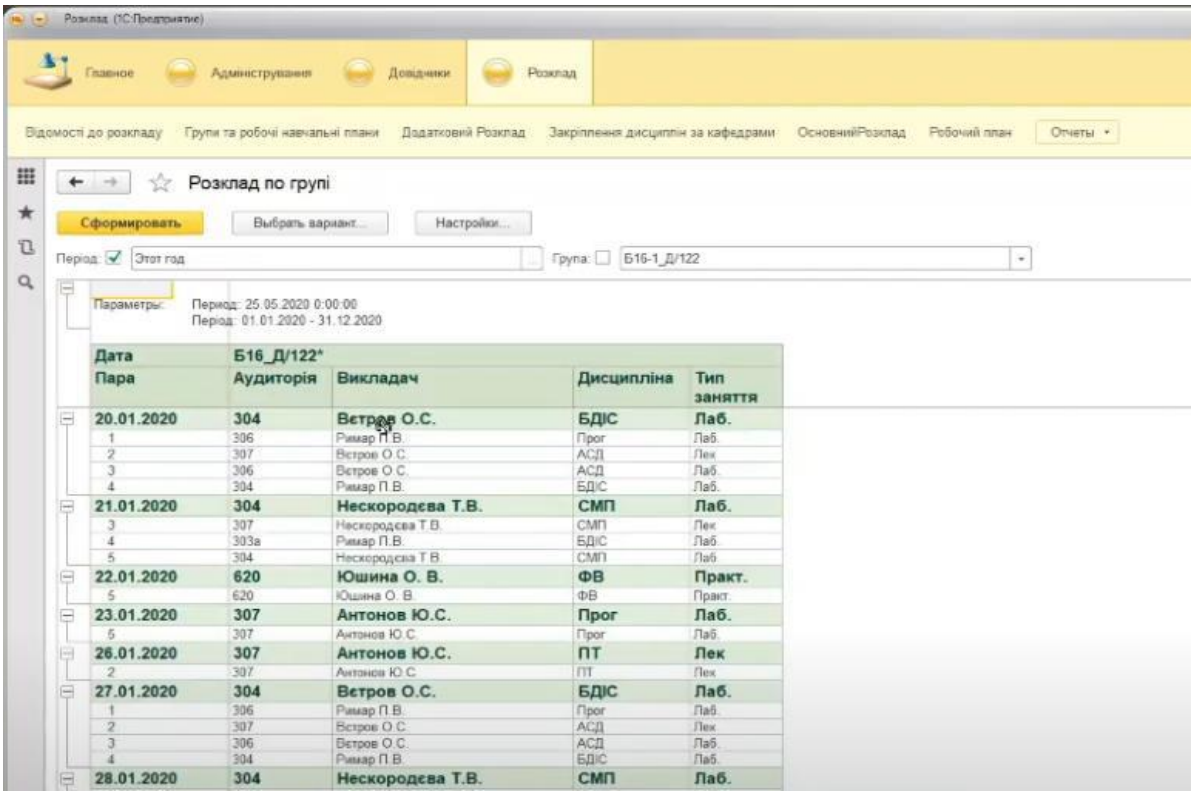
контроль відвідуваності, фінансовий облік, а також автоматичне складання розкладу занять. Однак система є складною в налаштуванні, має обмеження по інтеграції з сучасними веб-технологіями та потребує платної ліцензії.

Переваги:

- комплексна інтеграція з іншими модулями системи;
- широкі функціональні можливості;
- підтримка української мови та локалізованої звітності.

Недоліки:

- висока вартість впровадження;
- складність у використанні без спеціальної підготовки;
- обмежена адаптація під індивідуальні потреби університетів.



Дата / Пара	Б16_Д/122* Аудиторія	Викладач	Дисципліна	Тип заняття
20.01.2020	304	Вєтров О.С.	БДІС	Лаб.
1	306	Римар П.В.	Прог	Лаб.
2	307	Вєтров О.С.	АСД	Лек
3	306	Вєтров О.С.	АСД	Лаб.
4	304	Римар П.В.	БДІС	Лаб.
21.01.2020	304	Нескородєва Т.В.	СМП	Лаб.
3	307	Нескородєва Т.В.	СМП	Лек
4	303а	Римар П.В.	БДІС	Лаб.
5	304	Нескородєва Т.В.	СМП	Лаб.
22.01.2020	620	Юшина О. В.	ФВ	Практ.
5	620	Юшина О. В.	ФВ	Практ.
23.01.2020	307	Антонов Ю.С.	Прог	Лаб.
5	307	Антонов Ю.С.	Прог	Лаб.
26.01.2020	307	Антонов Ю.С.	ПТ	Лек
2	307	Антонов Ю.С.	ПТ	Лек
27.01.2020	304	Вєтров О.С.	БДІС	Лаб.
1	306	Римар П.В.	Прог	Лаб.
2	307	Вєтров О.С.	АСД	Лек
3	306	Вєтров О.С.	АСД	Лаб.
4	304	Римар П.В.	БДІС	Лаб.
28.01.2020	304	Нескородєва Т.В.	СМП	Лаб.

Рисунок 1.1– Приклад розкладу, сформованого в системі «1С: Підприємство» [5]

Ще одним прикладом є UniTime – відкрите програмне забезпечення, яке активно використовується в університетах США [6]. Воно дозволяє створювати навчальні, екзаменаційні та подієві розклади на основі алгоритмів оптимізації. UniTime має відкритий вихідний код і гнучкий модульний інтерфейс.

Переваги:

- безкоштовне ліцензування (open-source);
- гнучкість конфігурації та налаштувань;
- інтеграція з зовнішніми інформаційними системами.

Недоліки:

- висока складність встановлення та адміністрування;
- відсутність локалізації українською мовою;
- потреба у глибоких технічних знаннях.

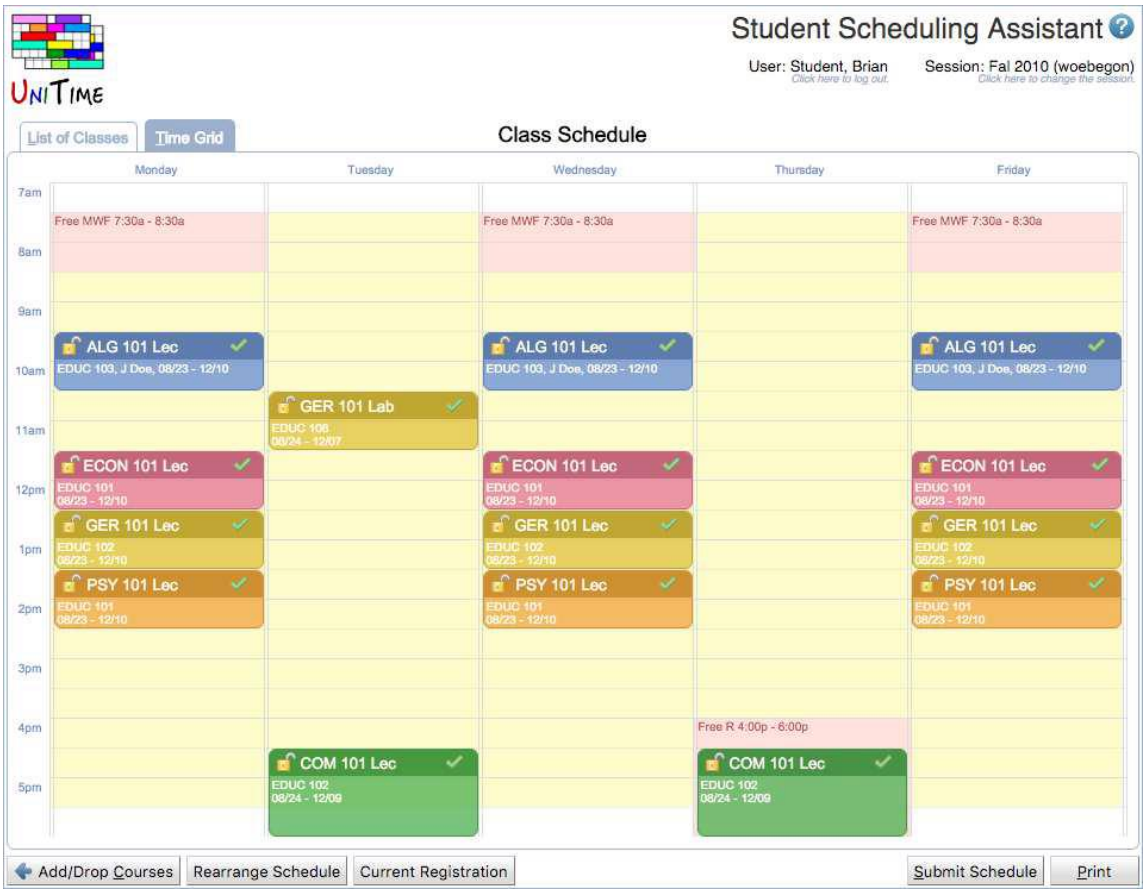


Рисунок 1.2– Інтерфейс системи UniTime [6]

У ході аналізу предметної області щодо існуючих програмних систем для складання розкладу виявлено, що більшість готових ПЗ мають надлишковий або, навпаки, обмежений функціонал, не повністю відповідають потребам Університету. Програмний застосунок для автоматизації формування розкладу занять в університеті повинен бути просторим у використанні для всіх категорій користувачів; мати зручний інтерфейс для створення, перегляду й редагування розкладу; забезпечувати підтримку веб-технологій та можливість інтеграції з календарними сервісами; мати відкриту архітектуру, що дозволить легко розширювати функціональність.

Таким чином, вивчення аналогів дало змогу сформувати концепцію програмного забезпечення, яке поєднує найкращі практики реалізації та відповідає вимогам сучасного освітнього середовища.

1.3 Постановка задачі на розробку програмного застосунку для автоматизації формування розкладу занять в університеті «Розклад занять в НУК»

На основі проведеного аналізу предметної області та аналогічних програмних систем було, було прийняте рішення розробити власне програмне забезпечення для автоматизації формування розкладу занять в університеті ”Розклад занять НУК”, що дозволить підвищити ефективність роботи диспетчерської служби університету, зменшити витрат часу на адміністрування, мінімізувати імовірність виникнення помилок при введені та обробки даних..

Вибір алгоритму для генерації розкладу є критичним кроком у розробці інформаційної системи. Варто розглянути кілька математичних алгоритмів, які можуть бути застосовані для цієї задачі.

Генетичний алгоритм натхнений принципами природного відбору та еволюції. Він працює з популяцією можливих рішень, комбінуючи та

модифікуючи їх для отримання оптимального результату [3]. Реалізовується алгоритм формулою:

$$P_{t+1} = select(mutate(crossover(P_t))) \qquad (1.1)$$

де Pt — популяція рішень на t кроці; $select$ — оператор схрещування; $mutate$ – оператор мутації; $crossover$ — оператор відбору.

Алгоритм дії генетичних алгоритмів включає кілька основних кроків. Спочатку ініціалізується початкова популяція рішень, кожне з яких оцінюється за допомогою функції пристосованості. Після цього виконуються операції схрещування та мутації для створення нових рішень. Потім відбираються найкращі рішення для формування наступної популяції. Цей процес повторюється до досягнення оптимального рішення. Генетичні алгоритми мають ряд переваг, зокрема, можливість знаходження глобально оптимальних рішень і придатність для складних задач. Однак, вони також мають недоліки, такі як високі обчислювальні витрати та необхідність налаштування параметрів для досягнення ефективних результатів.

Табу-пошук використовує метод локального пошуку з пам’яттю для уникнення локальних мінімумів. Він зберігає список заборонених (табу) рішень, щоб запобігти їх повторному відвідуванню [2,3]. Реалізовується алгоритм формулою:

$$\mathrm{S}_{\text{next}} = \arg \min_{S_F \epsilon N(S) \setminus T} f(S_F) \qquad (1.2)$$

де S_{next} – поточне рішення; $N(S)$ – сусідні рішення для S; T – табу-список заборонених рішень; $f(S_F)$ – функція оцінки рішення S_F.

Табу-алгоритм діє таким чином: спочатку ініціалізується початкове рішення. Далі обчислюються сусідні рішення, які є можливими варіантами переходу від поточного стану. З цих сусідніх рішень вибирається найкраще, яке не входить до табу-списку. Поточне рішення додається до табу-списку,

щоб уникнути повторення вже перевірених рішень. Процес повторюється до досягнення оптимального рішення. До переваг табу-алгоритму належить здатність уникати локальних мінімумів та гнучкість у налаштуванні параметрів алгоритму. Основними недоліками є високі обчислювальні витрати та складність налаштування табу-списку для ефективного функціонування алгоритму.

Жадібний алгоритм діє на основі прийняття локально оптимальних рішень на кожному кроці з метою отримання глобально оптимального результату [4]. Це один із найпростіших і найшвидших алгоритмів. Реалізовується алгоритм формулою:

$$\text{Вибір} = \arg\max_{x \epsilon X} f(x) \tag{1.3}$$

де X – множина всіх можливих виборів; $f(x)$ – функція, яка оцінює поточний вибір x.

Жадібний алгоритм діє наступним чином: спочатку визначаються всі можливі вибори для поточного кроку. Потім обирається найкращий вибір згідно з функцією оцінки. Після цього алгоритм переходить до наступного кроку і повторює процес до завершення. Перевагами жадібного алгоритму є його швидкість виконання та простота реалізації. Однак, основним недоліком є те, що він не завжди гарантує глобально оптимальне рішення, оскільки приймає локально найкращі рішення на кожному кроці.

З вище описаних алгоритмів було обрано жадібний алгоритм. Він краще підходить для реалізації поставлених задач завдяки його швидкості виконання та простоті реалізації, що є критично важливими факторами для швидкого генерування розкладів. На відміну від табу та генетичних алгоритмів, які можуть бути більш обчислювально витратними та складними в налаштуванні. Для реалізації обраного алгоритму в задачах генерації розкладу, спочатку необхідно визначити основні параметри:

- T (T) – кількість викладачів;
- D (D) – множина днів;

- P (P) – множина періодів;
- W (W) – кількість тижнів;
- L (L) – загальне навантаження;
- M (M) – максимальна кількість занять на день;
- A (A) – дозвіл на два заняття одного вчителя на день.

Для кожного заняття потрібно вибрати оптимальний слот часу та місця. Під слотом вважається місце в таблиці те буде розташований елемент викладача. Це можна зробити за допомогою функції оцінки придатності слота:

$$\text{Вибір слота} = \arg \max_{(d,p,w)\epsilon D\times P\times W} f(d,p,w) \tag{1.4}$$

де $(d,\ p,\ w)$ – день, період і тиждень; $f(d,\ p,\ w)$ – функція оцінки слота, яка визначає його придатність для заняття. Функція оцінки $f(d,\ p,\ w)$ може враховувати різні фактори, такі як наявність вільних викладачів, відсутність конфліктів з іншими заняттями, зручність часу для студентів тощо. Після вибору оптимального слота, потрібно призначити вчителя і предмет для цього заняття (t_i та s_i).

Загально алгоритм можна описати так:

Встановити початкові параметри, включаючи множину днів, періодів, кількість тижнів, загальне навантаження, максимальну кількість занять на день і дозвіл на два заняття одного викладача на день.

Вибір слота:

- для кожного заняття обчислити всі можливі слоти часу та місця;
- використати функцію $f(d,p,w)$ оцінки для визначення придатності кожного слота;
- вибрати слот з найвищою оцінкою $\arg \max_{(d,p,w)\epsilon D\times P\times W} f(d,p,w)$.

Призначення викладача і предмета:

- вибрати викладача t_i з урахуванням його доступності та відповідності предмету s_i;

- переконатися, що викладач не перевищує максимальну кількість занять на день M і дотримується обмежень щодо двох занять на день A. Повторення:
- повторювати процес для кожного заняття, поки не буде заповнено весь розклад, або поки не буде досягнуто загальне навантаження L.

Жадібний алгоритм підходить для реалізації задачі завдяки його швидкості та простоті, що дозволяє ефективно генерувати розклади з урахуванням основних обмежень і вимог.

Після аргументації вибору алгоритму визначемо основні функціональні та нефункціональні вимоги до майбутнього застосунку

Вимоги до складу виконуваних функцій:

- авторизація користувачів за ролями (адміністратор, викладач, студент);
- формування розкладу занять вручну та автоматично з урахуванням таких параметрів, як аудиторне навантаження, наявність викладачів, груп, аудиторій;
- адміністрування академічних груп;
- збереження та редагування онлайн-посилань на дистанційні заняття;
- експорт розкладу у форматах PDF/Excel;
- синхронізація з календарем Google.

Програмне забезпечення повинно мати зручний, інтуїтивно зрозумілий веб-інтерфейс для різних ролей користувачів: студентів, викладачів та адміністраторів. Інтерфейс має забезпечувати легкий доступ до функцій перегляду, створення, редагування розкладу та адміністрування груп. Підтримка багаторівневої авторизації забезпечить можливість входу до системи з урахуванням ролей користувачів: кожен має доступ лише до дозволеного функціоналу. Наприклад, адміністратори можуть редагувати розклад, а студенти лише переглядати.

Автоматичне формування розкладу буде здійснюватися шляхом реалізації алгоритму генерації розкладу на основі заданих параметрів: кількість тижнів, навантаження, допустима кількість занять на день, типи занять, побажання викладачів. Для цього використовується жадібний алгоритм як основа для побудови розкладу.

Також передбачено можливість внесення користувачами своїх побажань до розкладу, зокрема щодо часу проведення занять, типу предмету та інших факторів, що впливають на формування розкладу.

Вимоги до складу і параметрів технічних засобів:

- мінімальні ресурси клієнтської машини: 2 ГБ ОЗП;
- серверна частина: 4 ГБ ОЗП.

Вимоги до інформаційної та програмної сумісності:

- функціонування під управлінням ОС Windows 10+;
- підтримка REST API (опційно) для зовнішніх інтеграцій;
- браузер із підтримкою HTML5;
- серверна частина: підтримка Python 3.10+, СУБД PostgreSQL або SQLite;

Детальні вимоги до програмного продукту наведені у Додатку А «Технічне завдання».

Таким чином, систематизація та формалізація задач розробки дозволяє побудувати цілісну структуру майбутнього застосунку та забезпечити ефективне виконання всіх етапів дипломного проєктування.

2 РОЗРОБКА ПРОГРАМНОГО ЗАСТОСУНКУ ДЛЯ АВТОМАТИЗАЦІЇ ФОРМУВАННЯ РОЗКЛАДУ ЗАНЯТЬ В УНІВЕРСИТЕТІ «РОЗКЛАД ЗАНЯТЬ В НУК»

2.1 Аналіз вимог до програмного забезпечення

Детальний опис функціональних та нефункціональних вимог до програмного застосунку для автоматизації формування розкладу занять в університеті «Розклад занять в НУК» наведено в підрозділі 1.3 та Додатку А «Технічне завдання» кваліфікаційної роботи.

2.1.1 Побудова моделі варіантів використання програмного застосунку для автоматизації формування розкладу занять в університеті «Розклад занять в НУК»

На етапі аналізу предметної галузі та формуванні вимог було виявлено три дійові особи, які взаємодіють з програмним застосунком, а саме Студент, Викладач та Адміністратор.

В процесі аналізу вимог до програмного застосунку було визначено прецеденти для дійових осіб, які наведені нижче.

Для актора Адміністратор:

- авторизація;
- вихід з кабінету;
- генерація розкладу;
- перегляд розкладу;
- перегляд побажань;
- видалення користувачів;
- експорт розкладу.

Для актора Викладач:

- реєстрація;
- авторизація;
- перегляд профілю;
- перегляд розкладу;
- додавання побажання;
- видалення побажання;
- перегляд побажань;
- адміністрування академічних груп;
- додавання онлайн посилань.

Для актора Студент:

- реєстрація;
- авторизація;
- перегляд профілю;
- перегляд розкладу;
- додавання побажання.

На рисунку 2.1 наведена діаграма варіантів використання для розробляємого програмного застосунку.

Рисунок 2.1 - Діаграма варіантів використання програмного застосунку для автоматизації формування розкладу занять в університеті «Розклад занять в НУК»

В табл. 2.1 приведений короткий опис прецедентів ПЗ для виявлених дійових осіб.

Таблиця 2.1 – Прецеденти ПЗ та їх опис

Код	Актор	Назва	Формулювання
1	2	3	4
А1	Адміністратор	Авторизація	Дозволяє авторизуватися на сайті
А2	Адміністратор	Вихід з кабінету	Вихід з особистого кабінету
А3	Адміністратор	Генерація розкладу	Дозволяє сгенерувати розклад

Продовження таблиці 2.1

1	2	3	4
A4	Адміністратор	Перегляд розкладу	Дозволяє передивитися розклад
A5	Адміністратор	Перегляд побажань	Дозволяє продивитися побажання
A6	Адміністратор	Видалення користувачів	Дозволяє видалати користувачів всіх типів
A7	Адміністратор	Експорт розкладу	Дозволяє експортувати розклад у форматі XSLX/PDF
T1	Викладач	Реєстрація	Дозволяє зареєструватися на сайті
T2	Викладач	Авторизація	Дозволяє авторизуватися на сайті
T3	Викладач	Перегляд профілю	Дозволяє переглянути профіль викладача
T4	Викладач	Перегляд розкладу	Дозволяє переглянути розклад занять
T5	Викладач	Додавання побажання	Дозволяє написати побажання
T6	Викладач	Видалення побажання	Дозволяє видалити побажання
T7	Викладач	Перегляд побажань	Дозволяє переглянути побажання
T8	Викладач	Адміністрування академічних груп	Дозволяє видаляти та створювати академічні групи
T9	Викладач	Додавання онлайн посилань	Дозволяє додавати онлайн посилання у розклад
C1	Студент	Реєстрація	Дозволяє зареєструватися на сайті
C2	Студент	Авторизація	Дозволяє авторизуватися на сайті
C3	Студент	Перегляд профілю	Дозволяє переглянути профіль студента
C4	Студент	Перегляд розкладу	Дозволяє переглянути розклад занять
C5	Студент	Додавання побажання	Дозволяє написати побажання

2.1.2 Специфікації варіантів використання

Для конкретизації виявлених варіантів використання (див. табл. 2.1) було розроблені їх специфікації.

А3 – Генерація розкладу

Основна дійова особа: Адміністратор

Інші учасники прецеденту: Відсутні

Передумови: Адміністратор має бути авторизованим у системі

Короткий опис:

Цей варіант використання дозволяє адміністратору автоматично згенерувати розклад занять для всіх академічних груп на основі введених даних і побажань користувачів.

А7 – Експорт розкладу

Основна дійова особа: Адміністратор

Інші учасники прецеденту: Відсутні

Передумови: Генерація розкладу має бути завершена

Короткий опис:

Цей варіант використання дозволяє адміністратору експортувати згенерований розклад у файл формату XLSX або PDF для подальшого використання або розповсюдження.

