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.

«___» _____ 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти «бакалавр»

на тему: Розробка програмного забезпечення для обліку комп'ютерної техніки Мішково-Погорілівської загальноосвітньої санаторної школи-інтернату

Виконав: студент групи 4151

_____ Іванов Д.О.

(підпис, ПІБ)

Керівник роботи:

доцент кафедри. ПЗАС, к.т.н., доцент

(посада, науковий ступень вчене звання)

_____ Макарова Л.М.

(підпис, ПІБ)

Миколаїв – 2023 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)
«___» _____ 2023 р.

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

Студенту Іванову Денису Олексійовичу
(Прізвище, ім'я, по батькові)

1. Тема роботи: Розробка програмного забезпечення для обліку комп'ютерної техніки Мішково-Погорілівської загальноосвітньої санаторної школи-інтернату
Керівник роботи Макарова Л.М.
Затверджені наказом ректора №257-уч від «17» березня 2023 р.
2. Термін подання роботи: 29.05.2023 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
Титульний слайд, Мета та завдання кваліфікаційної роботи, Аналіз вимог до ПЗ, Архітектура ПЗ, Діаграма варіантів використання, Сценарії варіантів використання, Концептуальна модель даних, Діаграма класів, Діаграми взаємодії, Логічна модель бази даних, Технічний проект ПЗ, Результати розробки, Висновки, 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 20.03.2023 р.

К А Л Е Н Д А Р Н И Й П Л А Н

Номер	Назва етапів роботи	Терміни виконання	Примітка
1.	Підготовка розділу ВСТУП	24.03.2023	ПП*
2.	Аналіз практичної задачі інженерії ПЗ	31.03.2023	ПП*
3.	Аналіз вимог до ПЗ	07.04.2023	ПП*
4.	Розробка архітектури ПЗ	21.04.2023	
5.	Розробка проекту ПЗ	05.05.2023	
6.	Реалізація та тестування ПЗ	12.05.2023	
7.	Підготовка розділу Результати розробки ПЗ	17.05.2023	
8.	Підготовка розділу з охорони праці	19.05.2023	ПП*
9.	Підготовка розділу ВИСНОВКИ	24.05.2023	
10.	Оформлення списку використаних джерел та додатків	26.05.2023	
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.)	29.05.2023	не пізніше, ніж за 14 діб до захисту (згідно п.4.1 зазначеного Порядку)
12.	Підготовка презентації та доповіді	02.06.2023	
13.	Попередній захист роботи на засіданні кафедри ПЗАС	05.06.2023	
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	12.06.2023	

* - за результатами переддипломної практики (ПП), яка була з 01.02.2023 до 19.03.2023 р.

Студент _____ Іванов Д.О.
(підпис) (ПІБ)

Керівник роботи _____ Макарова Л.М.
(підпис) (ПІБ)

АНОТАЦІЯ

Кваліфікаційна робота присвячена розв’язанню практичної задачі інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов – розробці програмного забезпечення для обліку комп’ютерної техніки Мішково-Погорілівської загальноосвітньої санаторної школи-інтернату.

У кваліфікаційній роботі представлені аналіз практичної задачі інженерії програмного забезпечення за темою кваліфікаційної роботи, аналіз існуючих аналогів та постановка задачі на розробку програмного забезпечення. Розроблено ескізний, технічний та робочий проекти програмного забезпечення, представлено результати розробки та розділ з охорони праці.

Кваліфікаційна робота викладена на 79 сторінках машинописного тексту та містить: 12 рисунків, 20 таблиць, 5 додатків, список використаних джерел з 20 найменувань.

ABSTRACT

The qualification work is dedicated to solving the practical task of software engineering, characterized by complexity and uncertainty of conditions – software development to account computers of Mishkovo-Pogorilove secondary sanatorium boarding school.

In qualification work presents analysis of the practical task of software engineering on the topic of qualification work, analysis of existing analogues and formulation of the problem for software development. The sketch, technical and working projects of the software, results of development and the section on labor protection are developed.

The qualification work is presented on 79 pages of typewritten text and contains: 12 figures, 20 tables, 5 appendices, list of references from 20 names.

ЗМІСТ

ВСТУП

Комп`ютерна техніка вже давно увійшла до повсякденного шкільного життя. Спеціалізовані освітні програми, різноманітні застосунки значно полегшують вчителям організацію та проведення навчального процесу, а школярам допомагають отримувати знання звичним та цікавим способом [1]. В деяких шкільних програмах вивчення інформатики починається вже з першого класу [2]. Все це потребує наявності у школах як спеціалізованих комп`ютерних класів з відповідним обладнанням, так і мобільних пристроїв, наприклад, планшетів. До того ж, ряд дій, напряму не пов`язаних з викладанням інфоматики, також можуть бути вирішені з використанням комп`ютерної техніки, зокрема, демонстрація навчальних презентацій та фільмів, перевірка знань учнів. За допомогою комп`ютерної техніки також можливо покращити комунікації для педагогічного колективу, ведення обліку книг у шкільній бібліотеці, підготовку друкованих матеріалів для різноманітних шкільних заходів та багато іншого [3].