Т5 – Додавання побажання

Основна дійова особа: Викладач

Інші учасники прецеденту: Відсутні

Передумови: Користувач має бути авторизованим і мати доступ до особистого кабінету

Короткий опис:

Даний варіант використання дозволяє викладачу вказати свої побажання щодо часу проведення занять, аудиторій або інших умов для формування розкладу.

Т9 – Додавання онлайн-посилань

Основна дійова особа: Викладач

Інші учасники прецеденту: Студенти (опосередковано)

Передумови: Має бути сформований розклад або створений предмет

Короткий опис:

Дозволяє викладачу додавати онлайн-посилання (наприклад, Zoom, Meet) до занять у розкладі, що полегшує організацію дистанційного навчання.

С5 – Додавання побажання

Основна дійова особа: Студент

Інші учасники прецеденту: Відсутні

Передумови: Студент має бути зареєстрованим і авторизованим

Короткий опис:

Цей варіант дозволяє студенту внести побажання щодо розкладу занять, зокрема бажані вікна, заняття онлайн чи офлайн, або інші параметри.

2.2 Розробка архітектури програмного застосунку для автоматизації формування розкладу занять в університеті «Розклад занять в НУК»

Архітектура програмного застосунку складається з трьох основних компонентів: клієнтська частина, серверна частина та база даних. Кожен з цих компонентів має свої підсистеми та модулі, які забезпечують функціонування всієї програмної системи. Для візуалізації архітектури була використана діаграма компонентів, яка демонструє взаємодію між клієнтською частиною, серверною частиною та базою даних (рис. 2.2).

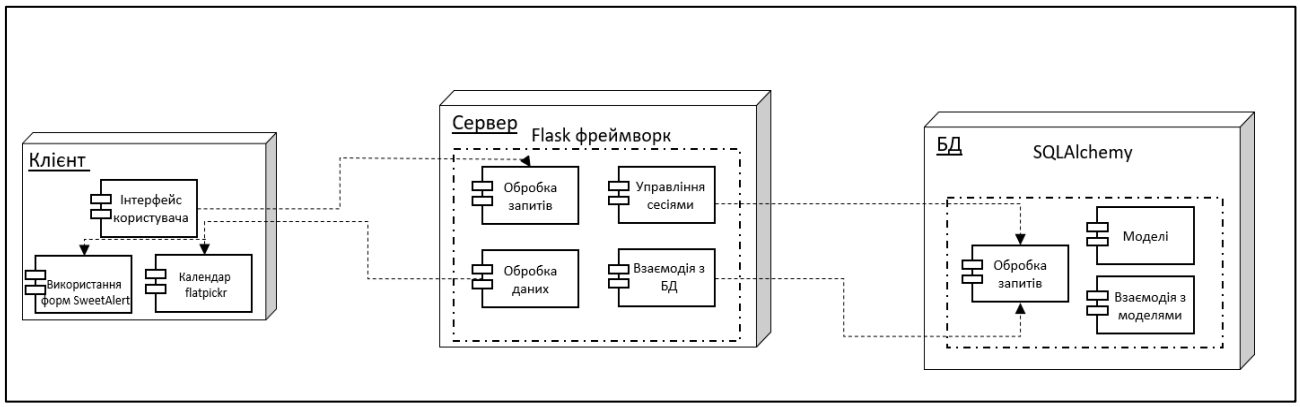


Рисунок 2.2 — Діаграма компонентів ПЗ

Клієнтська частина включає в себе наступні компоненти:

1 Інтерфейс користувача:

- використовує HTML, CSS та JavaScript для створення зручного та інтуїтивно зрозумілого інтерфейсу;
- взаємодіє з користувачами через браузер, забезпечуючи введення та отримання даних;

2 Використання форм SweetAlert:

- SweetAlert є бібліотекою для створення красивих та інтерактивних попереджень та повідомлень [9];
- використовується для відображення діалогових вікон, підтверджень дій, попереджень про помилки та успішні операції;

3 Календар flatpickr:

- Flatpickr є легкою бібліотекою для створення зручних календарів та вибору дат [10];
- забезпечує можливість вибору дат і часу, що особливо корисно для планування та генерації розкладів.

Серверна частина реалізована за допомогою фреймворку та включає наступні компоненти:

1 Flask фреймворк:

- Flask є мікрофреймворком для Python, який дозволяє швидко

створювати веб-додатки [11];

- забезпечує обробку HTTP-запитів, маршрутизацію, управління сесіями та взаємодію з шаблонами;

2 Обробка запитів:

- Flask обробляє запити від клієнтської частини, викликає відповідні функції обробки та формує відповіді;
- включає маршрутизацію, яка дозволяє спрямовувати запити на відповідні обробники;

3 Обробка даних:

- серверна частина обробляє дані, виконуючи необхідні операції, такі як валідація, обробка та підготовка даних для бази даних або клієнта;

4 Управління сесіями:

- Flask забезпечує автентифікацію та авторизацію користувачів;
- управління сесіями дозволяє відстежувати активні сеанси користувачів та зберігати їх стан;

5 Взаємодія з БД:

- Flask взаємодіє з базою даних та забезпечує зручний доступ до даних та управління ними.

База даних включає наступні компоненти:

1 SQLAlchemy є бібліотекою ORM, яка дозволяє працювати з базою даних на рівні об’єктів та забезпечує створення моделей, визначення зв’язків між ними та виконання CRUD-операцій (створення, читання, оновлення, видалення);

2 Моделі:

- моделі визначають структуру даних, що зберігаються в базі даних, та зв’язки між ними;
- моделі використовуються для зручного доступу до даних з рівня додатка;

3 Ввзаємодія з моделями:

- SQLAlchemy забезпечує створення, читання, оновлення та видалення даних у базі даних;
- дозволяє здійснювати складні запити та операції з даними.

Ця архітектура забезпечує високу гнучкість, масштабованість та легкість в підтримці системи. Кожен з компонентів може бути легко оновлений або замінений без значного впливу на інші частини системи, що робить систему надійною та ефективною.

2.3 Проєкт програмного забезпечення

2.3.1 Ескізний проєкт програмного забезпечення

2.3.1.1 Розробка діаграм діяльності прецедентів програмного забезпечення

Для формалізації представлення варіантів використання (див. рис. 2.1) в кваліфікаційній роботі були розроблені діаграми діяльності. На рисунках 2.3 – 2.4 представлені діаграми діяльності для деяких прецедентів програмного застосунку для автоматизації формування розкладу занять в університеті «Розклад занять в НУК».

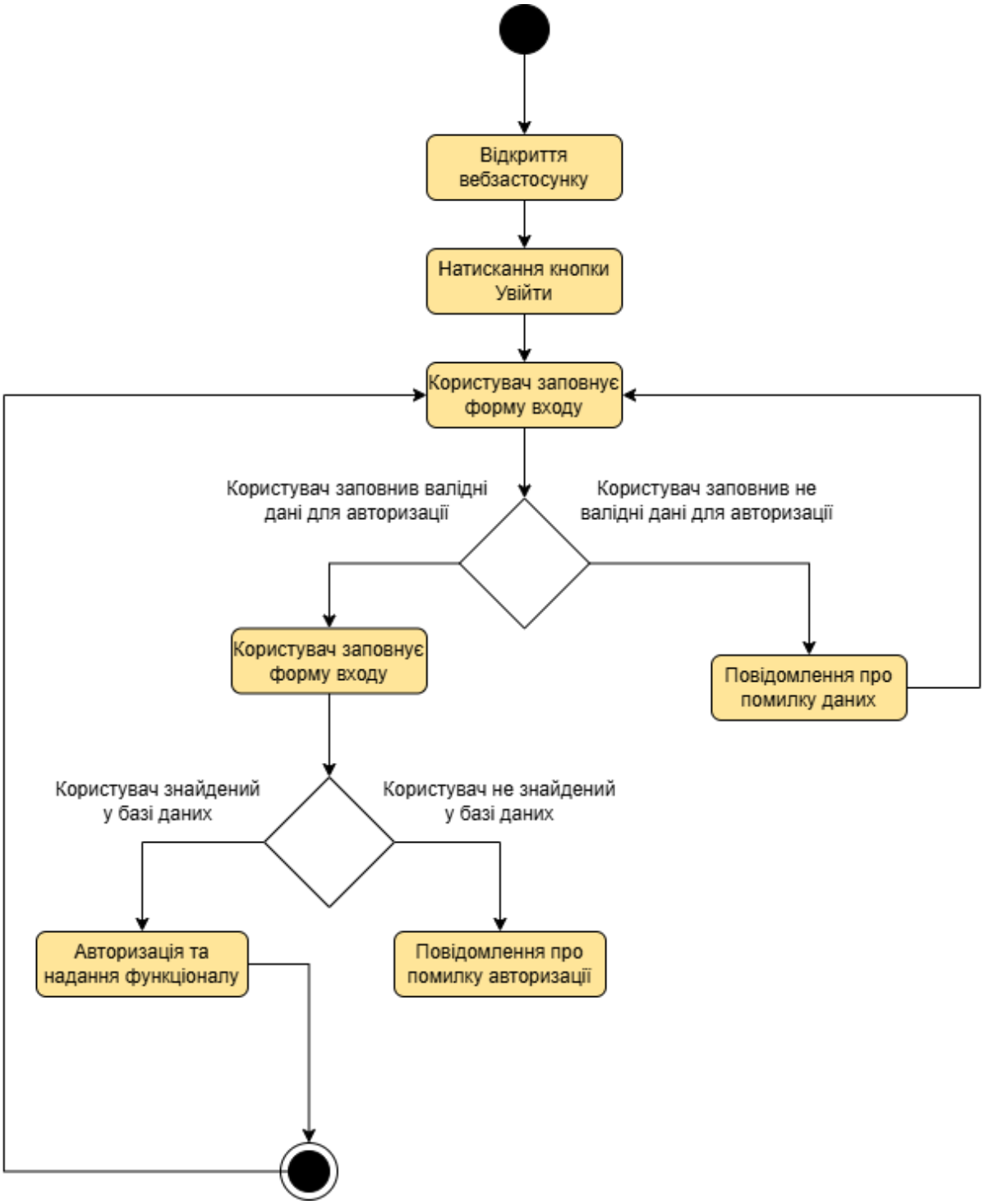


Рисунок 2.4 – Діаграма діяльності варіанту використання «Авторизація»

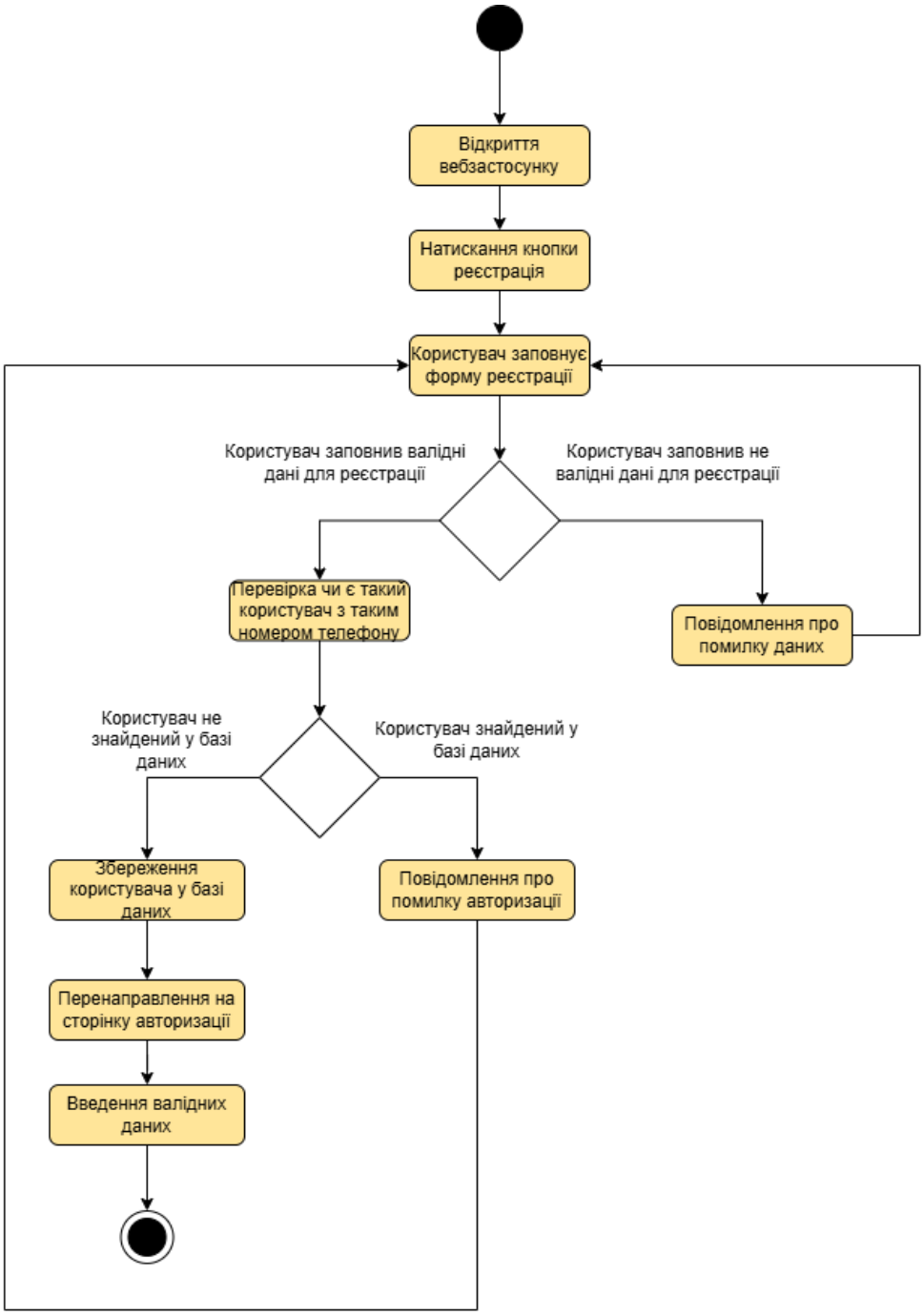


Рисунок 2.5 – Діаграма діяльності прецеденту “Реєстрація”

2.3.1.2 Проєкт користувацького інтерфейсу програмного забезпечення

Інтерфейс користувача уявляє собою набір елементів для обробки та відображення інформації, призначених для реалізації зручної та ергономічної роботи користувача з програмним забезпеченням.

На основі аналізу вимог до розробляємого ПЗ на етапі ескізного проєктування були розроблені ескізи екранних форм програмного застосунку для автоматизації формування розкладу занять в університеті «Розклад занять в НУК», які наведени на рисунках 2.6 –2. 7.

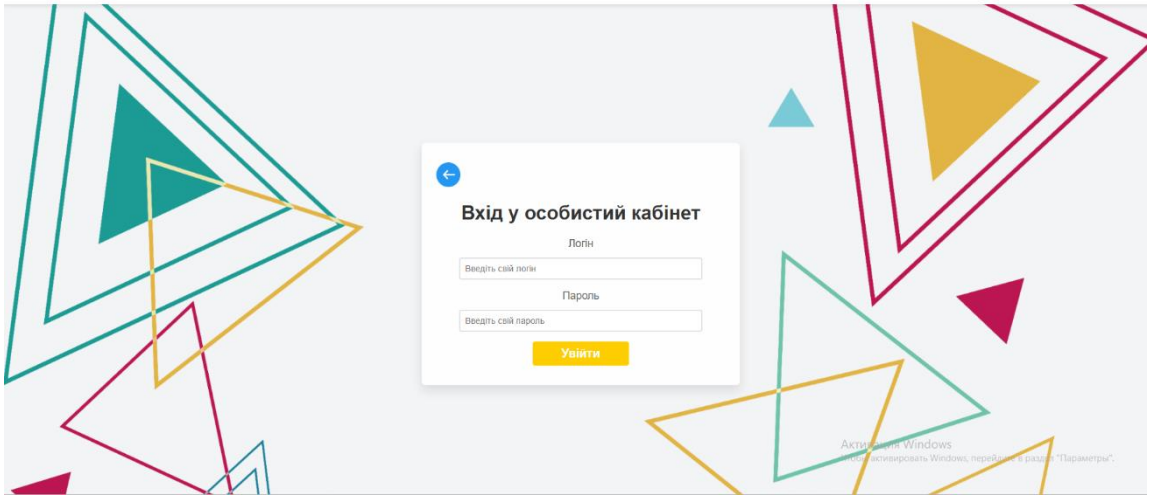


Рисунок 2.6 – Ескіз інтерфейсу сторінка ВХІД

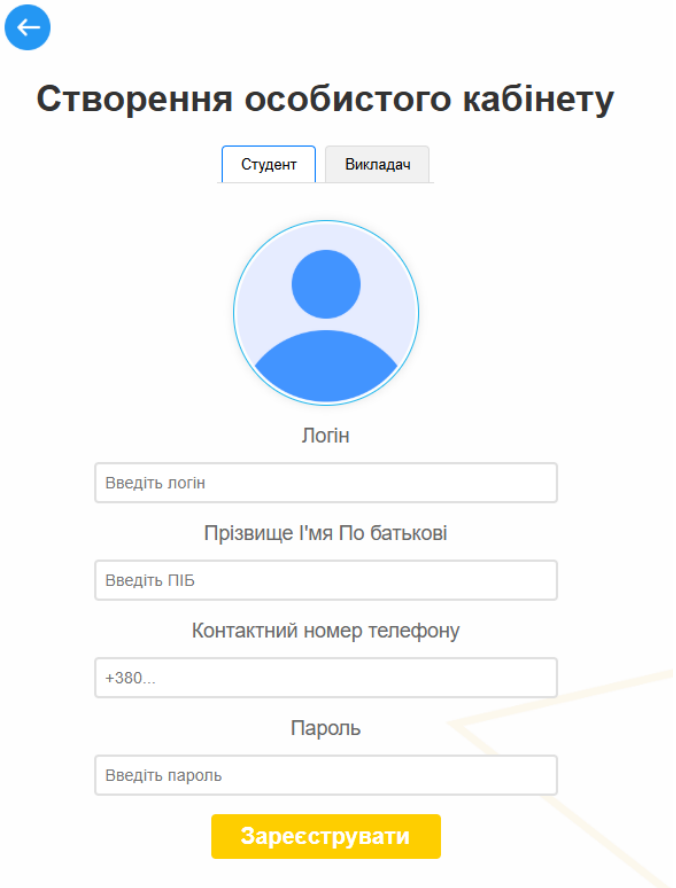


Рисунок 2.7 – Ескіз інтерфейсу сторінка РЕЄСТРАЦІЯ

2.3.2 Технічний проєкт програмного забезпечення

За результатами ескізного проєкту наступним кроком виконання кваліфікаційної роботи було розроблено технічний проєкт ПЗ. Результатом цього етапу було розроблено статичну та динамічну модель програмного застосунку для автоматизації формування розкладу занять в університеті «Розклад занять в НУК».

2.3.2.1 Структура бази даних ПЗ

База даних системи складається з чотирьох основних таблиць: User, Group, Event та Schedule (рис. 2.2). Кожна з цих таблиць має свої поля та взаємозв’язки між собою, що забезпечує ефективне зберігання та доступ до інформації.

Таблиця User зберігає інформацію про користувачів системи. Вона включає наступні поля:

- id: унікальний ідентифікатор користувача (ціле число, первинний ключ);
- username: логін користувача (рядок, унікальний, обов’язковий);
- real_name: справжнє ім’я користувача (рядок, обов’язковий);
- phone: номер телефону користувача (рядок, обов’язковий);
- password: пароль користувача (рядок, обов’язковий);
- profile_photo: шлях до фото профілю користувача (рядок, не обов’язковий, за замовчуванням «user.png»);
- role: роль користувача в системі (рядок, обов’язковий, за замовчуванням «user»);
- position: посада користувача (рядок, не обов’язковий);
- group_id: ідентифікатор групи, до якої належить користувач (ціле

число, зовнішній ключ до таблиці Group);

- managed_groups: взаємозв’язок з таблицею Group для визначення груп, якими керує користувач.

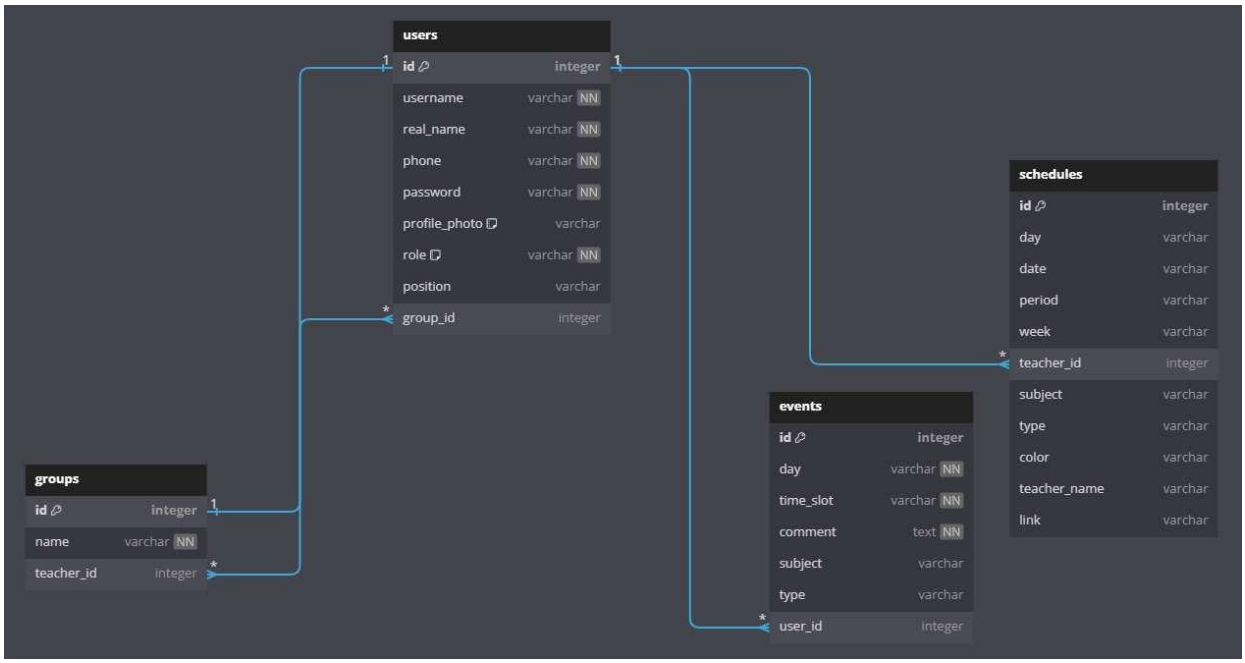


Рисунок 2.8 — Структура бази даних

Таблиця Group зберігає інформацію про групи студентів. Вона включає наступні поля:

- id: унікальний ідентифікатор групи (ціле число, первинний ключ);
- name: назва групи (рядок, обов’язковий);
- teacher_id: ідентифікатор викладача, який керує групою (ціле число, зовнішній ключ до таблиці User);
- users: взаємозв’язок з таблицею User для визначення студентів, що належать до групи.

Таблиця Event зберігає інформацію про побажання користувачів стосовно розкладу. Вона включає наступні поля:

- id: унікальний ідентифікатор події (ціле число, первинний ключ);
- day: день побажання(рядок, обов’язковий);

- time_slot: часовий слот побажання (рядок, обов’язковий);
- comment: коментар до побажання (текст, обов’язковий);
- subject: предмет побажання (рядок, не обов’язковий);
- type: тип побажання (рядок, не обов’язковий);
- user_id: ідентифікатор користувача, що створив подію (ціле число, зовнішній ключ до таблиці User);
- user: взаємозв’язок з таблицею User для визначення користувача, що створив подію.

Таблиця Schedule зберігає інформацію про розклад занять. Вона включає наступні поля:

- id: унікальний ідентифікатор розкладу (ціле число, первинний ключ);
- day: день тижня (рядок, не обов’язковий);
- date: дата заняття (рядок, не обов’язковий);
- period: період заняття (рядок, не обов’язковий);
- week: номер тижня (рядок, не обов’язковий);
- teacher_id: ідентифікатор викладача (ціле число, не обов’язковий);
- subject: назва предмета (рядок, не обов’язковий);
- type: тип заняття (рядок, не обов’язковий);
- color: колір для позначення заняття (рядок, не обов’язковий);
- teacher_name: ім’я викладача (рядок, не обов’язковий);
- link: посилання на онлайн-заняття (рядок, не обов’язковий);

Взаємозв’язки між таблицями:

- User - Group: взаємозв’язок «багато до одного» між таблицями User і Group, де кожен користувач може належати до однієї групи, а кожна група може мати багато користувачів.
- User - Group (керівництво): Взаємозв’язок «один до багатьох» між таблицями User і Group, де кожен викладач може керувати кількома групами.
- Event - User: Взаємозв’язок «багато до одного» між таблицями

Event і User, де кожна подія прив’язана до одного користувача.

- Schedule - User: Взаємозв’язок «багато до одного» між таблицями Schedule і User, де кожне заняття прив’язане до одного викладача.

Ця структура бази даних забезпечує ефективне зберігання та доступ до даних про користувачів, групи, події та розклад, дозволяючи реалізувати всі необхідні функції системи.

2.3.2.2 Динамічна модель програмного забезпечення

Діаграма послідовності є різновидом діаграм UML. Вона відображає взаємодію об’єктів, впорядковану за часом. Зокрема, такі діаграми демонструють залучені об’єкти та послідовність переданих між ними повідомлень.

Діаграми послідовності для варіанту використання «Авторизація» наведено на рисунку 2.9 .

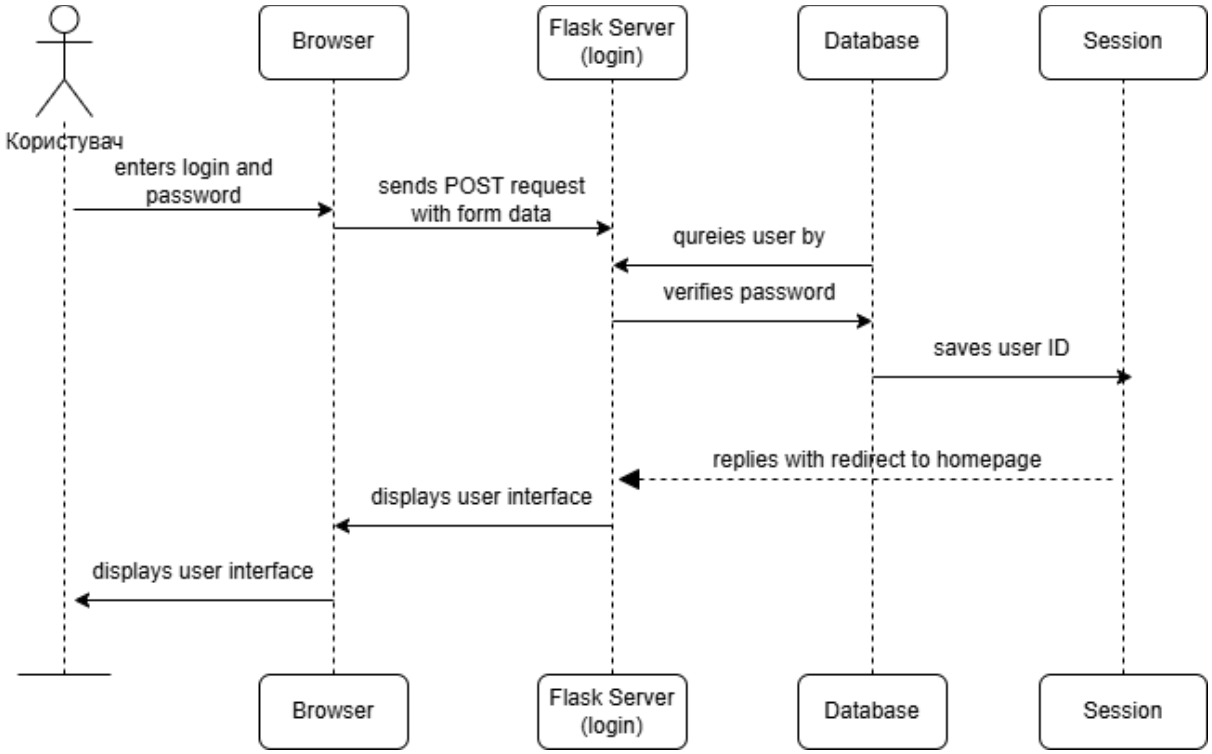


Рисунок 2.9 — Динамічна модель варіантів використання ”Авторизація”

2.4 Робочий проєкт. Реалізація програмного застосунку для автоматизації формування розкладу занять в університеті «Розклад занять в НУК»

2.4.1 Вибір та обґрунтування засобів розробки програмного забезпечення

Для успішної реалізації програмного застосунку обрано набір інструментів, що включає мови фронтенду та бекенду, а також середовище розробки.

Для фронтенд-розробки було обрано HTML, CSS та JavaScript. Ці мови забезпечують створення інтуїтивно зрозумілого та інтерактивного користувацького інтерфейсу. HTML основна мова розмітки веб-сторінок, яка використовується для створення структури сторінки. Вона дозволяє визначати елементи інтерфейсу, такі як форми, кнопки, таблиці та інші компоненти. CSS мова стилізації, що використовується для візуального оформлення HTML-елементів. CSS дозволяє задавати кольори, шрифти, розміри, відступи та інші стилі, що робить інтерфейс привабливим та зручним для користувачів. JavaScript мова програмування, що використовується для створення динамічних та інтерактивних веб-сторінок. JavaScript дозволяє реалізувати такі функції, як валідація форм, обробка подій, асинхронні запити до сервера та інше.

Для бекенд-розробки було обрано Python з використанням фреймворку Flask. Цей вибір обумовлений гнучкістю та зручністю Python, а також простотою та масштабованістю Flask. Python високорівнева мова програмування, що відрізняється простотою синтаксису та широкими можливостями. Вона підтримує об'єктно-орієнтоване, процедурне та функціональне програмування, що робить її універсальним інструментом для різних задач. Flask легкий веб-фреймворк для Python, який дозволяє швидко створювати веб-додатки. Flask надає необхідний набір інструментів для обробки HTTP-запитів, керування сесіями, роботи з шаблонами та іншими

аспектами веб-розробки. Завдяки своїй мінімалістичній архітектурі, Flask легко інтегрується з іншими бібліотеками та розширеннями [7, 8].

Для розробки системи було обрано Visual Studio Code (VS Code) [9] популярне інтегроване середовище розробки (IDE), що забезпечує зручність роботи та багатофункціональність. VS Code безкоштовне та кросплатформене середовище розробки, що підтримує широкий спектр мов програмування та розширень. VS Code надає потужні інструменти для редагування коду, налагодження, управління версіями, інтеграції з системами контролю версій (Git) та інші корисні функції. Завдяки великій кількості розширень, користувачі можуть налаштувати VS Code під свої потреби, додавати підтримку нових мов програмування, інструменти для тестування, налагодження та багато іншого [9].

Таким чином, вибір інструментів для реалізації системи був зроблений на користь надійних та перевірених технологій, що забезпечують високу продуктивність, гнучкість та зручність у використанні. Використання HTML, CSS та JavaScript для фронтенду, Python та Flask для бекенду, а також Visual Studio Code як середовище розробки, дозволяє створити потужну та ефективну систему автоматичної генерації розкладу занять.

2.4.2 Опис та реалізація ключових функцій

Авторизація.

Реєстрація та авторизація користувачів є ключовими функціями системи, що забезпечують контроль доступу до різних частин веб-застосунку. Цей процес включає в себе перевірку даних, збереження нових користувачів та автентифікацію існуючих користувачів.

Функція реєстрації обробляє запити на створення нових користувачів. Вона перевіряє коректність введених даних, унікальність логіна та номеру телефону, завантажує профільне фото, якщо воно було надане, та створює новий об’єкт користувача в базі даних.

На рис 2.10 представлено лістинг програмного коду для обробки запитів на реєстрацію користувачів. Він перевіряє, чи користувач з таким же логіном або номером телефону вже існує в базі даних. Якщо знайдено дублікати, користувачеві відображається відповідне повідомлення про помилку.

```
@main.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        username = request.form['username']
        realname = request.form['realname']
        phone = request.form['phone']
        password = request.form['password']
        position = request.form.get('position')
        existing_user = User.query.filter_by(username=username).first()
        existing_phone = User.query.filter_by(phone=phone).first()
        if existing_user:
            flash('Користувач з таким логіном вже існує. Спробуйте інший!', 'error')
            return render_template('register.html')
        if existing_phone:
            flash('Користувач з таким номером телефону вже існує.', 'error')
            return render_template('register.html')
```

Рисунок 2.10 – Лістинг програмного коду для обробки запитів на реєстрацію користувачів

На рис 2.11 представлено лістинг програмного коду для перевірки правильності формату телефонного номера та обробки завантаження профільного фото. Якщо фото не було завантажено, використовується зображення за замовчуванням. Також перевіряється довжина логіна, пароля та ПІБ, а також заповненість необхідних полів.

```
if not (phone.startswith('+') and len(phone) == 13 and phone[1:].isdigit()):
    flash('Номер телефону повинен починатися з +, містити тільки цифри та мати довжину 12 символів без +.', 'error')
    return render_template('register.html')
if 'profile_photo' in request.files and request.files['profile_photo'].filename != '':
    profile_photo = save_profile_image(request.files['profile_photo'])
else:
    profile_photo = 'user.png'
if len(username) < 8 or len(password) < 8 or len(realname) < 8:
    flash('Поля Логін, Пароль та ПІБ мають містити не менше 8 символів!', 'error')
    return render_template('register.html')
if not username or not password or not realname or not phone:
    flash('Заповніть всі необхідні поля!', 'error')
    return render_template('register.html')
```

Рисунок 2.11 – Лістинг програмного коду для перевірки правильності формату телефонного номера та обробки завантаження профільного фото

Наступний фрагмент (рис.2.12) визначає роль користувача (адміністратор, викладач або звичайний користувач) і створює новий об'єкт

користувача, який зберігається у базі даних. Після успішної реєстрації користувача система переадресовує його на відповідну домашню сторінку.

```
        if username == "admin_admin" and password == "admin_admin":
            role = 'admin'
        else:
            role = 'teacher' if position else 'user'
        new_user = User(
            username=username,
            real_name=realname,
            phone=phone,
            password=password,
            profile_photo=profile_photo,
            role=role,
            position=position
        )
        db.session.add(new_user)
        db.session.commit()
        flash('Реєстрація пройшла успішно!', 'success')
        session.clear()
        session['user_id'] = new_user.id
        if new_user.role == 'admin':
            return redirect(url_for('user.admin_home'))
        else:
            return redirect(url_for('user.user_home'))
    return render_template('register.html')
```

Рисунок 2.12 – Лістинг програмного коду для створення нового об'єкту користувача

Функція авторизації обробляє запити на вхід користувачів у систему. Вона перевіряє правильність введених логіна і пароля, автентифікує користувача і перенаправляє його на відповідну домашню сторінку (рис. 2.13).

```python
@main.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        if len(username) < 8 or len(password) < 8:
            flash('Поля Логін та Пароль мають містити не менше 8 символів!', 'error')
            return render_template('login.html')
        user = User.query.filter_by(username=username, password=password).first()
        if not user:
            flash('Помилка входу! Перевірте дані та спробуйте ще раз.', 'error')
            return render_template('login.html')
        user_login(user)
        flash('Вхід успішний!', 'success')
        if user.role == 'admin':
            return redirect(url_for('user.admin_home'))
        else:
            return redirect(url_for('user.user_home'))
    return render_template('login.html')
```

Рисунок 2.13 – Лістинг програмного коду функції авторизації

Представлений на рис. 2.13 фрагмент коду обробляє запити на авторизацію користувачів. Він перевіряє, чи логін і пароль містять не менше 8 символів, а також перевіряє наявність користувача з такими даними в базі. Якщо користувача знайдено, виконується його вхід у систему та перенаправлення на відповідну домашню сторінку залежно від ролі.

Маршрути для домашніх сторінок відображають відповідні інтерфейси для користувачів та адміністраторів після успішного входу. На рис. 2.14 представлено лістинг програмного коду, що відображає домашню сторінку користувача після успішного входу. Він перевіряє автентифікацію поточного користувача та передає його дані для відображення у відповідному шаблоні:

```python
@user.route('/user_home')
@login_required
def user_home():
    if current_user.is_authenticated:
        profile_photo = current_user.profile_photo
        return render_template('user_home.html', user=current_user, profile_photo=profile_photo)
    else:
        return "Користувача не знайдено, будь-ласка увійдіть ще раз!", 404
```

Рисунок 2.14 – Лістинг програмного , що відображає домашню сторінку користувача після успішного входу

На рис. 2.15 представлено лістинг програмного коду, що відображає відображає домашню сторінку адміністратора після успішного входу. Він перевіряє автентифікацію поточного користувача та передає його дані для відображення у відповідному шаблоні.

```python
@user.route('/admin_home')
@login_required
def admin_home():
    if current_user.is_authenticated:
        profile_photo = current_user.profile_photo
        return render_template('admin_home.html', user=current_user, profile_photo=profile_photo)
    else:
        return "Користувача не знайдено, будь-ласка увійдіть ще раз!", 404
```

Рисунок 2.15 – Лістинг програмного , що відображає домашню сторінку адміністратора після успішного входу

Адміністрування користувачів.

Включає в себе управління списком користувачів, пошук по користувачах, а також видалення користувачів. Цей процес забезпечує зручний інтерфейс для адміністраторів, що дозволяє легко виконувати необхідні операції з даними користувачів. HTML- код забезпечує структуру таблиці для відображення списку користувачів, а також поле для пошуку користувачів. Лістинг блоку коду (рис. 2.16) створює HTML-структуру для

відображення користувачів. Він містить заголовок, поле для пошуку та таблицю з колонками для логіну, реального імені, ролі, контактного телефону, групи або кураторства, посади та дій.

```
<div id="users" class="content">
    <h1>Адміністрування користувачів системи</h1>
    <input type="text" id="search-input" placeholder="Пошук користувачів..." oninput="fetchUsers()">
    <table class="group-table">
        <thead>
            <tr>
                <th>Логін</th>
                <th>Реальне ім'я</th>
                <th>Роль</th>
                <th>Контактний телефон</th>
                <th>Група\Куратор групи</th>
                <th>Посада</th>
                <th>Дія</th>
            </tr>
        </thead>
        <tbody id="users-body"></tbody>
    </table>
</div>
```

Рисунок 2.16 – Лістинг програмного , що коду створює HTML-структуру для відображення користувачів

Функція fetchUsers (рис. 2.17) викликається при введенні тексту в поле пошуку і надсилає запит до сервера для отримання списку користувачів:

```
function fetchUsers() {
    const searchInput = document.getElementById('search-input');
    const usersBody = document.getElementById('users-body');

    if (!searchInput || !usersBody) {
        console.error('Required elements not found in the DOM');
        return;
    }
    const searchQuery = searchInput.value;
    fetch(`/get_users?search=${searchQuery}`)
        .then(response => response.json())
        .then(data => {
            usersBody.innerHTML = '';
            data.users.forEach(user => {
                const userRow = document.createElement('tr');
                userRow.innerHTML = `
```

Рисунок 2.17 – Лістинг програмного функції fetchUsers

Ця функція отримує введений користувачем пошуковий запит, надсилає

його до сервера та відображає результати у таблиці. Для кожного користувача створюється новий рядок у таблиці з відповідними даними.

Алгоритм генерації розкладу

При натисканні кнопки «Згенерувати розклад» запускається процес збирання даних та генерації розкладу. Спочатку здійснюється зчитування параметрів користувача, таких як загальне навантаження, максимальна кількість занять на день, тип розкладу та дозволи на два заняття одного вчителя на день.

```
document.getElementById('generate_schedule').addEventListener('click', function() {
    const totalLoad = parseInt(document.getElementById('total_load').value);
    const maxClasses = parseInt(document.getElementById('max_classes').value);
    const scheduleType = document.querySelector('input[name="schedule_type"]:checked').value;
    const scheduleDuration = 4;
    const allowTwoClasses = document.querySelector('input[name="allow_two_classes"]:checked').value === 'yes';
```

Рисунок 2.18 – Лістинг програмного коду для зчитування параметрів користувача

Після перевірки введених значень, здійснюється запит до сервера для отримання даних про викладачів, які зберігаються у змінній teachers. Ці дані включають ID викладача, ім’я, предмет та колір, призначений для візуалізації.

Після отримання даних про викладачів здійснюється ініціалізація таблиці розкладу з часовими слотами та днями тижня. Кожна клітинка таблиці отримує атрибути data-day, data-period та data-week, що визначають день, період та тиждень відповідно:

```
function generateSchedule(totalLoad, maxClasses, scheduleType, scheduleDuration, teachers, allowTwoClasses) {
    const timeSlots = [
        '1 пара: 8:30 - 9:50',
        '2 пара: 10:10 - 11:30',
        '3 пара: 12:00 - 13:20',
        '4 пара: 13:40 - 15:00',
        '5 пара: 15:20 - 16:40',
        '6 пара: 17:00 - 18:20',
        '7 пара: 18:40 - 20:00'
    ];
    const days = ['Понеділок', 'Вівторок', 'Середа', 'Четвер', 'П\'ятниця'];
    const weekBodies = [
        document.getElementById('week-1-body'),
        document.getElementById('week-2-body'),
        document.getElementById('week-3-body'),
        document.getElementById('week-4-body')
    ];
    weekBodies.forEach(weekBody => {
        weekBody.innerHTML = '';
        for (let i = 0; i < timeSlots.length; i++) {
            const row = document.createElement('tr');
            const timeCell = document.createElement('td');
            timeCell.innerText = timeSlots[i];
            row.appendChild(timeCell);
            for (const day of days) {
                const cell = document.createElement('td');
                cell.classList.add('droppable');
                cell.dataset.day = day;
                cell.dataset.period = i + 1;
                cell.dataset.week = weekBodies.indexOf(weekBody) + 1;
                row.appendChild(cell);
            }
            weekBody.appendChild(row);
        }
    });
```

Рисунок 2.19 – Лістинг програмного коду для ініціалізації таблиці розкладу з часовими слотами та днями тижня

Основний цикл розподілу занять по слотам розкладу. Перевіряється наявність вільних слотів, заповнюються вони заняттями, та здійснюється їх відображення на сторінці. Частина коду (рис. 2.20) відповідає за основний процес заповнення розкладу. Вона перевіряє наявність вільних слотів, заповнює їх заняттями та відображає результати на сторінці.

```
            const teacherIndex = filledSlots % teachers.length;
            const teacher = teachers[teacherIndex];
            const type = types[filledSlots % types.length];
            if (!occupiedSlots[week]) {
                occupiedSlots[week] = {};}
            if (!occupiedSlots[week][day]) {
                occupiedSlots[week][day] = {};}
            if (!occupiedSlots[week][day][teacher.name]) {
                occupiedSlots[week][day][teacher.name] = new Set();}
            const teacherDailyCount = [...occupiedSlots[week][day][teacher.name]].filter(item => item.includes(teacher.subject)).length;
            if (allowTwoClasses || (!allowTwoClasses && teacherDailyCount < 1)) {
                const scheduleItem = document.createElement('div');
                scheduleItem.classList.add('schedule-item');
                scheduleItem.style.borderColor = teacher.color;
                console.log(`Setting data-teacher-id: ${teacher.id}`); // Перевірка
                scheduleItem.setAttribute('data-teacher-id', teacher.id);
                scheduleItem.setAttribute('data-subject', teacher.subject);
                scheduleItem.setAttribute('data-type', type);
                scheduleItem.innerHTML = `<div>${teacher.name}</div><div>${teacher.subject}</div><div>${type}</div>`;
                cell.appendChild(scheduleItem);
                filledSlots++;
                dailySlots[week][day]++;
                occupiedSlots[week][day][teacher.name].add(`${teacher.subject}:${type}`);
            } }}
        slotIndex++;
        iterations++;
        if (slotIndex >= totalSlots) {
            slotIndex = 0;}}
    initDragAndDrop();
}
```

Рисунок 2.20 – Лістинг програмного коду для перевірки наявності вільних слоті та заповнення їх заняттями

Функціонал викладача

Функціонал викладача щодо управління групами студентів. Викладач може створювати нові групи, додавати студентів до груп, видаляти студентів з груп, а також видаляти самі групи. Крім того, викладач може переглядати існуючі групи та керувати ними. Код починається з функцій для відображення поточного часу та дати на сторінці, а також для відкриття календаря при кліку (див. рис. 2.21).

```
function updateDateTime() {
    const now = new Date();
    const options = { weekday: 'long', year: 'numeric', month: 'long', day: 'numeric' };
    const dateString = now.toLocaleDateString('uk-UA', options);
    const timeString = now.toLocaleTimeString('uk-UA');
    document.getElementById('current-datetime').innerText = `${dateString}, ${timeString}`;
}
```

Рисунок 2.21 – Лістинг програмного коду функції для відображення поточного часу та дати на сторінці

На основі ролі користувача (userRole), функція showContent керує відображенням відповідного контенту. Функція createGroup (рис. 2.22) відповідає за створення нової групи. Вона використовує бібліотеку SweetAlert для відображення модального вікна, де викладач може ввести назву групи та додати студентів до неї:

```
function createGroup() {
    Swal.fire({
        title: 'Створити нову групу',
        html: `
            <input type="text" id="group-name" class="swal2-input" placeholder="Назва групи">
            <p>Додайте студентів до групи:</p>
            <div id="users-list"></div>
            <button id="load-users" class="button" onclick="loadUsers()">Завантажити користувачів</button>
        `,
        showCancelButton: true,
        confirmButtonColor: '#9a00fd',
        cancelButtonColor: '#2e4957',
        confirmButtonText: 'Створити',
        cancelButtonText: 'Скасувати',
        preConfirm: () => {
            const groupName = Swal.getPopup().querySelector('#group-name').value.trim();
            const selectedUsers = Array.from(document.querySelectorAll('.user-checkbox:checked')).map(cb => cb.value);
            if (!groupName) {
                Swal.showValidationMessage('Назва групи не може бути пустою');
                return false;
            }
            if (selectedUsers.length === 0) {
                Swal.showValidationMessage('Додайте хоча б одного студента до групи');
                return false;
            }
            return { groupName: groupName, users: selectedUsers };
        }
    }).then((result) => {
        if (result.isConfirmed) {
            saveGroup(result.value);
        }
    });
}
```

Рисунок 2.22 – Лістинг програмного коду функції createGroup, яка відповідає за створення нової групи

Повний текст програмних модулів наведено у Додатку Б до кваліфікаційної роботи.