Комп`ютерна техніка зазвичай укомплектована операційними системами та пакетами прикладних програм. Обліку підлягають операції інвентаризації як самого обладнання, так і встановленого програмного забезпечення.

Згідно графіка навчального процесу в спеціалізованому комп`ютерному класі проводяться заняття з інформатики для учнів різних класів. За вимогами організації навчального процесу для кожного класу та кожної роботи повинні існувати інструктивно-методичні матеріали та відповідне програмне забезпечення. Окрім того, комп’ютерна техніка повинна підтримуватися у робочому стані. З цією метою є графік профілактичних робіт, виконання яких вимагає певного часу та інколи наявності витратних матеріалів та комплектуючих.

З метою оптимізації роботи комп’ютерної техніки виникає необхідність у автоматизації обліку як безпосередньо комп’ютерної техніки та програмного

забезпечення, так і інструктивно-методичних матеріалів та використання комп'ютерного часу.

Автоматизація інвентаризації комп`ютерної техніки – це процес, який повинен проводитися дуже обережно. По-перше, це вибір та установка системи обробки даних про комп`ютерну техніку школи. По-друге, це ще і питання розвитку та адаптації навчального процесу до потреб сьогодення з метою найбільшого використання переваг, яке надає використання комп`ютерної техніки. Автоматизація інвентаризації комп`ютерної техніки проводиться також з метою отримання достовірної і актуальної інформації.

Кваліфікаційна робота присвячена розв'язанню практичної задачі інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов, а саме розробці програмного забезпечення для обліку комп'ютерної техніки Мішково-Погорілівської загальноосвітньої санаторної школи-інтернату.

У кваліфікаційній роботі потрібно виконати аналіз практичної задачі інженерії програмного забезпечення: розглянути питання автоматизації обліку комп'ютерної техніки, пакетів прикладних програм та інструктивно-методичних матеріалів, виконати постановку задачі на розробку програмного забезпечення та розробити програмне забезпечення для обліку комп'ютерної техніки Мішково-Погорілівської загальноосвітньої санаторної школи-інтернату, для чого виконати наступні завдання: провести аналіз вимог до програмного забезпечення, розробити архітектуру програмного забезпечення, розробити проєкт програмного забезпечення, виконати реалізацію та тестування програмного забезпечення.

Об'єктом, який розглядається в кваліфікаційній роботі, є процес розробки програмного забезпечення для обліку комп'ютерної техніки Мішково-Погорілівської загальноосвітньої санаторної школи-інтернату.

Впровадження розробки забезпечить отримання необхідної інформації щодо комп'ютерної техніки за запитами користувачів та оперативне отримання аналітичних звітів.

1 АНАЛІЗ ПРАКТИЧНОЇ ЗАДАЧІ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ПОСТАНОВКА ЗАДАЧІ НА РОЗРОБКУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОБЛІКУ КОМП'ЮТЕРНОЇ ТЕХНІКИ МІШКОВО-ПОГОРІЛІВСЬКОЇ ЗАГАЛЬНООСВІТНЬОЇ САНАТОРНОЇ ШКОЛИ-ІНТЕРНАТУ

1.1 Аналіз практичної задачі інженерії програмного забезпечення

Розробці програмного забезпечення для обліку комп'ютерної техніки Мішково-Погорілівської загальноосвітньої санаторної школи-інтернату передує етап аналізу практичної задачі інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов, та включає в себе різні аспекти, пов`язані з розробкою програмного забезпечення: вимоги до функціональних характеристик, вимоги до організаційних вхідних та вихідних даних, вимоги до надійності і т.д.

Як було сказано вище, інвентаризація комп`ютерної техніки у школі – це процес, який повинен проводитися ретельно та обережно. Основна складність постає у тому, що комп`ютерна техніка у школі призначена для декількох цілей – як навчальних, так і допоміжних, відповідно і облік та інвентаризація будуть дещо відрізнятися.

Зазвичай, для забезпечення організації роботи комп'ютерного класу призначається завідувач комп'ютерного класу з числа викладачів, які проводять заняття у комп'ютерному класі. Завідувач комп'ютерного класу є організатором обладнання класу, роботи викладачів та школярів по застосуванню комп`ютерної техніки. Завідувач комп'ютерного класу забезпечує використання класу у відповідності з навчальним планом школи, розробляє перспективний план обладнання класу, приймає заходи щодо його дообладнання та поповнення інструктивно-методичними матеріалами і технічними засобами навчання згідно з прийнятим переліком, несе

відповідальність за збереження наявного в класі обладнання. Завідувач комп'ютерного класу несе відповідальність за ведення журналу інвентаризації, утримання комп`ютерної техніки у постійній готовності до застосування, своєчасність і ретельність профілактичного технічного обслуговування комп`ютерної техніки, за підтримання у комп'ютерному класі санітарно-гігієнічних вимог та вимог техніки безпеки.

Завідувач комп'ютерного класу бере участь у плануванні завантаження комп'ютерного класу навчальними, факультативними та іншими заняттями зі школярами; всі види занять у комп'ютерному класі проводяться за обов'язкової присутності викладача.

Завідувач комп'ютерного класу несе відповідальність за своєчасне проведення первинного та періодичного інструктажу з техніки безпеки, які проводяться, як правило, викладачами, які проводять заняття в комп'ютерному класі. Всі відомості з проведення інструктажу школярів заносяться в спеціальний журнал реєстрації інструктажу по техніці безпеки.

Також допомогу в роботі завідувачу комп'ютерного класу може надавати лаборант, який знаходиться в безпосередньому підпорядкуванні завідуючого комп'ютерного класу і звітує перед ним за збереження, правильне зберігання і використання комп`ютерної техніки. За планом викладача і під його керівництвом лаборант готує комп`ютерну техніку до заняття. Лаборант забезпечує дотримання школярами правил техніки безпеки, постійну готовність протипожежних засобів і засобів першої допомоги, реєструє відмови комп`ютерної техніки під час занять, а також проводить дрібний ремонт комп`ютерної техніки, що вийшла з ладу.

Згідно розкладу навчальних занять формується графік проведення занять в комп'ютерному класі. Ведеться облік виконання робіт в комп'ютерному класі за день. Дані обліку аналізуються керівництвом школи з метою підвищення рівня організації навчального процесу.

Крім того, завідувач комп'ютерного класу або лаборант можуть відповідати також за збереження і правильне використання іншої комп`ютерної

техніки, наявної у школі, окрім обладнання комп`ютерного класу, своєчасність і ретельність профілактичного технічного обслуговування цієї комп`ютерної техніки та її дрібний ремонт.

1.2 Аналіз існуючих програмних продуктів

Існує ряд програмних продуктів для обліку та інвентаризації комп’ютерної техніки, які мають багато можливостей і функцій, але вони не задовольняють більшості вимог до розроблюваного програмного забезпечення.

Одним з таких програмних продуктів є «Hardware Inspector» [4]. Даний програмний продукт є інструментом автоматизації інвентаризації та зручного обліку комп'ютерної техніки або іншого обладнання в організаціях.

З його допомогою є можливість вчасно отримувати всю необхідну інформацію, а також формувати різноманітні звіти, планувати ремонт, обслуговування та оновлення комп’ютерної техніки. Програмний продукт має можливість вести повну історію всіх змін, що відбуваються з підзвітною комп’ютерною технікою.

Даний програмний продукт має такі функціональні можливості:

– можливість обліку робочих місць і деталізації окремих пристроїв;

– можливість вести по кожному пристрою історію ремонту, а також інших робіт обслуговування;

– можливість ручного або автоматичного заповнення бази даних;

– можливість використання великого набору звітів;

– можливість обліку інвентарних номерів;

– можливість працювати в мережі.

Основними перевагами даного програмного продукту є:

– можливість використовувати необмежену кількість баз даних;

– можливість використання механізму пошуку;

– можливість налаштування інтерфейсу.

Однак даний програмний продукт є платним, тому не задовольняє вимогам до розроблюваного програмного забезпечення.

Аналогами цього програмного продукту є Network Inventory Advisor [5] та HPE OpenView [6]. Але вони також мають тільки платні версії.

Серед безкоштовних програм-аналогів можна відмітити OCS Invetory [7]. Цей програмний продукт дозволяє:

– автоматично збиратися в мережі та систематизувати дані по апаратному забезпеченню комп'ютерних систем (як робочих станцій, так і серверів);

– переглядати та систематизувати отриману інформацію, складати звіти;

– підтримує різні версії ОС (Windows 7/2008, Windows 8/8.1/2012, Windows 10, Mac OS, Linux);

– має мультимовну підтримку.

Але він також не задовольняє вимогам до розроблюваного програмного забезпечення в частині складності налаштування та підтримки його роботи.

Отже, більшість розглянутих існуючих програмних продуктів або мають тільки платні версії, або мають обмежений набір функцій та складні в налаштуванні. Тому розробка програмного забезпечення для обліку комп’ютерної техніки Мішково-Погорілівської загальноосвітньої санаторної школи-інтернату є актуальною.