2.4.3 Тестування та випробування програмного застосунку для автоматизації формування розкладу занять в університеті «Розклад занять в НУК»

Тестування є важливими етапом в процесі розробки програмної системи та перевірки її працездатності. В кваліфікаційні роботі було проведено

функціональне (ручне) тестування програмного застосунку для автоматизації формування розкладу занять в університеті «Розклад занять НУК». Функціональне тестування, метою якого є виявити, чи правильно система виконує свої функції з погляду користувача, дає змогу перевірити чи відповідає функціональність ПЗ заявленим вимогам, описаним у технічному завданні. Це дозволить детально перевірити всі функції та елементи інтерфейсу системи. Ручне тестування є ефективним способом виявлення можливих проблем та оцінки зручності використання системи з точки зору кінцевого користувача. Такий підхід дозволяє швидко реагувати на зміни та адаптувати тестові сценарії під нові вимоги. Для перевірки коректності роботи ПЗ був розроблений план тестування та чек-лист з переліком тестових випадків (табл. 2.2).

Таблиця 2.2 – Чек-лист тестування програмного застосунку

№	Дія	Очікуваний результат	Результат
1	Реєстрація нового користувача	Користувача створено в БД	Успішно
2	Вхід користувача	Авторизація виконана	Успішно
3	Створення групи	Групу збережено	Успішно
4	Додавання посилання	Посилання з’явилася у розкладі	Успішно
5	Генерація розкладу	Відображення за часом і днями	Успішно
6	Призначення до групи	Користувач доданий до групи	Успішно
7	Перегляд розкладу	Відображення розкладу	Успішно
8	Видалення користувача	Видалення користувача з БД	Успішно
9	Видалення групи	Видалнення з БД	Успішно
10	Створення побажання	Відображення побажання	Успішно

Результати випробування програмного застосунку для автоматизації формування розкладу занять в університеті «Розклад занять НУК», що проводилось у відповідності до Програми та методики випробувань (див. Додаток Д), показали його повну працездатність. Таким чином, можна зробити іисновок, що ПЗ відповідає вимогам до функціональності, продуктивності та зручності використання. Всі критичні дії були виконані без помилок, з відповідними результатами.

3 РЕЗУЛЬТАТИ РОЗРОБКИ ПРОГРАМНОГО ЗАСТОСУНКУ ДЛЯ АВТОМАТИЗАЦІЇ ФОРМУВАННЯ РОЗКЛАДУ ЗАНЯТЬ В УНІВЕРСИТЕТІ «РОЗКЛАД ЗАНЯТЬ В НУК»

У результаті виконання кваліфікаційної роботи було успішно розроблено програмний застосунок для автоматизації формування розкладу занять в університеті «Розклад занять в НУК». Розроблений програмний продукт повністю відповідає вимогам технічного завдання (Додаток А). Використання програмного застосунку дозволить підвищити ефективність роботи диспетчерської служби університету, зменшити витрат часу на адміністрування, мінімізувати імовірність виникнення помилок при введені та обробки даних. Використання ПЗ сприятиме підвищенню задоволеності користувачів за рахунок зручного інтерфейсу та широких можливостей керування.

Для фронтенд-розробки програмного застосунку було використано мову програмування JavaScript, мову розмітки веб-сторінок HTML та мову стилізації CSS. Для реалізації серверної частини використовувалися мова програмування Python та фреймворк Flask.

Тексти програмних модулів розробленого застосунку наведені у Додатку Б кваліфікаційної роботи.

Опис програми знаходиться у Додатку В.

Інструкція користувача програмного застосунку для автоматизації формування розкладу занять в університеті «Розклад занять в НУК» приведена у Додатку Г.

Програма та методика випробування представлено у Додатку Д.

Розмір програмного забезпечення. Програмний застосунок для автоматизації формування розкладу занять в університеті «Розклад занять в НУК» складається з наступних компонентів:

Веб-сервіс складається з 4 069 рядків коду, написаних мовами Python (588), JavaScript (1 409), CSS (1 261) та HTML (811). Розмір сирцевого коду на

диску становить приблизно 146.1 КБ. Після компіляції Python-скриптів та включення всіх ресурсів (зображення, шрифти, бібліотеки), повний розмір веб-сервісу становить понад 8 МБ.

Головним вихідним файлом для клієнтської частини є index.html, що відкривається у браузері, а також скрипти JavaScript і CSS, які відповідають за візуалізацію інтерфейсу. Серверна частина реалізована на Python та виконується у середовищі Python Runtime. Запуск веб-сервісу здійснюється через файл run.py.

Результати розробки програмного застосунку для автоматизації формування розкладу занять в університеті «Розклад занять в НУК» представлені на 3.1 – 3.11.

Першим, що побачить користувач, буде головна сторінка. Вона містить логотип системи та кнопку «Вхід» для авторизації. Під кнопкою є посилання «Зареєструйтесь», яке перенаправляє на форму реєстрації (рис. 3.1).

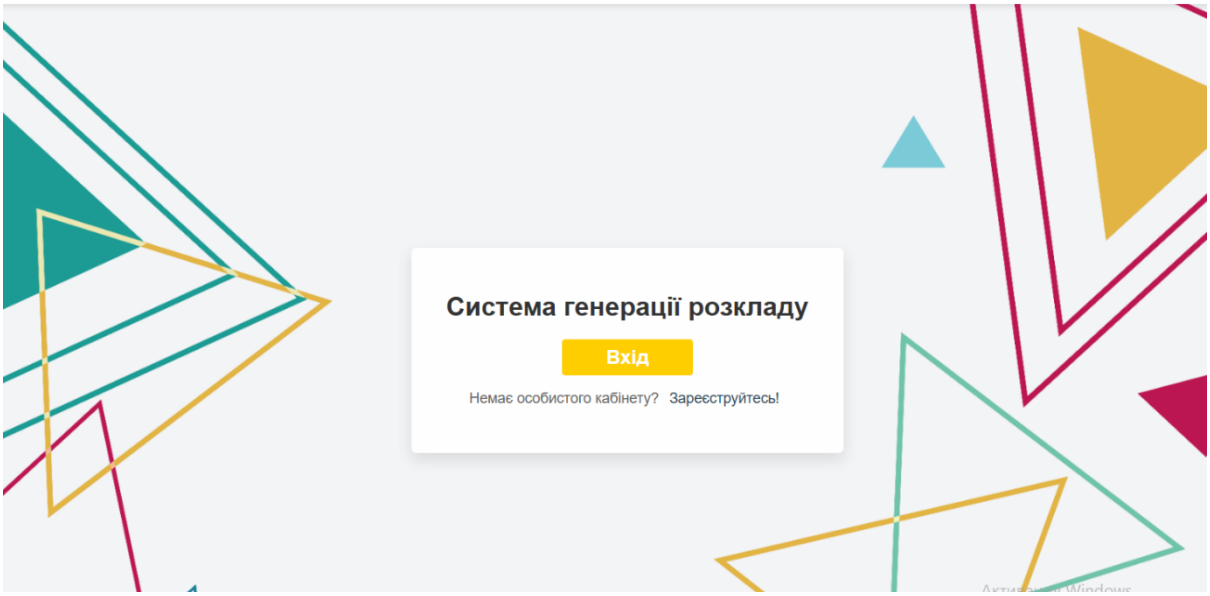


Рисунок 3.1 – Головна сторінка застосунку

Якщо користувач натисне на кнопку «Вхід», він побачить форму входу рис. 3.2), де потрібно ввести логін та пароль для доступу до особистого

кабінету.

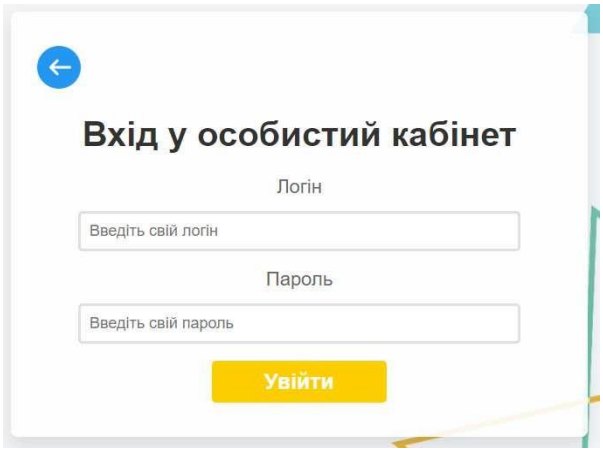


Рисунок 3.2 – Форма входу в особистий кабінет

Якщо користувач перейде по посиланню «Зареєструйтесь», він потрапить на сторінку реєстрації (рис. 3.3).

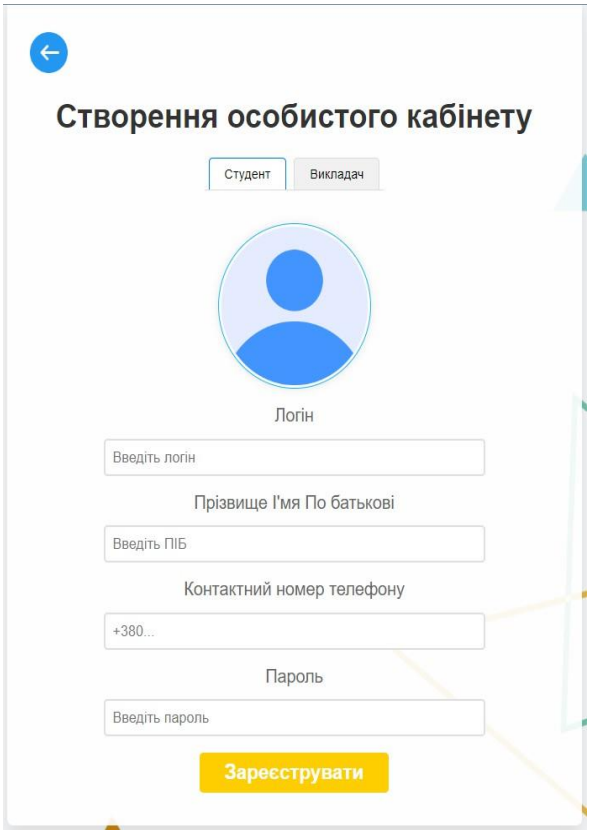


Рисунок 3.3 — Форма реєстрації користувача

Форма реєстрації має дві вкладки: «Студент» та «Викладач». У вкладці «Студент» користувач вводить свій логін, повне ім’я, контактний номер телефону та пароль. У вкладці «Викладач» додатково є поле з випадаючим списком, де можна обрати посаду (викладач, професор, доцент, асистент).

Якщо користувач з роллю «Студент» перейде до розділу «Розклад» то він побачить сторінку з таблицею розкладу занять на тиждень, організовану за днями та парами (рис. 3.4).

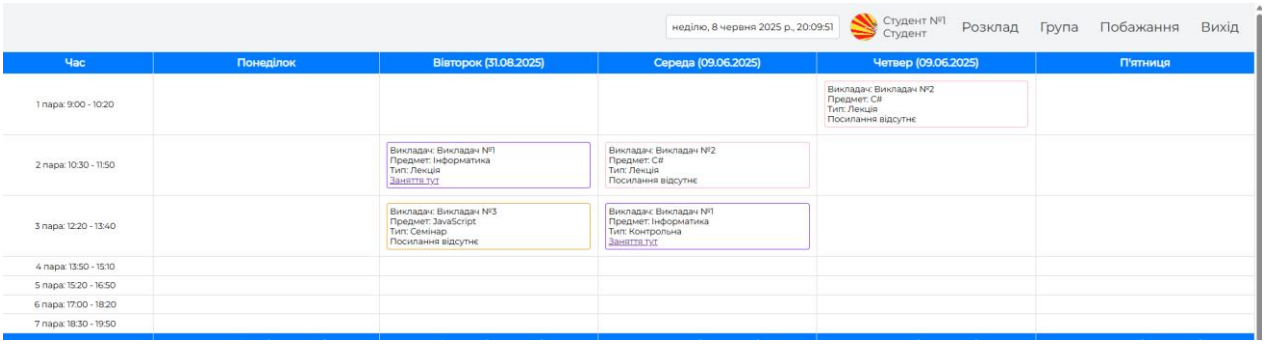


Рисунок 3.4 — Сторінка «Розклад» для користувача Студент

Посилання «Заняття тут» дозволяє перейти до онлайн-заняття або додаткової інформації, якщо посилання активне. Для викладача цей розділ буде аналогічним тільки з можливість вводу посилання та збереження його у базу даних.

Розділ «Група» для користувача з роллю «Студент» (рис. 3.5). Ця сторінка дозволяє переглянути інформацію про свою групу.

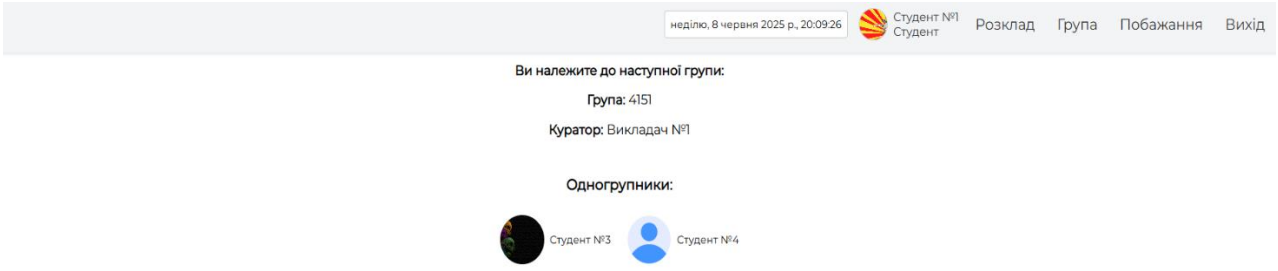


Рисунок 3.5 — Сторінка «Групи» для користувача Студента

Основний блок сторінки розділу містить повідомлення «Ви належите до наступної групи:», назву групи «4151», а також інформацію про куратора групи «Викладач №1». Під заголовком «Одногрупники:» відображаються профілі одногрупників з аватарами та іменами, наприклад, «Студент №3» та «Студент №4». Ця сторінка дозволяє студенту швидко отримати інформацію про свою групу, одногрупників та куратора.

Розділ «Побажання» для викладача та студента виглядає майже однаково, з деякими відмінностями в формі додавання побажань. На основному екрані відображається таблиця з часовими слотами і днями тижня. Користувачі можуть додавати свої побажання, натискаючи на кнопки з плюсом у відповідних комірках (рис. 3.6).

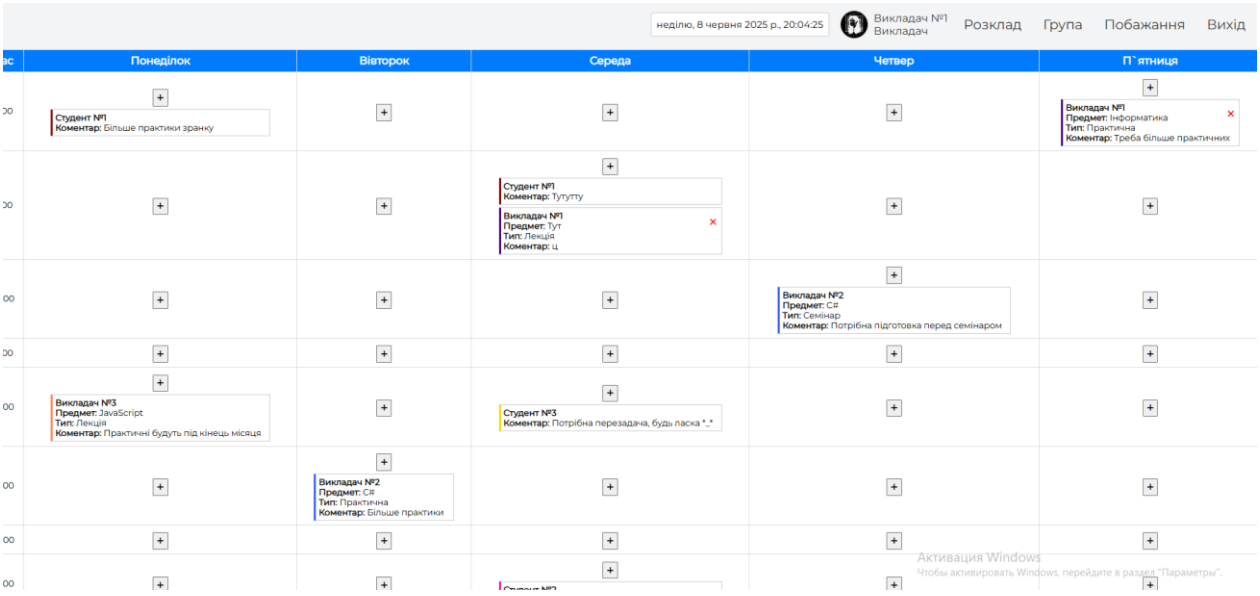


Рисунок 3.6 — Сторінка «Побажання»

Коли студент додає побажання, відкривається проста форма з полем для введення коментаря. Студенти можуть залишати коментарі щодо побажань до розкладу. Коли викладач додає побажання, відкривається більш детальна форма, що включає поле для введення назви предмету, вибору типу заняття (Лекція, Практична, Семінар, Контрольна) та коментаря. Це дозволяє

викладачам надавати більш конкретні побажання щодо розкладу (рис. 3.7).

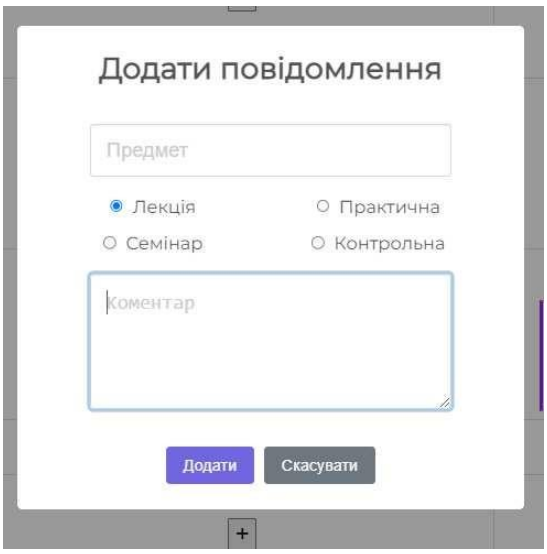


Рисунок 3.7 — Форма додавання побажання для викладача

Важливо зазначити, що для кожного користувача відображається червоний хрестик біля побажань, які він сам додав та вони ідентифікуються відповідним кольором для кращого вигляду. Це дозволяє користувачу видаляти свої власні побажання з розкладу.

Сторінка розділу «Група», яку бачить користувач з роллю «Викладач» (рис. 3.8). Цей розділ призначений для керування групами студентів.

Основний блок сторінки починається з повідомлення «Ви керуєте наступними групами:», під яким розташована таблиця з інформацією про групу. Заголовок таблиці вказує на назву групи. Колонки таблиці містять деталі про логін, ПІБ та номер телефону студентів. Окрема колонка «Дії» містить червоні іконки для видалення користувачів.

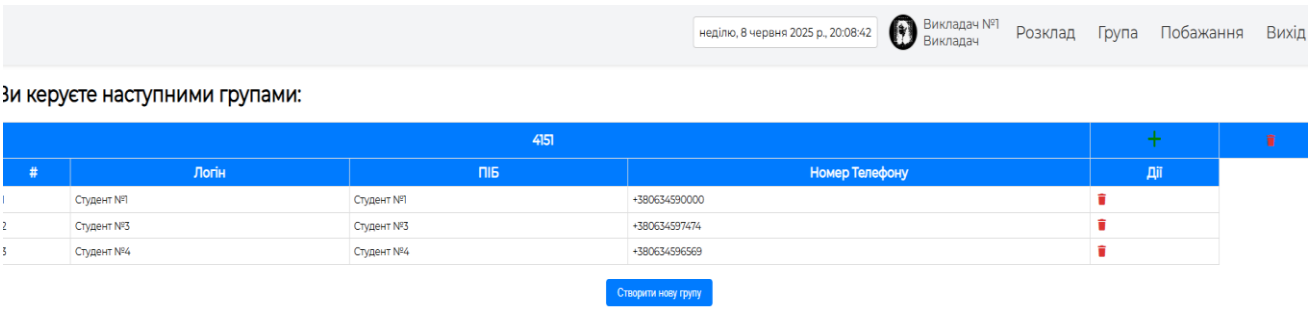


Рисунок 3.8 — Сторінка «Групи» для користувача викладача

Сторінка також надає викладачу кілька функціональних можливостей. Кнопка «Створити нову групу» відкриває форму створення групи, де можна ввести назву групи та додати студентів (рис. 3.9)

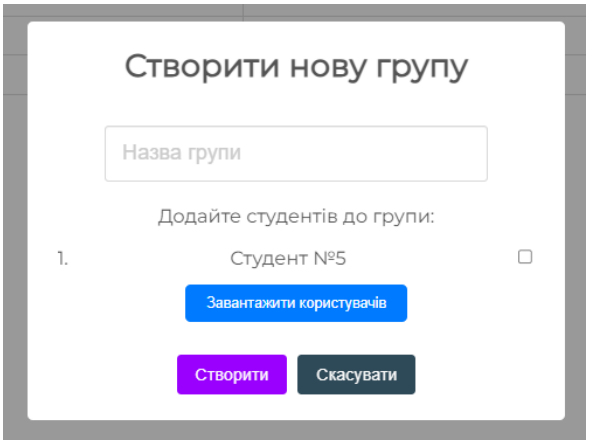


Рисунок 3.9 — Форма створення групи

Якщо викладач натисне на червону іконку кошика біля групи або студента, з’являється форма підтвердження видалення відповідно групи чи студента (рис. 3.10).