1.3 Постановка задачі на розробку програмного забезпечення для обліку комп’ютерної техніки Мішково-Погорілівської загальноосвітньої санаторної школи-інтернату

Виходячи з попередньо проведеного аналізу та існуючих програмних продуктів, була виконана постановка задачі на розробку програмного забезпечення для обліку комп’ютерної техніки Мішково-Погорілівської загальноосвітньої санаторної школи-інтернату. Розробка є доцільною, програмне забезпечення повинно забезпечити автоматизацію обліку

комп’ютерної техніки, її програмного забезпечення, інструктивно-методичних матеріалів та використання комп’ютерного часу.

Експлуатаційним призначенням є зберігання, перегляд, пошук, редагування та видалення інформації про комп’ютерну техніку, її програмне забезпечення, інструктивно-методичні матеріали, використання комп’ютерного часу, а також інвентаризації комп’ютерної техніки у базі даних.

Функціональним призначенням є автоматизація обліку комп’ютерної техніки Мішково-Погорілівської загальноосвітньої санаторної школи-інтернату, яка покращить та полегшить обробку потрібної інформації.

Вимоги до функціональних характеристик

Програмне забезпечення повинно забезпечувати виконання наступних функцій:

– авторизація користувача;

– введення, перегляд та редагування даних про комп’ютерну техніку;

– введення, перегляд та редагування даних про програмне забезпечення;

– введення, перегляд та редагування даних про інструктивно-методичні матеріали;

– введення, перегляд та редагування даних про використання комп’ютерного часу;

– формування інвентаризаційних відомостей;

– формуванні звітів.

Вимоги до організації вхідних та вихідних даних

Введення даних здійснюється за допомогою клавіатури та миши. Дані вводяться вручну або обираються з випадаючих списків.

Обмін інформацією між програмним забезпеченням і базою даних відбувається за допомогою файлів з довільним доступом типу .mdb.

Виведення даних здійснюється за допомогою пристроїв виведення даних (монітор та принтер).

Вимоги до складу та параметрів технічних засобів

Система повинна задовольняти вказаним вимогам на комп’ютері наступної мінімальної комплекції:

– CPU: Intel(R) i3;

– RAM: 4 GB;

– HDD: 10 GB вільного місця.

Вимоги до інформаційної та програмної сумісності

Для повноцінного функціонування системи необхідно використовувати ОС Windows версії не нижче 7.

Для роботи системи необхідно СУБД Access не нижче ніж Access 2013.

2 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОБЛІКУ КОМП'ЮТЕРНОЇ ТЕХНІКИ МІШКОВО-ПОГОРІЛІВСЬКОЇ ЗАГАЛЬНООСВІТНЬОЇ САНАТОРНОЇ ШКОЛИ-ІНТЕРНАТУ

2.1 Аналіз вимог до програмного забезпечення

2.1.1 Побудова моделі варіантів використання

Будь-яке програмне забезпечення має на меті допомогу користувачу у виконанні його повсякденних задач, для чого в першу чергу визначають вимоги до розроблюваного програмного забезпечення. Однак спроба сформулювати такі вимоги не завжди буде вдалою з першої спроби, особливо, якщо майбутні користувачі програмного забезпечення не мають такого досвіду.

Щоб полегшити задачу формулювання вимог до майбутнього програмного забезпечення можна використовувати ряд UML-діаграм, зокрема, почати можна з UseCase-діаграми, або діаграми варіантів використання, яку ще називають діаграмою прецедентів. Варіанти використання це – опис послідовності дій, які може здійснювати програмне забезпечення у відповідь на зовнішні дії користувачів або інших програм. Варіанти використання відображають функціональність програмного забезпечення [8].

Діаграма варіантів використання описує функціональне призначення програмного забезпечення, тобто те, що програма повинна робити. Діаграми варіантів використання розроблюються для того, щоб:

– визначити загальні кордони та предметну галузь майбутнього програмного забезпечення;

– визначити функціональні вимоги до проектованого програмного забезпечення;

– розробити початкову концептуальну модель майбутнього програмного забезпечення;

– підготувати початкову документацію для взаємодії між всіма учасниками розробки програмного забезпечення.

Діаграма варіантів використання програмного забезпечення для обліку комп’ютерної техніки Мішково-Погорілівської загальноосвітньої санаторної школи-інтернату наведена на рисунку 2.1.