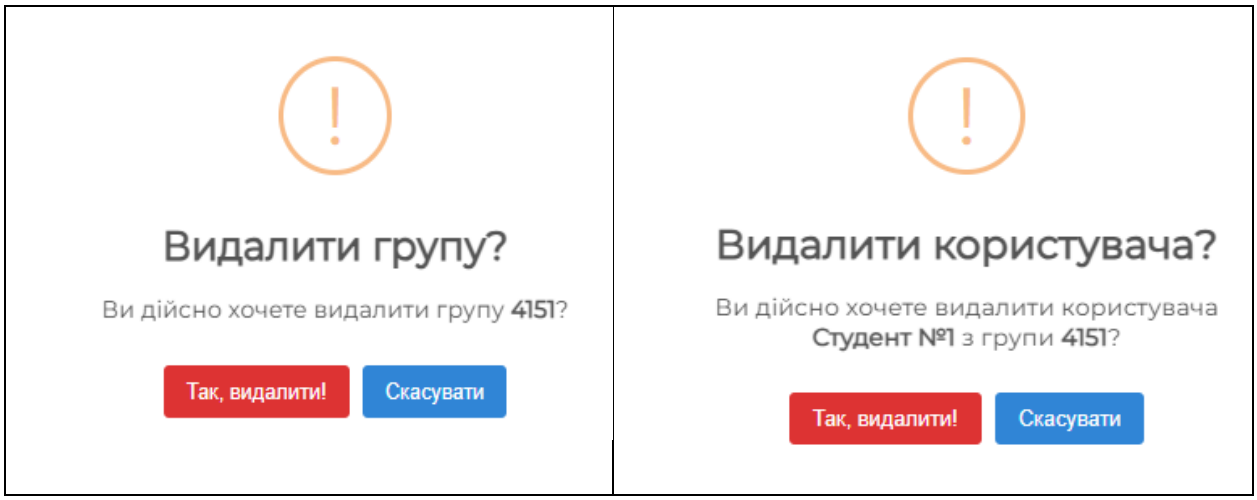


Рисунок 3.10 — Форми підтвердження видалення

Ця сторінка дозволяє викладачу легко керувати групами студентів, створювати нові групи та видаляти існуючі групи або окремих студентів із груп.

Сторінка розділу «Користувачі», яку бачить користувач з роллю «Адміністратор» (рис. 3.11). Цей розділ призначений для управління всіма користувачами системи.

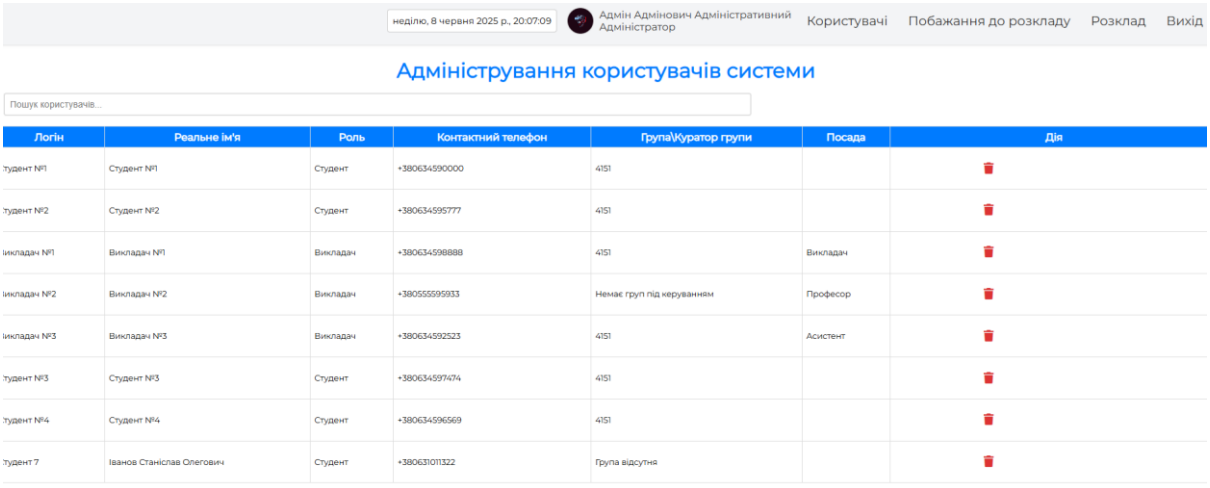


Рисунок 3.11 — Розділ «Адміністрування користувачів »

Під час випробування функціональності програмного застосунку «Розклад занять в НУК»» підтверджено:

− відображення відповідного повідомлення при неправильному вводі електронної пошти або паролю на сторінці "Логін" веб-сервісу;

− правильне відображення розкладу груп, рядка пошуку груп, перемикача дня тижня, тем та мови, а також кнопки меню на головній сторінці;

− відображення вікна редагування предмета при натисканні на нього;

− відображення вікна додавання предмета при натисканні на пусте поле;

− оновлення бази даних Firestore з відповідними змінами після додавання предмета.

Детальний опис випробування програмного забезпечення наведено у Додатку Д.

Виходячи з вищесказаного, можна зробити висновок, що було досягнуто мети роботи, а саме розроблено програмний застосунок для автоматизації формування розкладу занять в університеті ”Розклад занять НУК”, який задовольняє функціональним та нефункціональним вимогам та успішно пройшов тестування та випробування.

4 ОХОРОНА ПРАЦІ

Охорона праці — це комплекс заходів, методів і засобів, що мають на меті зменшення рівня виробничого травматизму та запобігання ситуаціям, які можуть негативно вплинути на здоров’я працівників на підприємствах або в окремих галузях.

У законодавстві України поняття «охорона праці» розкривається в статті 1 Закону України «Про охорону праці», де визначено, що це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних, лікувально-профілактичних заходів, спрямованих на збереження здоров’я працівників та підтримку їхньої працездатності в процесі трудової діяльності.

Цей закон також визначає ключові напрями реалізації конституційного права громадян на безпечні умови праці та охорону здоров’я під час виконання трудових обов’язків. Він регулює взаємовідносини між роботодавцем, власником підприємства та працівником у сфері охорони праці, а також встановлює єдиний підхід до організації безпеки праці на всій території України за участю уповноважених державних органів.

4.1 Аналіз небезпечних і шкідливих факторів у офісному приміщенні з персональними комп'ютерами

Головна завдання охорони праці – звести до мінімальної ймовірність ураження або захворювання працівника, а також виявлення і вивчення шкідливих факторів, їхній вплив на людину і навколишнє середовище.

Небезпечним виробничим чинником називається такий виробничий чинник, вплив котрого на працюючого у визначених умовах призведе до травми або до іншого раптового, різкого погіршення самопочуття [4].

До факторів, які впливають на виконання виробничого процесу відносять освітлення, електробезпеку, шум та вібрацію, пожежну безпеку, мікроклімат приміщення, пив та інші.

Шумом є всякий небажаний для людини звук. Нормою виробничого шуму є рівень звуку до 85 дБ. Якщо рівень перешкод становить 20 дБ, то такий шум не заважає розбірливості мови. З підвищенням рівня перешкод до 70 дБ та вище мова стає нерозбірливою.

Робота програміста має негативний вплив на організм. Найбільшому ризику піддаються зорова, опорно-рухова та нервово-психічна системи. Монітор випускає випромінювання декількох видів: рентгенівське, ультрафіолетове, інфрачервоне, електромагнітне. Для кожного з цих випромінювань розроблені гранично допустимі норми. Норми передбачають, що опроміненню піддається верхня частина тулуба. Згадані норми встановлені з розрахунку на кожен вид опромінення в окремо, хоча реально всі поля діють одночасно, а їх комплексний вплив досі не досліджено.

Відеоапаратура порушує рівновагу між позитивно і негативно зарядженими іонами в повітрі. Електростатичне поле дисплея притягає негативні іони, порушуючи загальний баланс атмосфери. Це також шкодити здоров'ю. Вже через годину роботи біля монітора спостерігається майже повне зникнення негативних іонів. Вісь чому необхідно, щоб до робочого місця за комп'ютером проникало свіже повітря.

Освітленість робочої зони комп'ютера повинна відповідати стандартам та бути вище яскравості монітора. Відстань від очей до екрана повинна бути не менше ніж півметра. Висота крісла має бути такою, щоб очі булі на одному рівні з центром монітора. Фахівці підтверджують, що саме очі найбільш страждають при роботі з комп'ютером. Занадто відсунутий монітор призводить до перенавантаження м' язів голови та шиї, через що тиск на їх роботові зростає приблизно в три вражай, судини шиї стискаються, погіршуючи кровопостачання до голови. Крім того, людині, що сидить у такій незручній позі, доводитися щоразу відкидати голову назад, для фіксування уваги на ближчих об' єктах. Це посилює вигин шийного відділу хребта. Через деякий годину це може призвести до головної болі та болю у кінцівках,

оскільки нерви, що відходять від спинного мозку в області шиї, простягаються до кінчиків пальців.

Малорухома та тривала сидяча поза за комп'ютером шкідлива для опорно-рухового апарату, що веде до застою крові в органах людини. Це особливо проявляється при фізіологічному положенні різних частин тіла і тривало повторюваних одноманітних рухах. Під час роботи за комп'ютером людина сидить кілька годин поспіль в незручному становищі. Це не тільки загрожує втомою і загальним знесиленням організму, а й може призвести до 81 розвитку остеохондрозу різних ділянок хребта - шийного, грудного, попереково-крижового.

Суттєвий вплив на стан організму ІТ-спеціаліста та його працездатність здійснює мікроклімат (метеорологічні умови), під яким розуміють клімат внутрішнього середовища цих приміщень, що визначається діючою на організм людини сукупністю температури, вологи, руху повітря та теплового випромінювання нагрітих поверхонь. Несприятливі метеорологічні умови приводять до швидкої втоми працівника, частішу його захворюваність та зниження продуктивності праці.

Світло впливає не лише на функції органів зору, а й на діяльність організму в цілому. Для створення оптимальних умів зорової роботи слід враховувати не лише кількість та якість освітлення, а й кольорове оточення.

При освітленні ІТ-відділу використовують природне освітлення, що створюється світлом неба, штучне, здійснюване електричними лампами, і сполучене, при якому у світлий година доби недостатній за нормами, природне освітлення доповнюється штучним.

Для забезпечення здоров'я працюючих приймаються необхідні запобіжні міри захисту, що повністю або частково ізолюють доступ до зони, в якій діють шкідливі фактори, та виключають їх дію.

До методів боротьби із шумом відносяться :

- зменшення шуму в джерелі виникнення;
- зміна напрямку випромінювання шуму;

- акустична обробка приміщень;

- установка звукоізолюючих огороджень, кожухи, екрани, кабіни;

- установка глушителів шуму;

- використання засобів індивідуального захисту (вкладиші, навушники, шоломи).

Заходи, щодо зниження негативного впливу мікроклімату:

- впровадження раціональних технологічних процесів;

- механізація та автоматизація виробничих процесів;

- захист працівників різними типами екранів;

- раціональна теплова ізоляція устаткування;

- раціональне розміщення устаткування;

- раціональне планування та конструкторське рішення виробничих приміщень;

- раціональні вентиляція та опалювання.

Для збереження здоров'я очей, та запобіганню перевтоми, робоче приміщення спеціаліста ІТ-відділу повинне відповідати наступним вимогам:

- створювати на робочій поверхні освітленість, що відповідає характеру зорової роботи і не є нижчою за встановлені норми;

- не повинне чинити засліплюючи дії як від самих джерел освітлення, так і від інших предметів, що знаходяться в полі зору;

- забезпечити достатню рівномірність освітлення; - не створювати на робочій поверхні тіней;

- повинне бути надійним в експлуатації, економічним та естетичним.

Робоче місце повинно відповідати сучасним стандартам ергономіки, забезпечуючи раціональне розміщення технічного обладнання (монітора, клавіатури, принтера) та документації на робочій поверхні.

1. Висота стільниці повинна бути регульованою в межах 680–800 мм. Розміри по ширині та глибині мають забезпечувати комфортне виконання операцій у межах досяжності рухового поля рук — рекомендована ширина становить 600–1400 мм, глибина — 800–1000 мм.

2. Конструкція столу повинна передбачати простір для ніг: висотою не менше 600 мм, шириною не менше 500 мм, глибиною на рівні колін — від 450 мм, а на рівні витягнутої ноги — не менше 650 мм. Стілець має бути обертовим, регульованим по висоті, з можливістю налаштування нахилу та кута сидіння і спинки, а також відстані між спинкою і переднім краєм сидіння. Поверхня сидіння має бути рівною, а передній край — плавно заокругленим.

3. Робочі місця доцільно розміщувати таким чином, щоб природне освітлення надходило здебільшого з лівого боку.

4. Екран монітора повинен бути розташований на відстані 600–700 мм від очей працівника, але не ближче 600 мм, залежно від розміру шрифтів і символів. Положення екрана має бути зручним для перегляду під вертикальним кутом, що становить приблизно +30° відносно нормального рівня погляду.

5. Клавіатуру слід встановлювати на стільниці на відстані 100–300 мм від її переднього краю. Її конструкція повинна передбачати наявність опори (з матеріалу з високим коефіцієнтом тертя), що запобігає мимовільному зміщенню. Також має бути можливість регулювання нахилу клавіатури в межах 5–15°.

6. Для зменшення впливу електромагнітних випромінювань комп’ютерного обладнання слід використовувати сертифіковані засоби захисту — наприклад, екрани-фільтри, локальні світлофільтри та інші пристрої, випробувані у відповідних лабораторіях і схвалені гігієнічними нормами.

У результаті проведеного дослідження можна зробити наступні висновки. Під час розумової або фізичної праці людина сприймає комплекс шкідливих виробничих чинників, які можуть позитивно або негативно відзначитись на її здоровий' ї та продуктивності праці. Рівень цих факторів залежить від виду робочої діяльності, але можна сказати, що навіть робота в офісі може нашкодити здоров'ю працівника. Керівництву підприємства необхідно забезпечувати працівників та робітників необхідними засобами

індивідуального захисту, вчасно слідкувати за станом робочих місць, що відповідають інструкції з техніки безпеки та не шкодити здоров'ю людини.

4.2 Розрахунок системи штучного освітлення у офісному приміщені з персональними комп'ютерами

У даному приміщенні слід передбачити організацію змішаного освітлення, яке поєднує природне та штучне світло. Природне освітлення забезпечується за рахунок бічного світлопроникнення через віконні прорізи, тоді як штучне світло використовується у випадках, коли природного освітлення недостатньо для створення комфортних умов праці.

У цьому випадку застосовується загальне штучне освітлення, розрахунок якого виконується за методикою світлового потоку, з урахуванням частки світла, що відбивається від поверхонь стелі та стін [5].

Габарити приміщення становлять: довжина А=5 м, ширина В=4 м, висота Н=2,5 м. Висота розташування робочої зони hр=1 м. Для освітлення передбачено використання світильників типу УПД. Згідно з нормативними вимогами, мінімальна освітленість для ламп розжарювання становить E_{min}=100лк, коефіцієнт відображення стелі ρп =70%, стін ρз = 50%, робочої поверхні ρр=30%. Напруга мережі 210 В.

Розрахуємо відстань від стелі до робочої поверхні:

$$Но = h - hр = 2.5 - 1 = 1.5 \text{ м}, \quad (4.1)$$

де Н - висота приміщення, м; hр - висота робочої поверхні, м.

Визначимо відстань від стелі до світильника (hс) :

$$hс= 0.2 * Но = 0.2 * 1.5 = 0.3 \text{ м} \quad (4.2)$$

Висота підвішування світильника над освітлюваною поверхнею (h) :

$$h = Но - hс = 1.5 - 0.3 = 1.2 \text{ м} \quad (4.3)$$

Висота підвішування світильника над підлогою (Нп):

$$Нп = h + hр = 1.2 + 1= 2.2 \text{ м} \quad (4.4)$$

Для того, щоб досягти найбільшої рівномірності освітлення приймаємо відношення L/h = 1,5. Таким чином відстань між центрами світильників :

$$L = 1.5 * h = 1.5 * 1.2 = 1.8 \text{ м.}$$

Визначимо необхідну кількість світильників:

$$N = S/l2 = 20 / 1.822 = 6.04 \quad (4.5)$$

$$N = S/l2 = 20 / 1.822 = 6.04 \ (6.5)$$

$$I(h * (A+B)) = 20 / (1.2 * 9) = 1.85.$$

При і = 0,57, коефіцієнтах відображення стелі ρп =70%, стін ρз = 50%, робочої поверхні ρр=30% для світильника УПД коефіцієнт використання світлового потоку η = 0,28.

Світловий потік однієї лампи, лм:

$$\Phi = \frac{E_{min} \times S \times Z \times K_з}{N \times \eta \times \gamma}; \quad (4.6)$$

де Emin - рівень мінімальної освітленості за нормами, лк;

S - освітлювана площа приміщення, м2 ;

Кз - коефіцієнт запасу;

Z - коефіцієнт мінімальної освітленості;

N - число світильників;

η - коефіцієнт використання світлового потоку ламп, встановлених у світильнику.

Коефіцієнт запасу Кз враховує експлуатаційне зниження освітленості, в порівнянні із запроектованою, внаслідок забруднення, а також зменшення світлового потоку ламп у процесі їх експлуатації. Для нашого приміщення коефіцієнт мінімальної освітленості (Кз) = 1,3.

Коефіцієнт мінімальної освітленості Z дорівнює відношенню середньої освітленості до мінімальної. Він враховує нерівномірність освітленості і залежить в основному від відношення відстаней між світильниками і від їх типів. Значення цього коефіцієнту для ламп розжарювання приймають Z=1,15.

Ф(6* 0.28) = 1334 лм.

За знайденим світловим потоком вибираємо лампу потужністю 200 Вт, що має світловий потік 3200 лм (Г125-135-200) найбільш близький до розрахункового.

При цьому фактична освітленість (лк) дорівнює:

$$E_{\phi} = E_{\min} \times \frac{\Phi_{л}}{\Phi_{р}}, \quad (4.7)$$

де Фл - світловий потік обраної лампи, лм;

Фр - світловий потік лампи, отриманої розрахунком, лм.

Єф= 75* 3200/1334 = 179,91 лк.

Загальна потужність освітлювальної установки Рз, Вт, визначається за формулою:

$$P_{з} = P_{л} \times N;, \quad (4.8)$$

де Рз - потужність обраної лампи, Вт

Рз = 200 / 6 = 1200 Вт =1,2 кВт

Загальна потужність освітлювальної установки становитиме при цьому 1200Вт.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було успішно вирішено задачу інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов, а саме розроблено програмний застосунок для автоматизації формування розкладу занять в університеті ”Розклад занять НУК”

В ході виконання роботи були вирішені наступні завдання:

- розглянути особливості автоматичного формування розкладу;

- на основі аналізу існуючих програмних продуктів для автоматизації складання розкладу була обґрунтована необхідність розробки програмного застосунку для автоматизації формування розкладу занять в університеті ”Розклад занять НУК” та виконана постановка задачі на розробку ПЗ;

- сформулювані та задокументувані вимоги на розробку програмного застосунку для автоматизації формування розкладу занять в університеті ”Розклад занять НУК”;

- виконані всі етапи проектування програмного забезпечення (ескізний, технічний, робочий);

- було проведено тестування програмного застосунку. Усі функції та елементи системи було перевірено вручну, і вони працюють відповідно до очікувань, що підтверджує успішність виконаного тестування;

- розроблена необхідна програмна документація.

Також в кваліфікаційній роботі були розглянути питанні охорони праці.

У майбутньому розроблений програмний застосунок для формування розкладу занять можна вдосконалювати, додаючи новий функціонал та покращуючи алгоритм побудови розкладу

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1 Математичне моделювання процесу складання розкладу в університеті [Електронний ресурс] // Науковий блог. URL: https://naub.oa.edu.ua/matematychne-modelyuvannya-protsesu-skladannya-rozkladu-v-universyteti/. (дата звернення: 22.02.2025).

2 Raúl Mencía C. M. Schedule Generation Schemes and Genetic Algorithm for the Scheduling Problem with Skilled Operators and Arbitrary Precedence Relations [Електронний ресурс] / Carlos Mencía María R. Sierra Raúl Mencía // ResearchGate. URL: https://www.researchgate.net/publication/283669174_Schedule_generation_schemes_and_genetic_algorithm_for_the_scheduling_problem_with_skilled_operators_and_arbitrary_precedence_relations (дата звернення: 25.03.2025).

3 Застосування генетичного алгоритму до задачі складання навчального розкладу. URL:https://pm-mm.dp.ua/index.php/pmmm/article/view/237 (дата звернення: 26.03.2025).

4 Жадібні алгоритми [Електронний ресурс] // DevZone. – 2020. – Режим доступу до ресурсу: https://devzone.org.ua/post/zadibni-alhorytmy (дата звернення: 26.03.2025).

5 Программа 1С Підприємство, що це таке? URL: https://itez.com.ua/what-is-1c.html/ (дата звернення: 25.04.2025).

6 University Timetabling.URL: https://www.unitime.org/ (дата звернення: 25.04.2025).

7 Flask. URL: https://flask.palletsprojects.com/en/3.0.x/ (дата звернення: 10.05.2025).

8 How I build Website using Flask, HTML, Javascript and Python. URL: https://medium.com/@fauzik354313/building-website-using-flask-html-javascript- and-python-study-case-791e336265f2 (дата звернення: 10.05.2025).

9 Visual Studio Code. URL: https://code.visualstudio.com/ (дата звернення: 12.05.2025).

10 Про охорону праці: Закон України від 14.10.1992р № 2694-XII. URL: https://zakon.rada.gov.ua/laws/show/2694-12#Text. (дата звернення: 10.03.2025)/

11 ДСТУ EN ISO 9241-5:2004. Вимоги до ергономіки офісного обладнання для роботи з дисплеями.[Чинний від 2006-01-01].Вид. офіц. Київ: Держспоживстандарт України, 2006. 34с. (Інформація та документація).

12 ДБН В.2.2-10-2001. Приміщення житлові та громадські. Основи проектування. [Чинний від 2001-06-04]. Вид. офіц. Київ, 2001. 171с. (Інформація та документація)

ДОДАТОК А

ТЕХНІЧНЕ ЗАВДАННЯ

на розробку програмного застосунку для автоматизації формування розкладу занять в університеті «Розклад занять НУК»

А.1 Вступ

Програмне забезпечення призначена для підвищення ефективності організації навчального процесу в університеті шляхом автоматизації формування розкладу занять. Застосунок дозволяє оптимізувати планування навчальних занять, уникати конфліктів при складанні розкладу, враховувати побажання викладачів, студентів і адміністрації, а також забезпечувати зручний доступ до актуальної інформації про розклад.