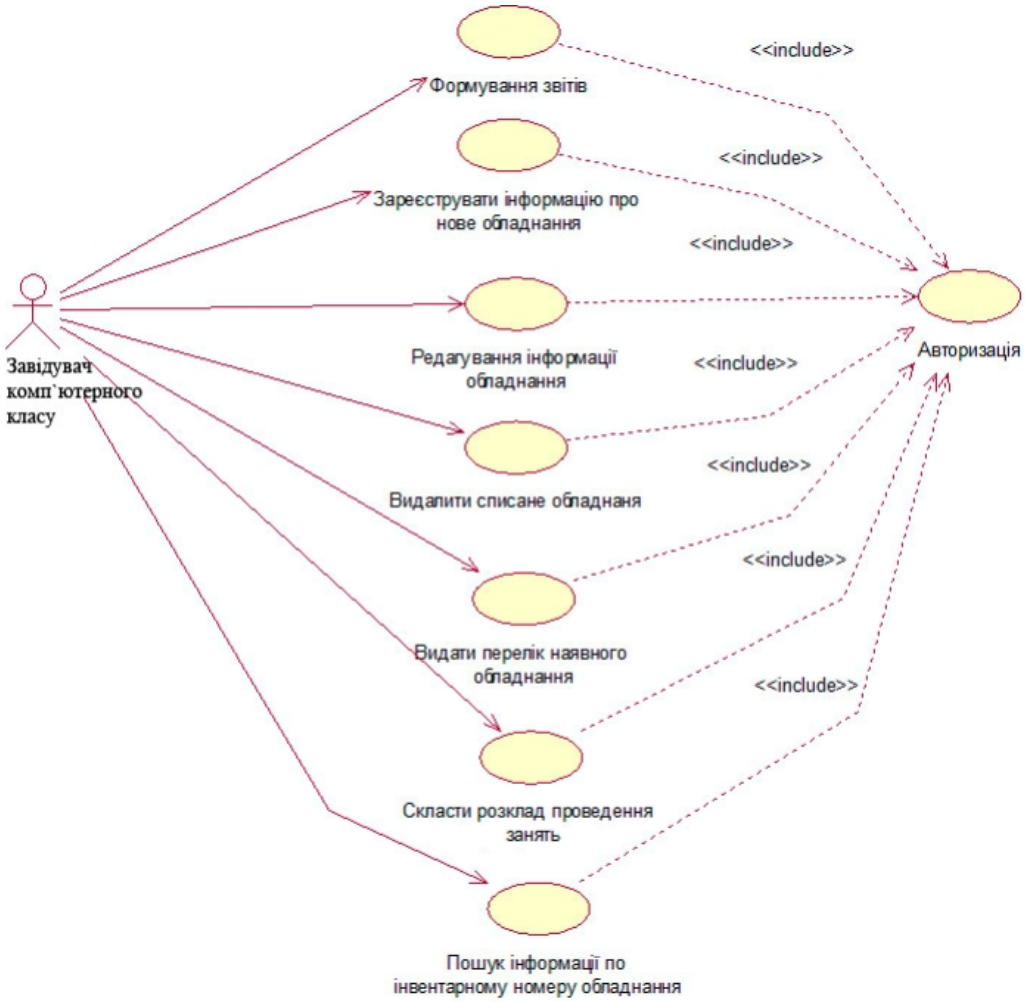

Рисунок 2.1 – Діаграма варіантів використання програмного забезпечення для обліку комп’ютерної техніки Мішково-Погорілівської загальноосвітньої санаторної школи-інтернату

Основним користувачем програмного забезпечення є: завідувач комп`ютерного класу. Діаграма варіантів використання є основою для визначення архітектури програмного забезпечення та інформаційних потоків.

2.1.2 Специфікація варіантів використання

Розробимо специфікації основних варіантів використання програмного забезпечення для обліку комп’ютерної техніки Мішково-Погорілівської загальноосвітньої санаторної школи-інтернату, деякі з них представлені у таблицях 2.1 – 2.4.

Таблиця 2.1 – Специфікація варіанту використання «Авторизація»

Складова	Опис
Короткий опис	Даний варіант використання дозволяє дійовій особі пройти перевірку доступу.
Учасники	Завідувач комп`ютерного класу
Стартова точка	Запуск програми
Передумови	Користувач повинен мати ідентифікатор (ім’я та пароль) для доступу в систему
Основний потік подій	Користувач вводить ім’я та пароль. Якщо данні вірні система ініціює вхід, інакше систем повідомляє про введення невірного паролю; дійовій особі пропонується можливість повторити введення або завершити варіант використання.
Альтернативний потік подій	–
Постумови	Перехід до головного меню

Таблиця 2.2 – Специфікація варіанту використання «Зареєструвати інформацію про нове обладнання»

Складова	Опис
1	2
Короткий опис	Даний варіант використання дозволяє занести в БД інформацію щодо нового обладнання
Учасники	Завідувач комп`ютерного класу
Стартова точка	Обрано довідник «Технічні характеристики»
Передумови	Виконання варіанта використання «Авторизація».

Продовження таблиці 2.2

1	2
Основний потік подій	1. Система переходить до головного меню; 2. Користувач обирає кнопку «Довідники»; 3. Користувач обирає кнопку «Технічні характеристики»; 4. Для додання інформації про обладнання користувач повинен ввести дані у відповідні поля на формі; 5. Натиснути кнопку «Додати». При некоректно введених даних: 1. Виводиться повідомлення про помилку. 2. Виділяються поля з невірно введеними даними 3. Інформація не заноситься до бази даних
Альтернативний потік подій	–
Постумови	Дані про обладнання заносяться до БД.

Таблиця 2.3 – Специфікація варіанту використання «Формування звітів»

Складова	Опис
Короткий опис	Даний варіант використання дозволяє користувачу працювати зі звітами «Технічні характеристики ПК», «Розклад проведення занять», «Дані комп`ютерного класу»
Учасники	Завідувач комп`ютерного класу
Стартова точка	Обрано пункт «Звіти»
Передумови	Виконання варіанта використання «Авторизація».
Основний потік подій	1. Система переходить до головного меню; 2. Користувач обирає кнопку «Звіти» 3. Користувач обирає потрібний звіт. 4. Система відкриває форму звіту. 5. Користувач вводить параметри відбору даних. 6. Система формує обраний звіт. При не заповненні обов’язкових полів, відбувається виконання альтернативного потоку.
Альтернативний потік подій	Система пропонує ввести дані знову або звіт не буде сформовано.
Постумови	Не визначені

Таблиця 2.4 – Специфікація варіанту використання «Пошук інформації по інвентарному номеру обладнання»

Складова	Опис
Короткий опис	Даний варіант використання дозволяє користувачу отримати інформацію по певному комп'ютеру.
Учасники	Завідувач комп`ютерного класу
Стартова точка	Обрано пункт «Запити»
Передумови	Виконання варіанта використання «Авторизація».
Основний потік подій	1. Користувач обирає пункт меню «Запити» 2. Користувач вводить певний інвентарний номер. 3. Користувач отримує потрібну іноформацію . Якщо користувач не ввів інвентарний номер, виконується альтернативний потік.
Альтернативний потік подій	Користувач переходить до головного меню.
Постумови	Не визначені.

Таким чином, у наведених таблицях представлені специфікації деяких варіантів використання програмного забезпечення для обліку комп'ютерної техніки Мішково-Погорілівської загальноосвітньої санаторної школи-інтернату, які використовуються при подальшому проєктуванні програмного забезпечення.

2.2 Архітектура програмного забезпечення

Архітектура програмного забезпечення – це його базова організація, реалізована через компоненти програмного забезпечення та їх відносини. Вона демонструє принципи, що покладені при проектувані програмного забезпечення, а також напрямок майбутнього розвитку програмного забезпечення [9].

Можна сформулювати наступну метамодель для визначення архітектури програмного забезпечення для обліку комп'ютерної техніки Мішково-

Погорілівської загальноосвітньої санаторної школи-інтернату, яка наведена на рисунку 2.2.

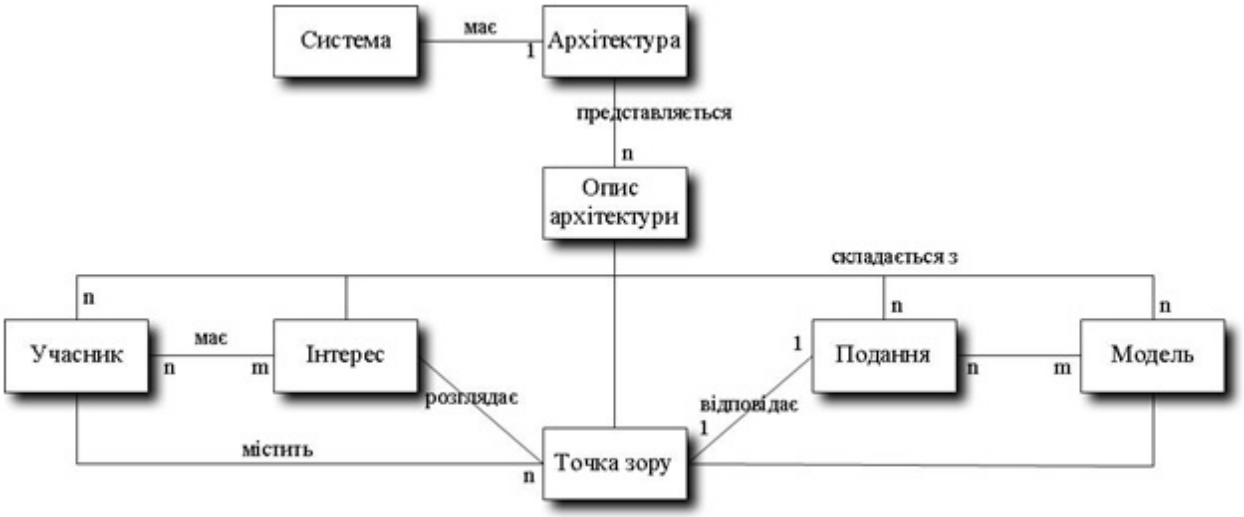

Рисунок 2.2 – Метамодель архітектури програмного забезпечення для обліку комп’ютерної техніки Мішково-Погорілівської загальноосвітньої санаторної школи-інтернату

Архітектура програмного забезпечення складається з усіх прийнятих проектних рішень з приводу структури програмного забезпечення, а також взаємодій між компонентами структури. Ці рішення забезпечать потрібний набір властивостей, які повинно підтримувати програмного забезпечення, щоб бути успішним. Проектні рішення надають концептуальну основу для розробки програмного забезпечення, його підтримки та обслуговування.