А.2 Підстави для розробки

Підставою для розробки програмного забезпечення є наказ на затвердження тем кваліфікаційних робіт бакалаврів зі спеціальності 121 «Інженерія програмного забезпечення» в Національному університеті кораблебудування ім. адмірала Макарова.

А.3 Призначення розробки

Розробка призначена для автоматизації процесу складання, редагування, перегляду та публікації розкладу занять в університеті. Програмна система передбачає багаторівневий доступ до функцій залежно від ролі користувача: адміністратора, викладача чи студента.

А.4 Вимоги до програми або програмного виробу

А.4.1 Вимоги до функціональних характеристик:

- авторизація користувачів за ролями (адміністратор, викладач, студент);
- формування розкладу занять вручну та автоматично з урахуванням таких параметрів, як аудиторне навантаження, наявність викладачів, груп, аудиторій;
- адміністрування академічних груп;
- збереження та редагування онлайн-посилань на дистанційні заняття;
- експорт розкладу у форматах PDF/Excel;
- синхронізація з календарем Google.

А.4.2 Вимоги до надійності

- коректна обробка помилок введення та збереження;
- забезпечення резервного копіювання бази даних;
- стабільна робота при одночасному доступі кількох користувачів.

А.4.3 Умови експлуатації:

- кліматичні умови для коректної роботи програми співпадають із вимогами для технічних засобів, на яких вона буде встановлена;
- використання у локальній мережі навчального закладу або на сервері з доступом через браузер;
- програмна система розрахована на користувача із середнім рівнем комп’ютерної грамотності.

А.4.4 Вимоги до складу і параметрів технічних засобів:

- мінімальні ресурси клієнтської машини: 2 ГБ ОЗП;
- серверна частина: 4 ГБ ОЗП.

А.4.5 Вимоги до інформаційної та програмної сумісності:

- функціонування під управлінням ОС Windows 10+;

- підтримка REST API (опційно) для зовнішніх інтеграцій;

- браузер із підтримкою HTML5;

- серверна частина: підтримка Python 3.10+, СУБД PostgreSQL або SQLite.

А4.6 Вимоги до маркування та пакування

Вимог до маркування та пакування не передбачається.

А.4.7 Вимоги до транспортування і зберігання

Файли проєкту зберігаються на сервері кафедри або в захищеному хмарному сховищі з налаштованим резервним копіюванням.

А.4.8 Спеціальні вимоги

Спеціальних вимог не передбачається.

А.5 Вимоги до програмної документації:

- Технічне завдання;

- Опис програми;

- Інструкція користувача;

- Програма та методика випробувань;

-Тексти програмних модулів.

А.6 Техніко-економічні показники

Техніко-економічні показники не розраховуються.

А.7 Стадії і етапи розробки

Стадії та етапи розробки програмного застосунку приведено у табл А.1

Таблиця А.1-Стадії та етапи розробки ПЗ

Стадія	Етапи робіт	Термін виконання
Технічне завдання	Збір та аналіз вимог, документування вимог.	15.03.2025
Ескізний проект	Розробка діаграми варіантів використання, розробка специфікацій, розробка діаграм діяльності.	28.03.2025
Технічний проект	Розробка статичної моделі ПЗ, розробка динамічної моделі ПЗ, деталізація алгоритмів, модулів	25.04.2025
Робочий проект	Вибір засобів розробки та мов програмування, розробка програмного коду, програмування, тестування, документація	12.05.2025

А.8 Порядок контролю і приймання

На кожному етапі створення програмного забезпечення проводиться контроль за відповідністю результатів технічному завданню. Проміжні результати узгоджуються з керівником практики. Приймання готової системи здійснюється на підставі програми і методики випробувань.

Прийом проводяться відповідно до програми і методики випробувань і документують за допомогою протоколу проведення випробувань. У разі

знаходження помилок підчас прийому програмного виробу складається акт про знайдені помилки, який підписуються представниками замовника і розробника і затверджуються керівником організації – замовника та організації – розробника. Розробник повинен на протязі не більше ніж 2 тижнів виправити зазначені помилки і оповістити замовника про повторне проведення перевірки

ДОДАТОК Б

ВИХІДНІ ТЕКСТИ ПРОГРАМНИХ МОДУЛІВ

Лістинг Б.1. Програмний код для обробки запитів на реєстрацію користувачів

```
@main.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        username = request.form['username']
        realname = request.form['realname']
        phone = request.form['phone']
        password = request.form['password']
        position = request.form.get('position')
        existing_user                                    =
User.query.filter_by(username=username).first()
        existing_phone                                   =
User.query.filter_by(phone=phone).first()

        if existing_user:
            flash('Користувач з таким логіном вже існує.
Спробуйте інший!', 'error')
            return render_template('register.html')

        if existing_phone:
            flash('Користувач з таким номером телефону вже
існує.', 'error')
            return render_template('register.html')

        if not (phone.startswith('+') and len(phone) == 13
and phone[1:].isdigit()):
            flash('Номер телефону повинен починатися з +,
містити тільки цифри та мати довжину 12 символів без +.', 'error')
            return render_template('register.html')

        if    'profile_photo'    in    request.files    and
request.files['profile_photo'].filename != '':
            profile_photo                                =
save_profile_image(request.files['profile_photo'])
        else:
            profile_photo = 'user.png'

        if  len(username)  <  8  or  len(password)  <  8  or
len(realname) < 8:
```

```
                flash('Поля Логін, Пароль та ПІБ мають містити не
менше 8 символів!', 'error')
                return render_template('register.html')

            if not username or not password or not realname or
not phone:
                flash('Заповніть всі необхідні поля!', 'error')
                return render_template('register.html')

            if username == "admin_admin" and password ==
"admin_admin":
                role = 'admin'
            else:
                role = 'teacher' if position else 'user'

            new_user = User(
                username=username,
                real_name=realname,
                phone=phone,
                password=password,
                profile_photo=profile_photo,
                role=role,
                position=position
            )
            db.session.add(new_user)
            db.session.commit()
            flash('Реєстрація пройшла успішно!', 'success')
            session.clear()
            session['user_id'] = new_user.id

            if new_user.role == 'admin':
                return redirect(url_for('user.admin_home'))
            else:
                return redirect(url_for('user.user_home'))

        return render_template('register.html')
```

Лістинг Б.2. Програмний код для перевірки правильності формату телефонного номера та обробки завантаження профільного фото

```
    if not (phone.startswith('+') and len(phone) == 13 and
phone[1:].isdigit()):
                flash('Номер телефону повинен починатися з +,
містити тільки цифри та мати довжину 12 символів без +.', 'error')
```

```
            return render_template('register.html')
```

Лістинг Б.3. Програмний код для створення нового об'єкту

```
    if username == "admin_admin" and password == "admin_admin":
            role = 'admin'
        else:
            role = 'teacher' if position else 'user'

        new_user = User(
            username=username,
            real_name=realname,
            phone=phone,
            password=password,
            profile_photo=profile_photo,
            role=role,
            position=position
        )
        db.session.add(new_user)
        db.session.commit()
        flash('Реєстрація пройшла успішно!', 'success')
        session.clear()
        session['user_id'] = new_user.id

        if new_user.role == 'admin':
            return redirect(url_for('user.admin_home'))
        else:
            return redirect(url_for('user.user_home'))

    return render_template('register.html')
```

Лістинг Б.4. Програмний код функції авторизації

```
@main.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']

        if len(username) < 8 or len(password) < 8:
            flash('Поля Логін та Пароль мають містити не
менше 8 символів!', 'error')
```

```
                return render_template('login.html')

            user    =    User.query.filter_by(username=username,
password=password).first()

            if not user:
                flash('Помилка входу! Перевірте дані та спробуйте
ще раз.', 'error')
                return render_template('login.html')

            user_login(user)
            flash('Вхід успішний!', 'success')
            if user.role == 'admin':
                return redirect(url_for('user.admin_home'))
            else:
                return redirect(url_for('user.user_home'))

        return render_template('login.html')
```

Лістинг Б.5. Програмний код, що відображає домашню сторінку користувача після успішного входу

```
    @user.route('/user_home')
    @login_required
    def user_home():
        if current_user.is_authenticated:
            profile_photo = current_user.profile_photo
            return             render_template('user_home.html',
user=current_user, profile_photo=profile_photo)
        else:
            return "Користувача не знайдено, будь-ласка увійдіть
ще раз!", 404
```

HTML- код забезпечує структуру таблиці для відображення списку користувачів, а також поле для пошуку користувачів:

```
    <div id="dynamic-content">
            <div id="users" class="content">
                <h1>Адміністрування користувачів системи</h1>
                <input       type="text"       id="search-input"
placeholder="Пошук користувачів..." oninput="fetchUsers()">
```

```
                <table class="group-table">
                    <thead>
                        <tr>
                            <th>Логін</th>
                            <th>Реальне ім'я</th>
                            <th>Роль</th>
                            <th>Контактний телефон</th>
                            <th>Група\Куратор групи</th>
                            <th>Посада</th>
                            <th>Дія</th>
                        </tr>
                    </thead>
                    <tbody id="users-body"></tbody>
                </table>
            </div>
```

Лістинг Б.6. Програмний код, для додавання студентів до групи

```
    @main.route('/add_students_to_group', methods=['POST'])
    def add_students_to_group():
        data = request.get_json()
        group_id = data.get('groupId')
        user_ids = data.get('users', [])

        if not group_id or not user_ids:
            return    jsonify({'success':    False,    'message':
'Invalid data'}), 400

        group = Group.query.get(group_id)
        if not group:
            return jsonify({'success': False, 'message': 'Group
not found'}), 404

        added_users = []
        for user_id in user_ids:
            user = User.query.get(user_id)
            if user:
                if user.group_id != group.id:
                    user.group_id = group.id
                    added_users.append(user_id)

        db.session.commit()

        if added_users:
```

```
            return    jsonify({'success':    True,    'message':
'Students added to group', 'added_users': added_users}), 200
        else:
            return  jsonify({'success':  False,  'message':  'No
students were added'}), 400
```

Лістинг Б.7. Алгоритм генерації розкладу

При натисканні кнопки «Згенерувати розклад» запускається процес збирання даних та генерації розкладу.

```
    document.getElementById('generate_schedule').addEventListen
er('click', function() {
        const                    totalLoad                    =
parseInt(document.getElementById('total_load').value);
        const                   maxClasses                    =
parseInt(document.getElementById('max_classes').value);
        const                  scheduleType                   =
document.querySelector('input[name="schedule_type"]:checked').va
lue;
        const scheduleDuration = 4;
        const                allowTwoClasses                  =
document.querySelector('input[name="allow_two_classes"]:checked'
).value === 'yes';
```

отримує атрибути data-day, data-period та data-week, що визначають день, період та тиждень відповідно:

```
    function      generateSchedule(totalLoad,      maxClasses,
scheduleType, scheduleDuration, teachers, allowTwoClasses) {
        const timeSlots = [
        'Перша пара: 9:00 - 10:20',
        'Друга пара: 10:30 - 11:50',
        'Третя пара: 12:20 - 13:40',
        'Четверта пара: 13:50 - 15:10',
        'П’ята пара: 15:20 - 16:50',
        'Шоста пара: 17:00 - 18:20',
        'Сьома пара: 18:30 - 19:50'
    ];


        const  days  =  ['Понеділок',  'Вівторок',  'Середа',
'Четвер', 'П\'ятниця'];

        const weekBodies = [
            document.getElementById('week-1-body'),
```

```
            document.getElementById('week-2-body'),
            document.getElementById('week-3-body'),
            document.getElementById('week-4-body')
        ];

        weekBodies.forEach(weekBody => {
            weekBody.innerHTML = '';
            for (let i = 0; i < timeSlots.length; i++) {
                const row = document.createElement('tr');
                const timeCell = document.createElement('td');
                timeCell.innerText = timeSlots[i];
                row.appendChild(timeCell);

                for (const day of days) {
                    const cell = document.createElement('td');
                    cell.classList.add('droppable');
                    cell.dataset.day = day;
                    cell.dataset.period = i + 1;
                    cell.dataset.week                                =
weekBodies.indexOf(weekBody) + 1;
                    row.appendChild(cell);
                }

                weekBody.appendChild(row);
            }
        });
```

Здійснюється ініціалізація змінних для подальшого розподілу занять у розкладі. Визначається загальна кількість слотів, які необхідно заповнити, та розподіл типів занять:

```
    document.querySelectorAll('.schedule-table          th[data-
day]').forEach(header => {
            const currentDate = new Date();
            header.setAttribute('data-date',
currentDate.toISOString().split('T')[0]);
        });

        const  totalSlots  =  days.length  *  timeSlots.length  *
scheduleDuration;
        const slotsToFill = Math.floor((totalSlots * totalLoad)
/ 10);

        const                  typeDistribution                  =
getTypeDistribution(scheduleType);
```

```
        const dailySlots = {};
        for (let week = 0; week < scheduleDuration; week++) {
            dailySlots[week] = {};
            days.forEach(day => {
                dailySlots[week][day] = 0;
            });
        }

        const types = [];
        Object.keys(typeDistribution).forEach(type => {
            const count = Math.floor(slotsToFill *
(typeDistribution[type] / 100));
            for (let i = 0; i < count; i++) {
                types.push(type);
            }
        });

        for (let i = types.length - 1; i > 0; i--) {
            const j = Math.floor(Math.random() * (i + 1));
            [types[i], types[j]] = [types[j], types[i]];
        }
```

Основний цикл розподілу занять по слотам розкладу. Перевіряється наявність вільних слотів, заповнюються вони заняттями, та здійснюється їх відображення на сторінці:

```
                    const teacherIndex = filledSlots %
teachers.length;
                    const teacher = teachers[teacherIndex];

                    const type = types[filledSlots %
types.length];

                    if (!occupiedSlots[week]) {
                        occupiedSlots[week] = {};
                    }
                    if (!occupiedSlots[week][day]) {
                        occupiedSlots[week][day] = {};
                    }
                    if (!occupiedSlots[week][day][teacher.name])
{
                        occupiedSlots[week][day][teacher.name] =
new Set();
                    }
```

```
                    const teacherDailyCount =
[...occupiedSlots[week][day][teacher.name]].filter(item =>
item.includes(teacher.subject)).length;

                    if (allowTwoClasses || (!allowTwoClasses &&
teacherDailyCount < 1)) {
                        const scheduleItem =
document.createElement('div');
                        scheduleItem.classList.add('schedule-
item');
                        scheduleItem.style.borderColor =
teacher.color;
                        console.log(`Setting data-teacher-id:
${teacher.id}`); // Перевірка
                        scheduleItem.setAttribute('data-
teacher-id', teacher.id);
                        scheduleItem.setAttribute('data-
subject', teacher.subject);
                        scheduleItem.setAttribute('data-type',
type);
                        scheduleItem.innerHTML =
`<div>${teacher.name}</div><div>${teacher.subject}</div><div>${t
ype}</div>`;
                        cell.appendChild(scheduleItem);
                        filledSlots++;
                        dailySlots[week][day]++;

occupiedSlots[week][day][teacher.name].add(`${teacher.subject}:$
{type}`);
                    }
                }
            }
            slotIndex++;
            iterations++;
            if (slotIndex >= totalSlots) {
                slotIndex = 0;
            }
        }

        initDragAndDrop();
    }
```

Лістинг Б.8. Програмний код, що відповідає за ініціалізацію функціоналу перетягування та видалення елементів викладачів по комірках

розкладу. Кожен елемент розкладу налаштовується на можливість перетягування:

```
    function initDragAndDrop() {
        const scheduleItems = document.querySelectorAll('.schedule-item');
        const droppables = document.querySelectorAll('.droppable');
        const scheduleTable = document.querySelector('.schedule-table');
        let draggedItem = null;
        let tooltipTimeout = null;

        scheduleItems.forEach(item => {
            item.setAttribute('draggable', false);

            item.addEventListener('mousedown', function(e) {
                if (e.button === 2) {
                    e.preventDefault();
                    this.remove();
                } else {
                    this.setAttribute('draggable', true);
                }
            });

            item.addEventListener('mouseup', function() {
                this.setAttribute('draggable', false);
            });
```

Лістинг Б.9. Функція collectScheduleData збирає всі дані з комірок таблиці розкладу, включаючи інформацію про день, період, тиждень, викладача, предмет, тип заняття та інші параметри:

```
    function collectScheduleData() {
        const scheduleItems = document.querySelectorAll('.schedule-item');
        const scheduleData = [];
        const dateMap = {};

        document.querySelectorAll('th[data-day]').forEach(header => {
            const day = header.getAttribute('data-day');
            const date = header.getAttribute('data-date');
            if (date) {
                dateMap[day] = date;
```

```
            }
        });

        scheduleItems.forEach(item => {
            const dayElement = item.closest('.droppable');
            const day = dayElement.dataset.day || null;
            const period = dayElement.dataset.period || null;
            const week = dayElement.dataset.week || null;
            const date = dateMap[day] || null;
```

Функціонал викладача

Лістинг Б.10. Програмний код функції createGroup, яка відповідає за створення нової групи. Вона використовує бібліотеку SweetAlert для відображення модального вікна, де викладач може ввести назву групи та додати студентів до неї:

```
    function createGroup() {
        Swal.fire({
            title: 'Створити нову групу',
            html: `
                <input type="text" id="group-name" class="swal2-
input" placeholder="Назва групи">
                <p>Додайте студентів до групи:</p>
                <div id="users-list"></div>
                <button      id="load-users"      class="button"
onclick="loadUsers()">Завантажити користувачів</button>
            `,
            showCancelButton: true,
            confirmButtonColor: '#9a00fd',
            cancelButtonColor: '#2e4957',
            confirmButtonText: 'Створити',
            cancelButtonText: 'Скасувати',
            preConfirm: () => {
                const                groupName                =
Swal.getPopup().querySelector('#group-name').value.trim();
                const              selectedUsers              =
Array.from(document.querySelectorAll('.user-
checkbox:checked')).map(cb => cb.value);

                if (!groupName) {
                    Swal.showValidationMessage('Назва групи не
може бути пустою');
                    return false;
                }
```

```
                if (selectedUsers.length === 0) {
                    Swal.showValidationMessage('Додайте хоча б
одного студента до групи');
                    return false;
                }

                return { groupName: groupName, users:
selectedUsers };
            }
        }).then((result) => {
            if (result.isConfirmed) {
                saveGroup(result.value);
            }
        });
    }
```

Лістинг Б.11. Програмний код функція confirmDeleteGroup, що підтверджує видалення групи, а deleteGroup виконує цю дію на сервері. Функція openAddStudentForm відкриває форму для додавання студентів до групи, а addStudentsToGroup відправляє дані на сервер для збереження:

```
    function confirmDelete(userId, userName, groupName) {
        Swal.fire({
            title: 'Видалити користувача?',
            html: `Ви дійсно хочете видалити користувача
<strong>${userName}</strong> з групи
<strong>${groupName}</strong>?`,
            icon: 'warning',
            showCancelButton: true,
            confirmButtonColor: '#d33',
            cancelButtonColor: '#3085d6',
            confirmButtonText: 'Так, видалити!',
            cancelButtonText: 'Скасувати'
        }).then((result) => {
            if (result.isConfirmed) {
                deleteUserFromGroup(userId, groupName);
            }
        });
    }
```

Лістинг Б.12. Програмний код функції openAddMessageModal, що відкриває модальне вікно, де користувач може додати побажання до певного

часового слота. Викладачі можуть також вказати предмет і тип заняття. Функція addMessageToCell додає нове побажання до відповідної клітинки розкладу, а saveEvent зберігає це побажання на сервері:

```
    function openAddMessageModal(cellId, userName, userRole, userId) {
        const [day, time_slot] = cellId.split('-');
        let htmlContent = `
            <textarea id="message-comment" class="swal2-textarea" placeholder="Коментар"></textarea>
        `;

        if (userRole === 'teacher') {
            htmlContent = `
                <input type="text" id="message-subject" class="swal2-input" placeholder="Предмет">
                <div class="message-type-container">
                    <label><input type="radio" name="message-type" value="Лекція" class="swal2-radio"> Лекція</label>
                    <label><input type="radio" name="message-type" value="Практична" class="swal2-radio"> Практична</label>
                    <label><input type="radio" name="message-type" value="Семінар" class="swal2-radio"> Семінар</label>
                    <label><input type="radio" name="message-type" value="Контрольна" class="swal2-radio"> Контрольна</label>
                </div>
                ${htmlContent}
            `;
        }

        Swal.fire({
            title: 'Додати повідомлення',
            html: htmlContent,
            showCancelButton: true,
            confirmButtonText: 'Додати',
            cancelButtonText: 'Скасувати',
            preConfirm: () => {
                const subject = userRole === 'teacher' ? document.getElementById('message-subject').value : null;
                const type = userRole === 'teacher' ? document.querySelector('input[name="message-type"]:checked')?.value : null;
                const comment = document.getElementById('message-comment').value;
```

```
                if (!comment || (userRole === 'teacher' &&
(!subject || !type ))) {
                    Swal.showValidationMessage('Будь ласка,
заповніть всі поля');
                }
                return { day, time_slot, subject, type, comment,
user_id: parseInt(userId) };
            }
        }).then((result) => {
            if (result.isConfirmed) {
                addMessageToCell(cellId, userName, result.value,
userId);
                saveEvent(result.value);
            }
        });
    }
```

ДОДАТОК В

ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

В.1. Загальні відомості

В.1.1 Позначення і найменування: Програмний застосунок для автоматизації формування розкладу занять в університеті «Розклад занять в НУК».

Програмне забезпечення призначене для підвищення ефективності організації навчального процесу в університеті шляхом автоматизації формування розкладу занять. Застосунок дозволяє оптимізувати планування навчальних занять, уникати конфліктів при складанні розкладу, враховувати побажання викладачів, студентів і адміністрації, а також забезпечувати зручний доступ до актуальної інформації про розклад.

В.1.2 Програмне забезпечення, необхідне для функціонування програми

Для коректної роботи із програмним забезпеченням необхідно:

- наявність ліцензійної версії операційної системи Windows 10 або вище;
- підтримка REST API (опційно) для зовнішніх інтеграцій;
- браузер із підтримкою HTML5;
- серверна частина вимагає підтримки Python 3.10+;
- СУБД PostgreSQL або SQLite.

− інтернет-з’єднання. Програмне забезпечення вимагає доступу до Інтернету.

В.1.3 Мови програмування

Мови програмування, використані в розробці Програмний застосунок для автоматизації формування розкладу занять в університеті «Розклад занять в НУК»:

- для фронтенд-розробки обрано HTML, CSS та JavaScript. Ці мови забезпечують створення інтуїтивно зрозумілого та інтерактивного користувацького інтерфейсу;
- для бекенд-розробки обрано Python з використанням фреймворку Flask.