Для зручної розробки та відокремлення бізнес-логіки від дизайну, при розробці програмного забезпечення для обліку комп’ютерної техніки Мішково-Погорілівської загальноосвітньої санаторної школи-інтернату, був обраний паттерн проектування MVC – Model-View-Controller, або Модель-Уявлення-Контролер.

Паттерн проектування MVC – це схема розподілу між даними програмного забезпечення, користувацьким інтерфейсом та логікою, що управляє програмою, на окремі компоненти: модель, уявлення та контролер

таким чином, що зміна кожного з цих компонентів може проходити незалежно від двох інших.

Патерн MVC дозволить спростити процес розробки шляхом поділу частини дизайну від логіки. Тим же способом вирішується проблема масштабування, відділення бізнес-логіки й обробки даних в модулях взаємодії з користувачем, в модулі моделі і контролерів, дозволяє зменшувати навантаження. Так само даний патерн проектування є дуже гнучким, дозволяючи вносити зміни в існуючий функціонал розроблюваного програмного забезпечення дуже легко і швидко [10].

Компонент View здійснює взаємодію між користувачем і системою. Компонент презентує і відображає дані, отримані з бази даних, у вигляді інтерфейсу користувача. Через даний компонент команди користувача надсилаються до компонента Controller, отримуються та відображаються дані з компонента Model.

Компонент Controller здійснює взаємодію між користувачем і компонентом, який надає дані. Компонент отримує команди користувача з компонента View та здійснює дії для зміни даних у компоненті Model згідно цих команд.

Компонент Model здійснює взаємодію між компонентом, який надає команди на зміну даних, та інтерфейсом користувача. Компонент отримує команди на зміну даних з компонента Controller, запитує дані у компонента View і надсилає потрібні дані до компонента View.

2.3 Проект програмного забезпечення

2.3.1 Ескізний проект програмного забезпечення

Ескізний проект програмного забезпечення для обліку комп’ютерної техніки Мішково-Погорілівської загальноосвітньої санаторної школи-інтернату включає в себе побудову сценаріїв варіантів використання у вигляді діаграми діяльності варіантів використання, розробку концептуальної моделі у вигляді ER-діаграми, а також розробку прототипу інтерфейсу користувача.

Побудова діаграми діяльності

Діаграма діяльності по суті представляє собою граф особливого виду, що представляє деякий автомат, де вершинами графа являються стани та інші типи елементів, що зображуються відповідними графічними символами. Дуги графа слугують для позначення переходів із одного стану в інший стан. Діаграми діяльності можуть вкладатися одна в одну для більш детального подання окремих елементів[8].

Програма починає роботу у стані ідентифікації користувача. Якщо у доступі відомвлено то іде перехід на стан завершення роботи. Якщо доступ отримано, програма переходить у стан вибору одного з трьох режимів, а саме перебування у режимі вибору звітів, перебування у режимі вибору довідника або перебування у режимі вибору запиту.

У режимі вибору звітів є такі звіти: "Технічні характеристики комп`ютера", "Дані по комп`ютерному класу".

У режімі вибору запитів є такі запити: "Характеристики комп`ютера", "Перелік комп`ютерів по класам", "Дані по комп`ютерному класу".

У режимі вибору довідників є такі довідники: "Типи комп`ютерної техніки", "Розклад занять", "Комп`ютерний клас", "Інструктивно-методичні матеріали", "Програмне забезпечення комп`ютера", "Технічні характеристики комп`ютера".

Після виконання необхідних дій користувача програма переходить у стан вибору режиму для завершення роботи.

Діаграми діяльності для деяких варіантів використання програмного забезпечення для обліку комп’ютерної техніки Мішково-Погорілівської загальноосвітньої санаторної школи-інтернату наведені на рисунках 2.3 – 2.4.