В.2 Функціональне призначення

В. 2.1 Класи вирішуваних завдань і призначення програми

Програмне забезпечення повинно виконувати наступні функції:

- авторизація користувачів за ролями (адміністратор, викладач, студент);
- формування розкладу занять вручну та автоматично з урахуванням таких параметрів, як аудиторне навантаження, наявність викладачів, груп, аудиторій;
- адміністрування академічних груп;
- збереження та редагування онлайн-посилань на дистанційні заняття;
- експорт розкладу у форматах PDF/Excel;
- синхронізація з календарем Google.

В.2.2 Функціональні обмеження

Відсутні

В.3 Опис логічної структури

В.3.1 Використовувані методи

При розробці програмного застосунку для автоматизації формування розкладу занять в університеті «Розклад занять в НУК» було використано обєктно-орієнтовані методи програмування.

В.3.2 Алгоритм програми

Програмний застосунок для автоматизації формування розкладу занять в університеті «Розклад занять в НУК» має наступну логічну структуру:

- Авторизація. Реєстрація та авторизація користувачів є ключовими функціями системи, що забезпечують контроль доступу до різних частин веб-застосунку. Цей процес включає в себе перевірку даних, збереження нових користувачів та автентифікацію існуючих користувачів. Функція реєстрації обробляє запити на створення нових користувачів. Функція авторизації обробляє запити на вхід користувачів у систему,. перевіряє правильність введених логіна і пароля, автентифікує користувача і перенаправляє його на відповідну домашню сторінку;

- Адміністрування користувачів. Адміністрування користувачів включає в себе управління списком користувачів, пошук по користувачах, а також видалення користувачів з системи;

- Алгоритм генерації розкладу запускає процес збирання даних та генерації розкладу.

Функціонал викладача. Функціонал викладача щодо управління групами студентів. Викладач може створювати нові групи, додавати студентів до груп, видаляти студентів з груп, а також видаляти самі групи. Крім того, викладач може переглядати існуючі групи та керувати ними. А саме:

- функція showContent керує відображенням відповідного контенту;

- функція createGroup відповідає за створення нової групи, використовує бібліотеку SweetAlert для відображення модального вікна, де викладач може ввести назву групи та додати студентів до неї;

- функція loadUsers завантажує список користувачів, які ще не знаходяться в групах, для додавання їх до новоствореної групи;

- функція saveGroup зберігає новостворену групу, відправляючи дані на сервер;

- функція confirmDelete підтверджує видалення користувача з групи, а deleteUserFromGroup виконує цю дію на сервері;

- функція confirmDeleteGroup підтверджує видалення групи, а deleteGroup виконує цю дію на сервері;

- функція openAddStudentForm відкриває форму для додавання студентів до групи, а addStudentsToGroup відправляє дані на сервер для збереження:;

- функція loadGroups завантажує та відображає всі групи, якими керує викладач.

Взаємодія користувача з розкладом. Викладачі можуть додавати предмети, типи занять і коментарі до конкретних часових слотів, тоді як інші користувачі можуть додавати лише коментарі. Код починається з ініціалізації сторінки, де отримуються роль, ID та ім'я користувача. На основі цих даних генерується таблиця розкладу і завантажуються всі події для поточного користувача. А саме:

- функція generateScheduleTable створює таблицю розкладу, де кожна клітинка представляє певний день і час, та додає кнопку для додавання побажань;

- функція openAddMessageModal відкриває модальне вікно, де користувач може додати побажання до певного часового слота. Викладачі можуть також вказати предмет і тип занятт;

- функція addMessageToCell додає нове побажання до відповідної клітинки розкладу,

- saveEvent зберігає це побажання на сервері;

- функція fetchAllEvents завантажує всі побажання користувача з сервера і відображає їх у відповідних клітинках розкладу. Цей код забезпечує повну функціональність взаємодії користувачів з розкладом, дозволяючи додавати, переглядати та видаляти побажання до занять.

В.4 Технічні засоби та програмні засоби

Вимоги до складу і параметрів технічних засобів:

- мінімальні ресурси клієнтської машини: 2 ГБ ОЗП;

- серверна частина: 4 ГБ ОЗП.

Вимоги до інформаційної та програмної сумісності:

- функціонування під управлінням ОС Windows 10+;

- підтримка REST API (опційно) для зовнішніх інтеграцій;

- браузер із підтримкою HTML5;

- серверна частина: підтримка Python 3.10+, СУБД PostgreSQL або SQLite.

В.5 Виклик та завантаження

Виклик і завантаження програмного застосунку відбувається шляхом написання команди python run.py у терміналі середовище Microsoft Visual Studio Code після цього у терміналі надсилается повідомлення Running on http://127.0.0.1:5000 Відкривши це посилання у браузері, користувач отримує доступ до розгорнутого веб-застосунку, що працює у локальному середовищі.

В.6 Вхідні дані

Дані можна вводити в систему за допомогою різних пристроїв введення, таких як клавіатура та миша. Введення даних може здійснюватися шляхом ручного введення або вибору зі списків, що надаються.

В.7 Вихідні дані

Дані можна виводити з системи за допомогою пристроїв виведення, таких як монітор. Це дозволяє користувачу переглядати інформацію, яка була оброблена системою.

ДОДАТОК Г

ІНСТРУКЦІЯ КОРИСТУВАЧА

Г.1 Призначення програми

Г.1.1 Загальна характеристика ПЗ

Найменування: програмний застосунок для автоматизації формування розкладу занять в університеті «Розклад занять НУК».

Програмне забезпечення призначено для підвищення ефективності організації навчального процесу в університеті шляхом автоматизації формування розкладу занять. Застосунок дозволяє оптимізувати планування навчальних занять, уникати конфліктів при складанні розкладу, враховувати побажання викладачів, студентів і адміністрації, а також забезпечувати зручний доступ до актуальної інформації про розклад.

Розробка призначена для автоматизації процесу складання, редагування, перегляду та публікації розкладу занять в університеті. Програмна система передбачає багаторівневий доступ до функцій залежно від ролі користувача: адміністратора, викладача чи студента.

Г.1.2 Склад функцій:

- авторизація користувачів за ролями (адміністратор, викладач, студент);
- формування розкладу занять вручну та автоматично з урахуванням таких параметрів, як аудиторне навантаження, наявність викладачів, груп, аудиторій;
- адміністрування академічних груп;
- збереження та редагування онлайн-посилань на дистанційні заняття;
- експорт розкладу у форматах PDF/Excel;

- синхронізація з календарем Google.

Г.2 Умови виконання програми

Г.2.1 Вимоги до технічних засобів, необхідних для функціонування програми:

- мінімальні ресурси клієнтської машини: 2 ГБ ОЗП;
- серверна частина: 4 ГБ ОЗП.

Г.2.2 Програмне забезпечення, необхідне для функціонування програми:

- операційна система Windows 10+;
- підтримка REST API (опційно) для зовнішніх інтеграцій;
- браузер із підтримкою HTML5;
- серверна частина: підтримка Python 3.10+, СУБД PostgreSQL або SQLite.

Г.2.3 Інші умови експлуатації:

- кліматичні умови для коректної роботи програми співпадають із вимогами для технічних засобів, на яких вона буде встановлена;
- використання у локальній мережі навчального закладу або на сервері з доступом через браузер;
- програмна система розрахована на користувача із середнім рівнем комп’ютерної грамотності.

Г.3 Виконання програми

Виклик і завантаження програмного застосунку відбувається шляхом написання команди python run.py у терміналі середовище Microsoft Visual Studio Code після цього у терміналі надсилается повідомлення Running on

http://127.0.0.1:5000 Відкривши це посилання у браузері, користувач отримує доступ до розгорнутого веб-застосунку, що працює у локальному середовищі.

Першим, що побачить користувач, буде головна сторінка. Вона містить логотип системи та кнопку «Вхід» для авторизації. Під кнопкою є посилання «Зареєструйтесь», яке перенаправляє на форму реєстрації (рис. Г.1).

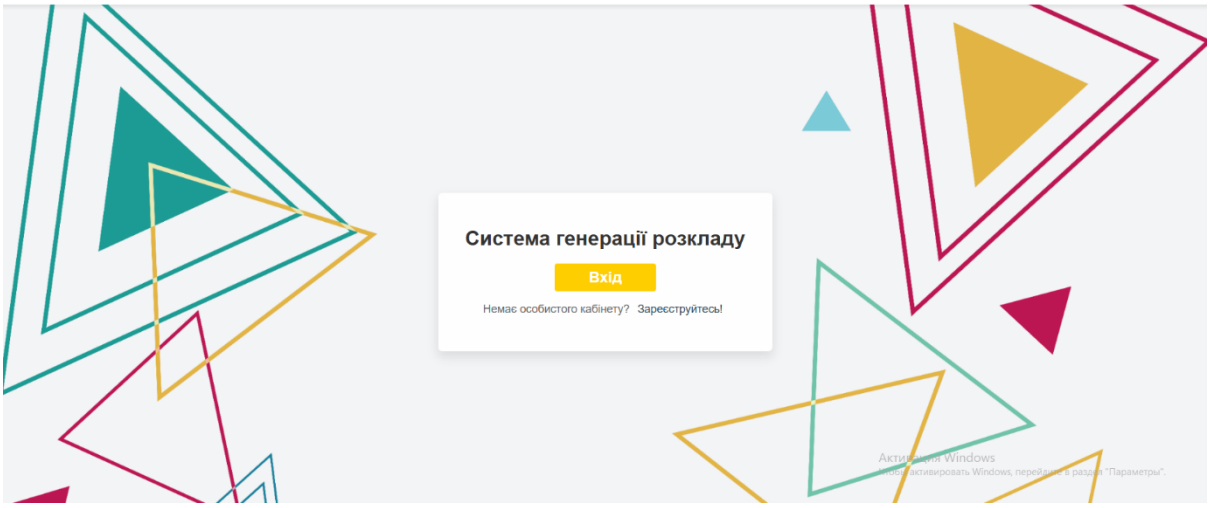


Рисунок Г.1 — Перша сторінка застосунку

Якщо користувач натисне на кнопку «Вхід», він побачить форму входу, де потрібно ввести логін та пароль для доступу до особистого кабінету (рис. Г.2).

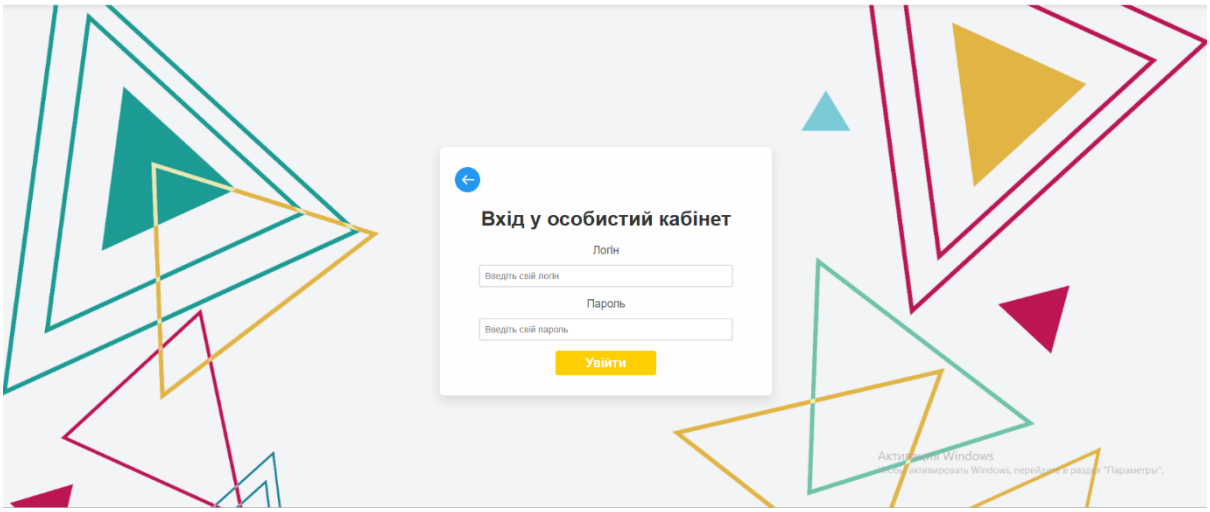


Рисунок Г.2 — Форма входу

Якщо користувач перейде по посиланню «Зареєструйтесь», він потрапить на сторінку реєстрації (рис. Г.3).

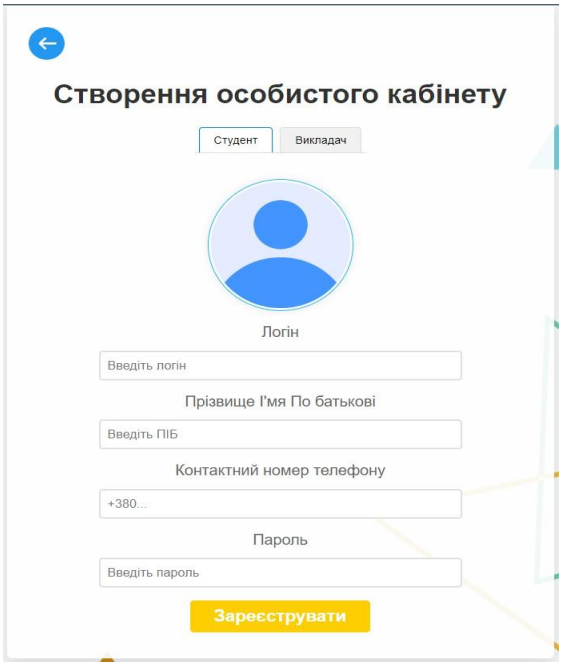


Рисунок Г.3 — Форма входу

Форма реєстрації має дві вкладки: «Студент» та «Викладач». У вкладці «Студент» користувач вводить свій логін, повне ім’я, контактний номер телефону та пароль. У вкладці «Викладач» додатково є поле з випадаючим списком, де можна обрати свою посаду (викладач, професор, доцент, асистент).

Дані повинні відповідати правилам: номер телефону повинен починатися з +, містити тільки цифри та мати довжину 12 символів без +.

Після успішної авторизації або реєстрації користувача перенаправить на основну сторінку викладача або студента в залежності від регістрації зображено на рисунках Г.4 та Г.5.

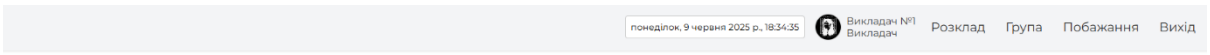
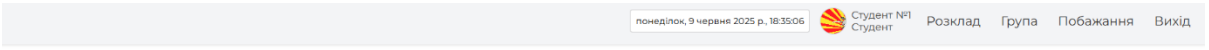


Рисунок Г.4 – Основна сторінка викладача



1

Рисунок Г.5 – Основна сторінка студента

Вхід як Студент .

На основній сторінці ми бачимо 4 кнопки

Розклад Група Побажання та Вихід

При натисканні кнопки Група відкирвается мені групи де відображается назва групи ім’я викладача та всі одногрупники зображення на рисунку Г.6.

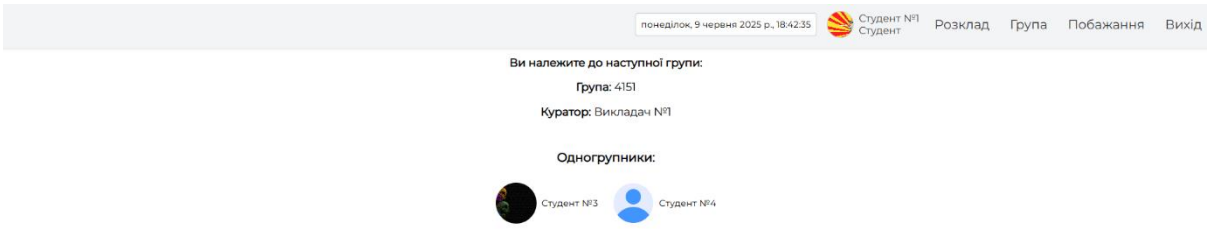


Рисунок Г.6 – Сторінка група

При натисканні кнопки Розклад користувач бачить розклад зображений на рис. Г.7

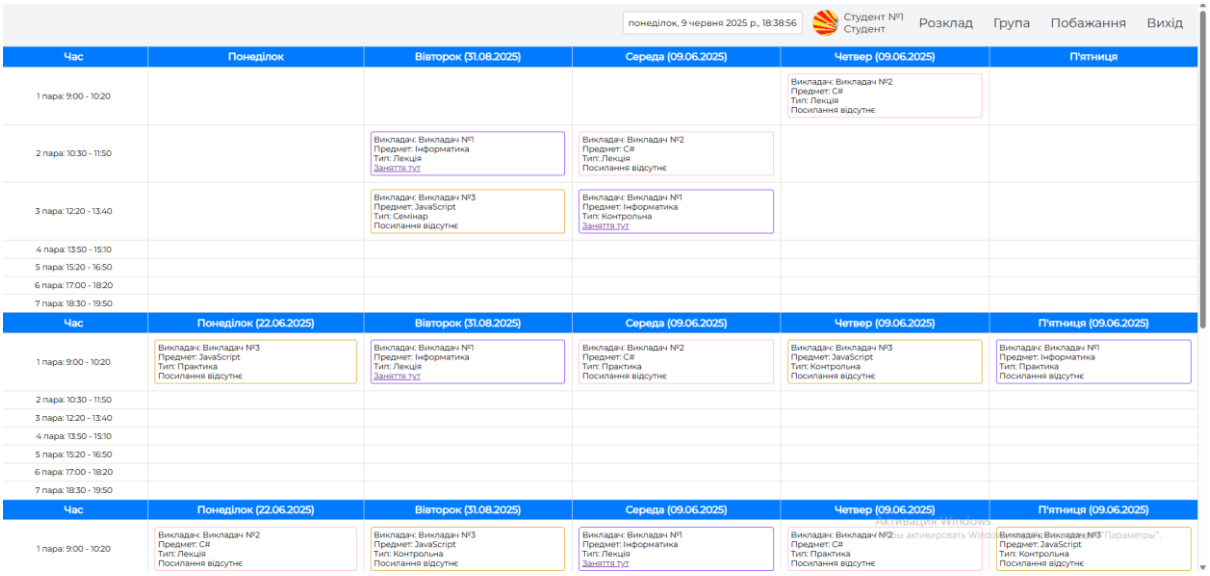


Рисунок Г.7 – Сторінка розкладу

При натисканні кнопки побажання відкривається сторінка побажання зображення на рис. Г.8

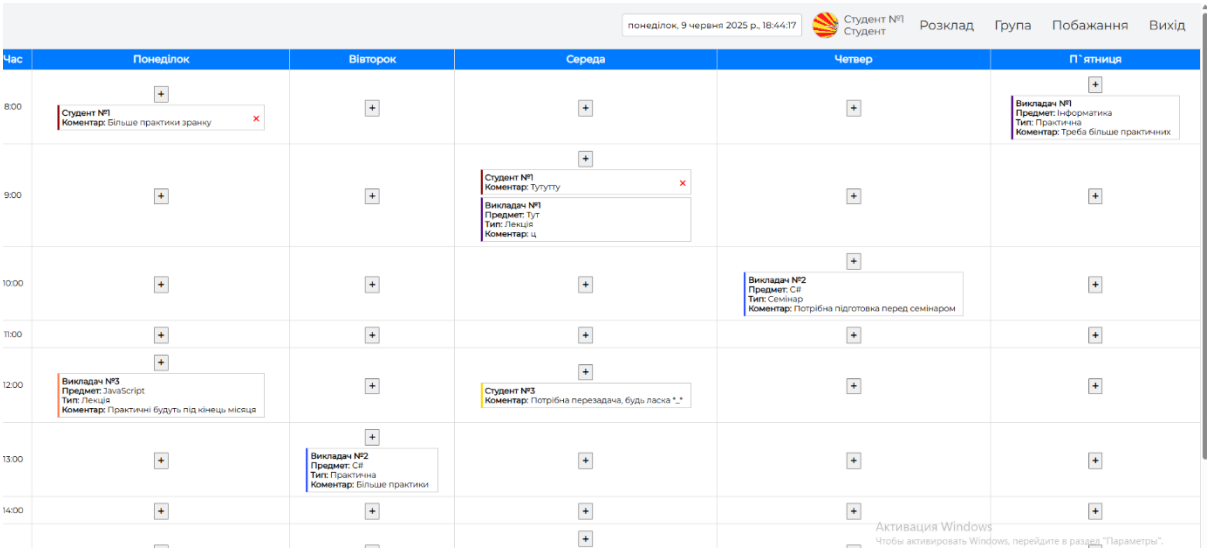


Рисунок Г.8 – Сторінка побажання

Для того щоб додати побажаня студент має натискнути на кнопку + після цього відкривается вікно де студент вводить свої побажання зображено на рис. Г.9

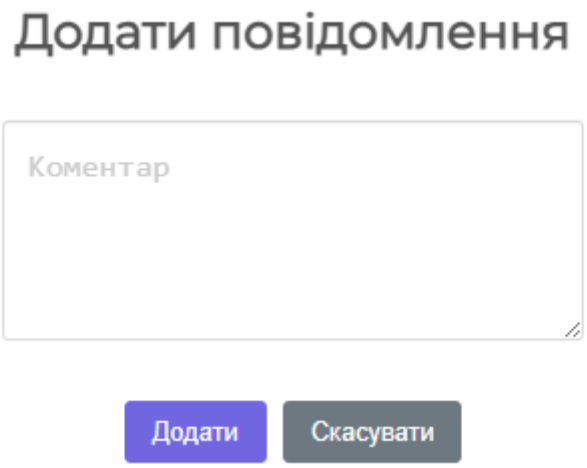


Рисунок Г.9 – Форма введення побажання

Студент за допомогою клавіатури вводить своє побажання та натискає на кнопку Додати і побажання автоматично додається і відображається у місці де студент його створив.

Авторизації як викладача

На основній сторінці ми бачимо 4 основних кнопки Розклад Група Побажання та Вихід (аналогічно як для Студента).

Перехід на сторінку розклад ззовні він повністю відповідає розкладу зі сторони Студента, але є одна відмінність, можна додати посилання до предмету викладача зображення на рис. Г.10.

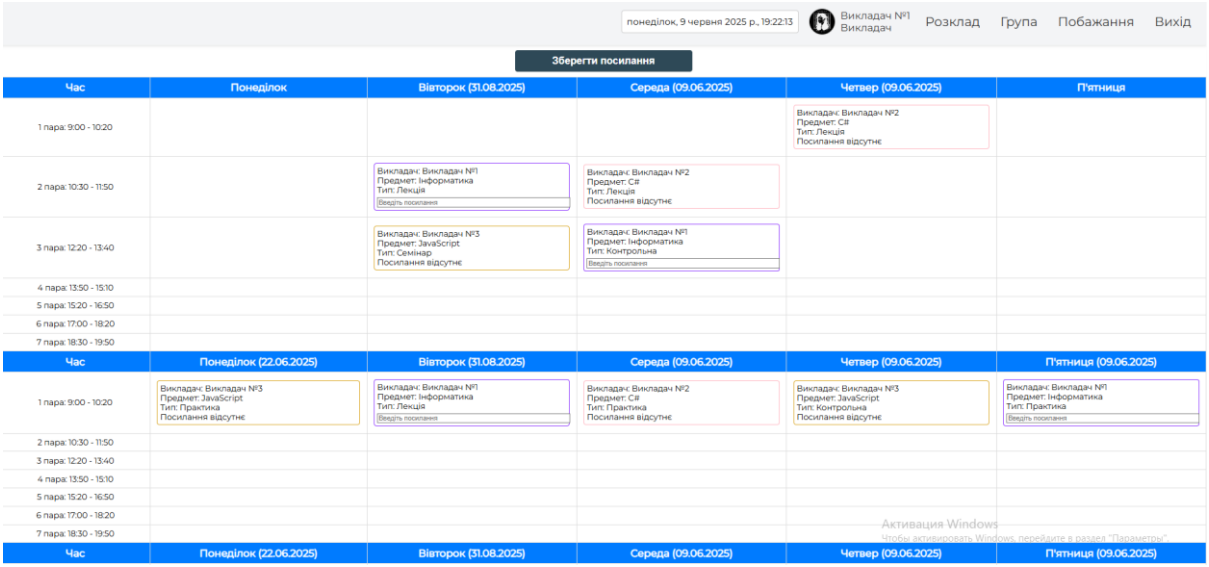


Рисунок Г.10 – Розклад зі сторони Викладача

Після додання посилання викладач має натиснути на кнопку зберегти посилання і посилання автоматично збережеться і відобразиться у студентів Коли викладач переходить на сторінку група викладач бачить усі групи якими він керує та студентів які належать до кожної з груп також викладач може додати нових студентів або видалити студентів та створити нову групу зображення на рисунку Г.11

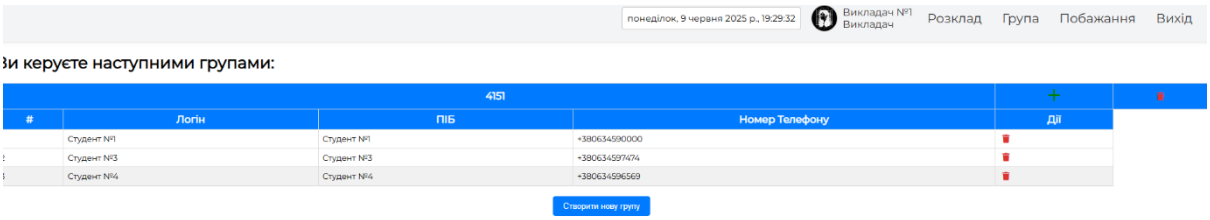


Рисунок Г.11 – Сторінка Група зі сторони викладача

Сторінка побажання працює однаково як для студентів так і для викладачів

Аміністратор.

Реєстрація для Адміністратора відсутня лише авторизація.

Інтерфейс Адміністратора, представлеий на рис. Г.12, має 4 основні кнопки Користувачі, Побажання до розкладу, Розклад та Вихід.

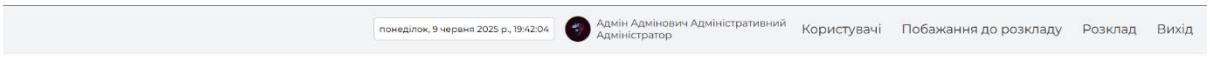


Рисунок Г.12 – Основна сторінка Адміністратора

На сторінці Користувачі У Адміністратора є функція видалити всіх користувачів як студентів так і викладачів рисунок Г.13.

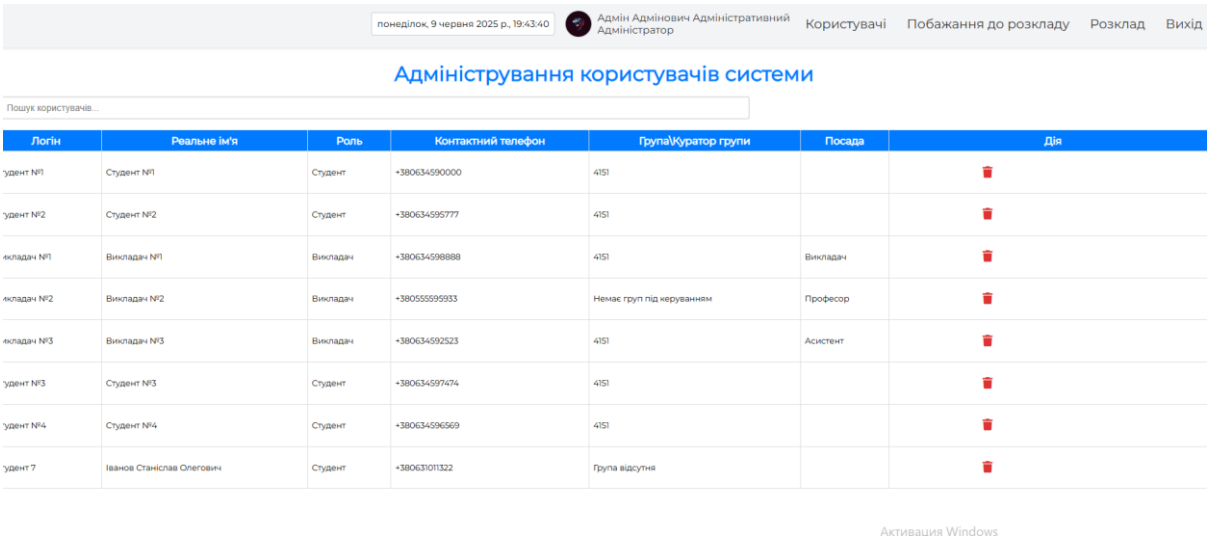


Рисунок Г.13 – Сторінка Користувачі

У вкладці Побажання до розкладу відображаються усі побажання користувачів рисунок Г.14.

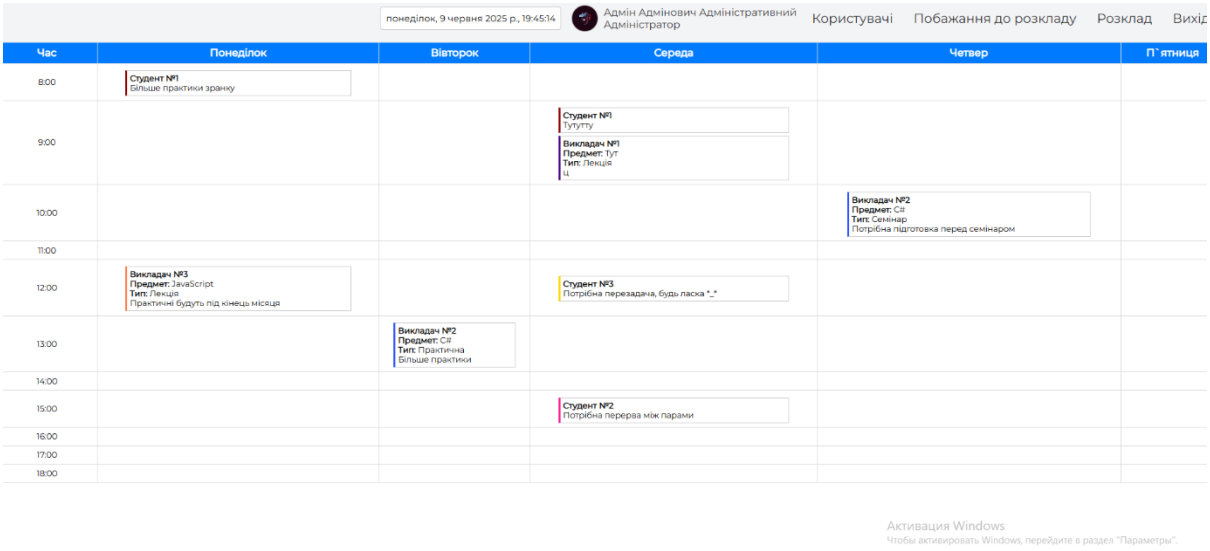


Рисунок Г.14 Сторінка Побажання до Розкладу

У вкладці Розклад є функція редагувати та генерувати розклад в залежності від потреб викладачів та студентів зображенням на Рисунку Г.15.

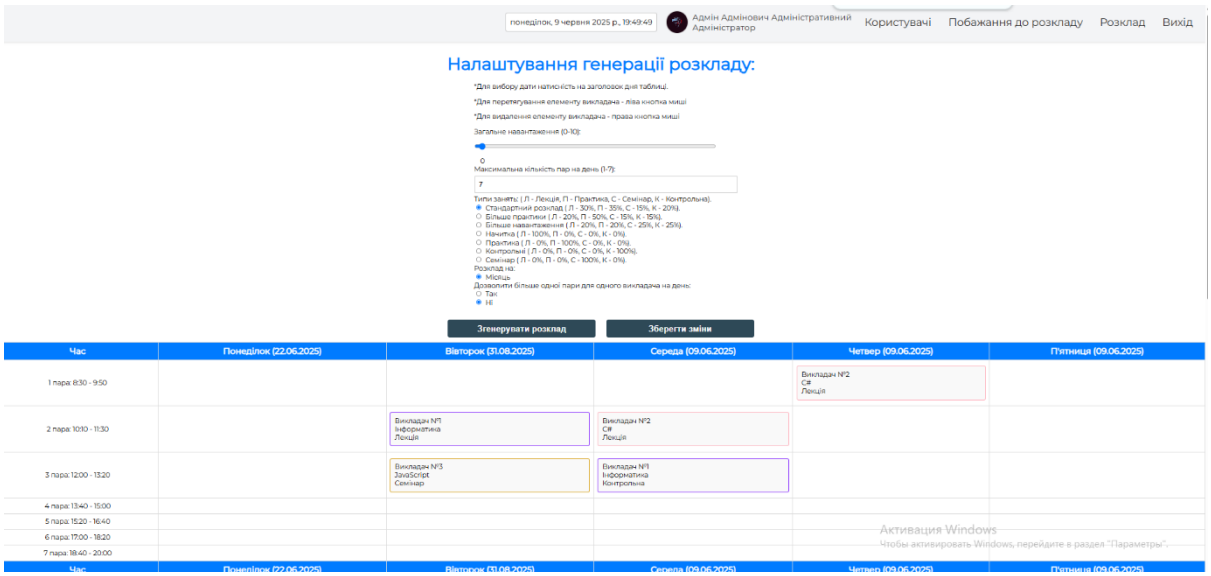


Рисунок Г.15 –Сторінка генерації розкладу

Адміністратор може змінювати графік занять за допомогою перетягування предметів на інші дати та час. Також може змінювати дати для цього потрібно натиснути на заголовок таблиці.

Для генерації розкладу потрібно обрати ступінь навантаження для цього потрібно скористатися шкалою і зупинитись на значені від 0 до 10 також обрати максимальну кількість пар на день від 1 до 7 обрати тип занять див. рис. Г.16.

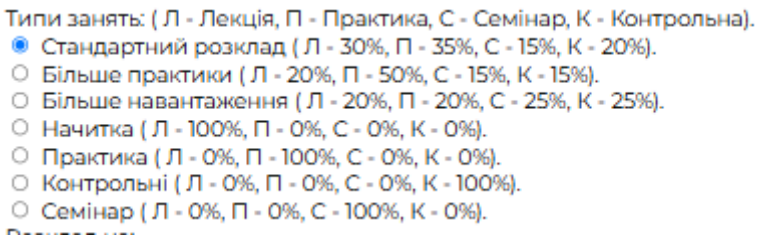


Рисунок Г.16 –Тип занять

та обрати умову кількості пар для одного викладача на день і натиснути на кнопку Згенерувати розклад. У випадку кінцевого результату далі потрібно натиснути Зберегти. Адміністратор може перегенерувати розклад або змінити його вручну

Г.4 Повідомлення оператору.

У разі некоректних даних, що вводяться користувачем, програма видає повідомлення про помилку.

- Для кожного користувача ПЗ номер телефону є унікальним, може бути лише 1 користувач з таким номером телефону. В разі реєстрації іншого користувача з уже існуючим телефонним номером, програма видає повідомлення «Користувач х таким номером вже існує» (рис. Г.17);

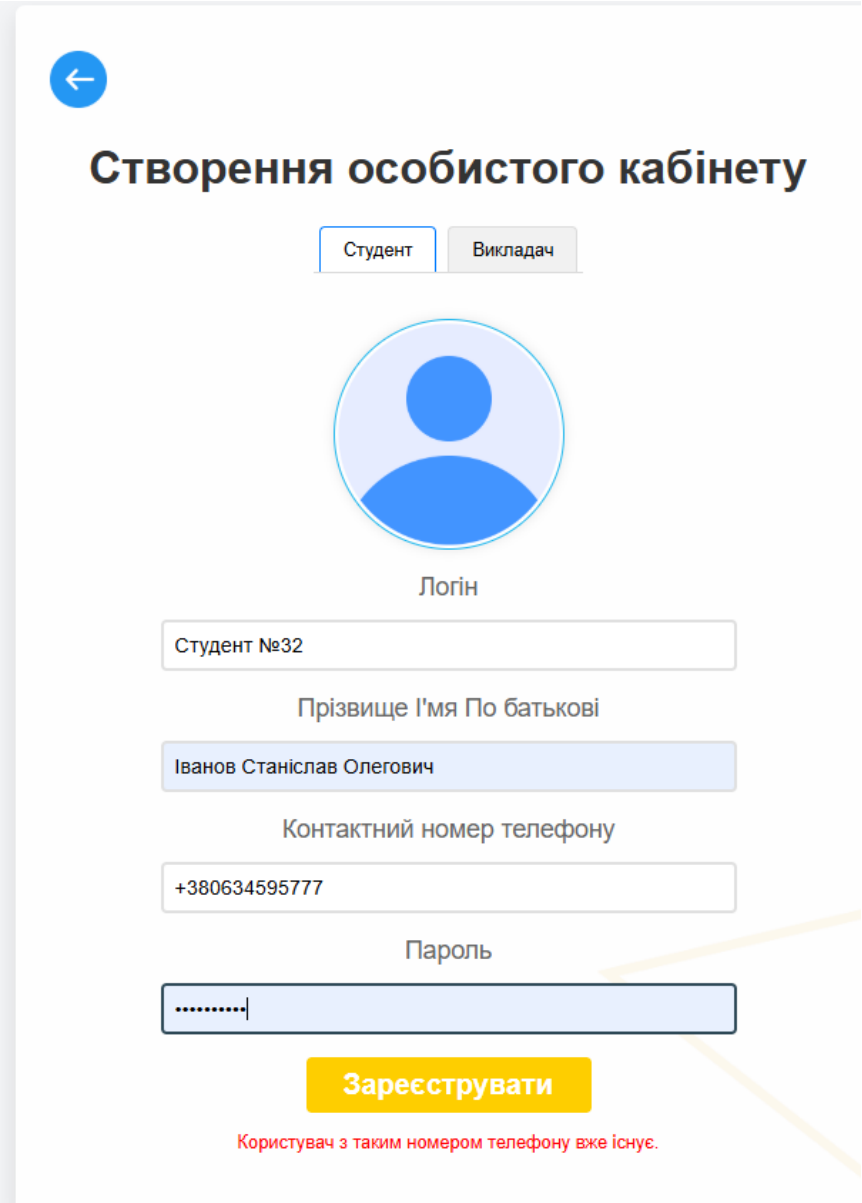


Рисунок Г.17— Повідомлення про некоректний номер телефону

- Для кожного користувача ПЗ Логін є унікальним. В разі реєстрації іншого користувача з уже Логіном, програма видає повідомлення «Користувач з таким логіном вже існує» (рис. Г.18).

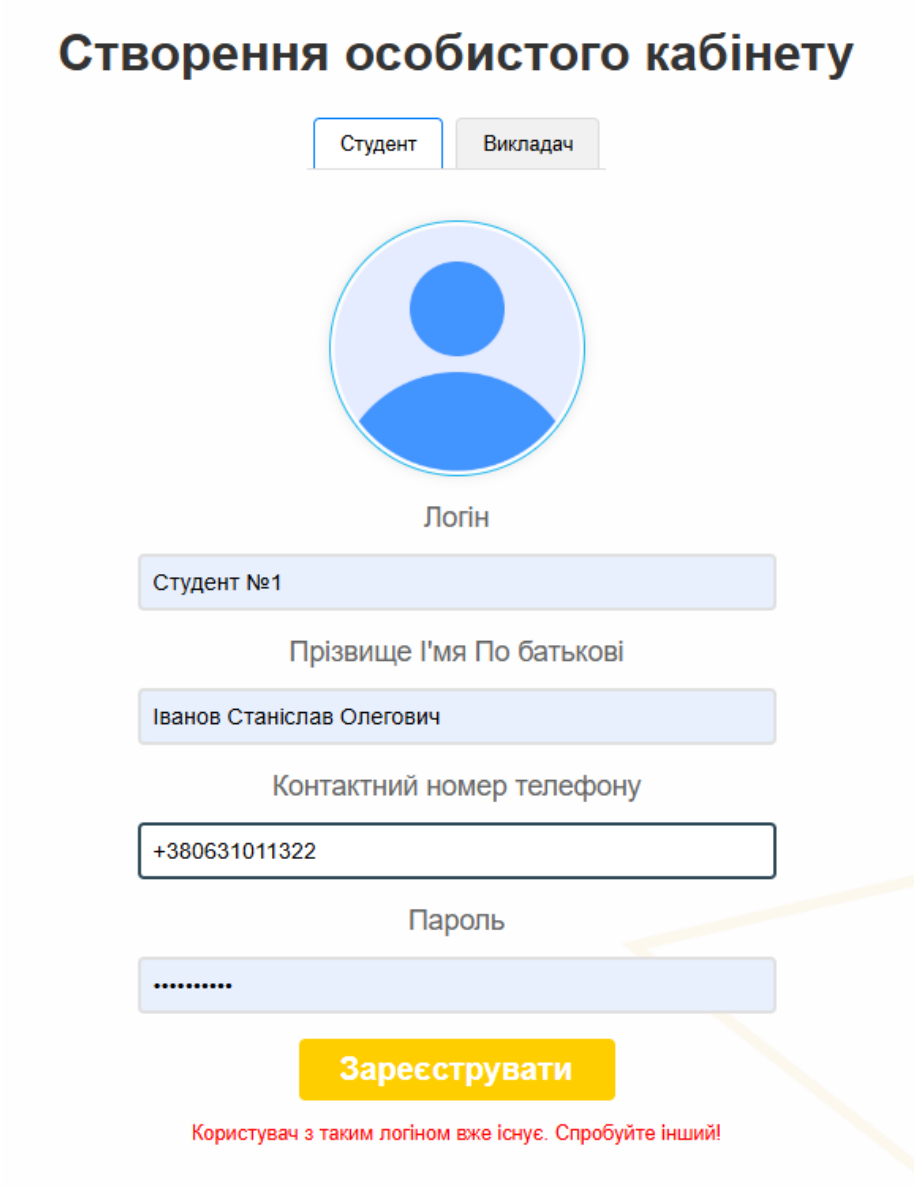


Рисунок Г.18 — Перевірка на унікальність логіну

ДОДАТОК Д

ПРОГРАМА ТА МЕТОДИКА ВИПРОБУВАНЬ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Д.1 Об'єкт випробувань

Повне найменування програми: програмного застосунку для автоматизації формування розкладу занять в університеті «Розклад занять НУК»

Коротка характеристика галузі застосування: Програмне забезпечення призначено для підвищення ефективності організації навчального процесу в університеті шляхом автоматизації формування розкладу занять. Застосунок дозволяє оптимізувати планування навчальних занять, уникати конфліктів при складанні розкладу, враховувати побажання викладачів, студентів і адміністрації, а також забезпечувати зручний доступ до актуальної інформації про розклад.

Д.2 Мета випробувань

Мета випробувань полягає в перевірці відповідності характеристик розробленого програмного застосунку функціональним та іншим вимогам, які визначені у Технічному завданні.

Д.3 Вимоги до програми

Програма повинна мати здатність виконувати наступні функції:

- авторизація користувачів за ролями (адміністратор, викладач, студент);
- формування розкладу занять вручну та автоматично з урахуванням таких параметрів, як аудиторне навантаження, наявність викладачів, груп, аудиторій;

- адміністрування академічних груп;
- збереження та редагування онлайн-посилань на дистанційні заняття;
- експорт розкладу у форматах PDF/Excel;
- синхронізація з календарем Google.

Д.4 Вимоги до програмної документації

Д.4.1 Програмна документація містить наступні матеріали:

- технічне завдання;
- вихідні тексти програмних модулів;
- опис програми;
- інструкція користувача;
- програма та методика випробувань програмного застосунку.

Д.4.2 Спеціальні вимоги

Спеціальні вимоги до програмної документації не висуваються.

Д.5 Засоби і порядок випробувань

Д.5.1 Програмні та технічні засоби випробувань

Д.5.1.1 Технічні засоби:

– Процесор: AMD Ryzen 5 5500, 3.60 GHz

– Оперативна пам’ять: 16 ГБ

– Диск: SSD 500 ГБ

– Операційна система: Windows 10 x64

Д.5.1.2 Програмні засоби:

– Python 3.11

– Flask 2.x

– SQLite 3

– Google Chrome 108+

Д.5.2 Порядок проведення випробувань

Випробування програмного застосунку для автоматизації формування розкладу занять в університеті «Розклад занять НУК» включає в себе аналіз вимог до функціональних характеристик програмного застосунку та перевірку вимог до документації.

Д.6 Методи випробувань

Д.6.1 Методика проведення перевірки вимог до програмної документації

Під час перевірки вимог до програмної документації необхідно перевірити

- наявність всіх необхідних документів і їх відповідність вимогам до змісту і формату;

- відповідність документів (див п.Д.4.1) вимогам стандарту та ін. нормативам.

Д.6.2 Методика проведення перевірки вимог до функціональних характеристик програмного продукту

Під час випробування програмного застосунку для автоматизації формування розкладу занять в університеті «Розклад занять НУК» провести наступні тести (див. табл.Д.1) для перевірки вимог до функціональних характеристик:

Таблиця Д.1 – тести для перевірки вимог до функціональних характеристик ПЗ

№	Дія	Очікуваний результат
1	2	3
1	Реєстрація нового користувача	Користувача створено в БД
2	Вхід користувача	Авторизація виконана

Кінець таблиці Д.1

1	2	3
3	Створення групи	Групу збережено
4	Додавання посилання	Посилання з’явилася у розкладі
5	Генерація розкладу	Відображення за часом і днями
6	Призначення до групи	Користувач доданий до групи
7	Перегляд розкладу	Відображення розкладу
8	Видалення користувача	Видалення користувача з БД
9	Видалення групи	Видалнення з БД
10	Створення побажання	Відображення побажання

Очікувані результати використовуються як еталон для оцінювання поведінки програмної системи під час її випробувань.