Рисунок 2.3 – Діаграма діяльності для варіанту використання "Формування звітів"

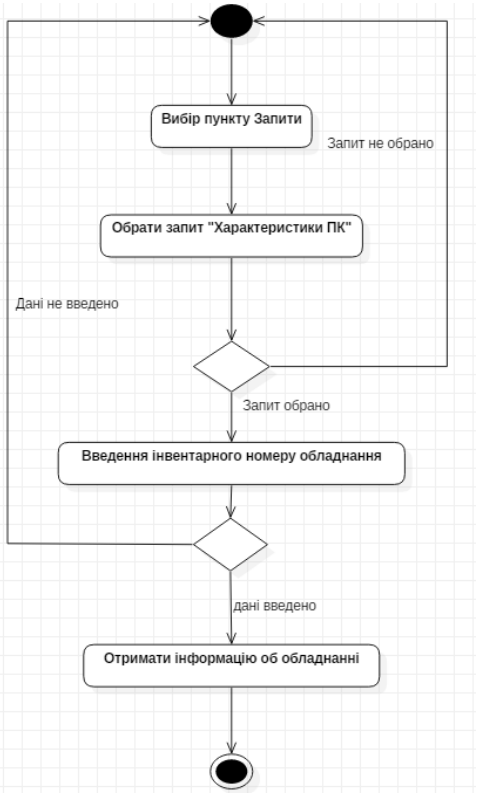

Рисунок 2.4 – Діаграма діяльності для варіанту використання запиту "Характеристики комп`ютера"

Кожна дія, представлена на діаграмі діяльності може виконуватись або один, або більше одного разу під час одного виконання діаграми дічльності. Ці дії отримують дані, перетворюють отримані дані та вимагають певної послідовності.

Розробка концептуальної моделі

З метою визначення зв’язків між потоками дани, на основі вимог до функціональних характеристик, визначених в технічному завданні, розробляють концептуальну модель даних. Концептуальна модель даних програмного забезпечення для обліку комп’ютерної техніки Мішково-Погорілівської загальноосвітньої санаторної школи-інтернату представлена у вигляді ER-діаграми та наведена на рисунку 2.5. Вона дозволяє визначити сутності, їх атрибути, а також зв’язки між сутностями.

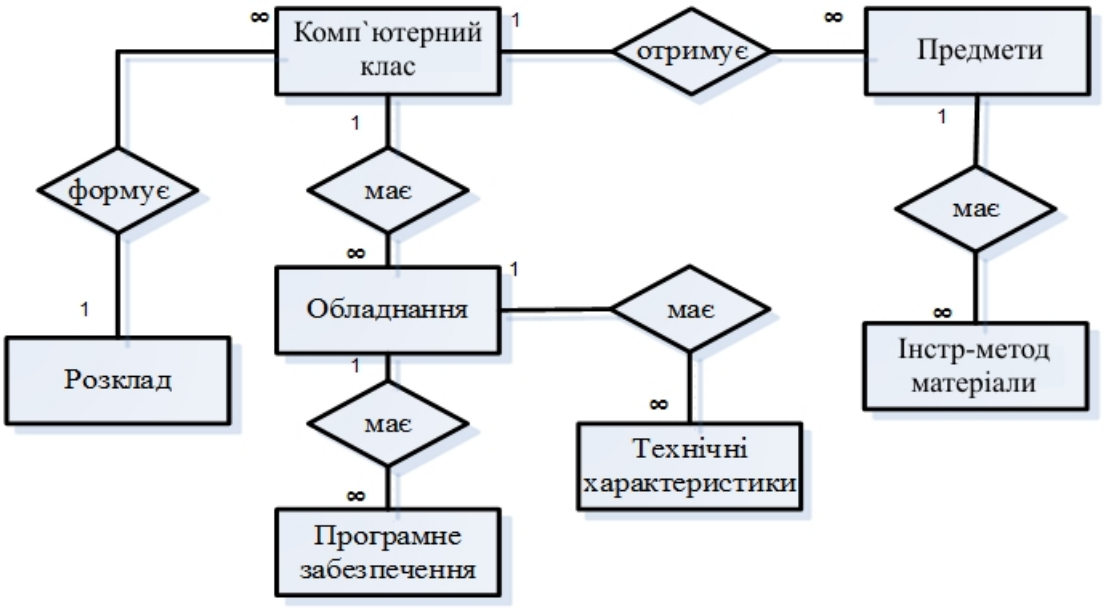

Рисунок 2.5 – Концептуальна модель даних програмного забезпечення для обліку комп’ютерної техніки Мішково-Погорілівської загальноосвітньої санаторної школи-інтернату

Кожна з сутностей може являти собою безліч екземплярів реальних чи абстрактних об’єктів, що володіють загальними атрибутами чи характеристиками. Будь-який об’єкт програмного забезпечення може бути представлений тільки однією сутністю, що унікально ідентифікована. Відношення в своєму загальному вигляді являються зв’язком між двома або більше сутностями [11].

Розробка прототипу інтерфейсу користувача

Користувацький інтерфейс– це сукупність засобів, які обробляють та відображають інформацію у максимально зручному для користувача вигляді. Зазвичай користувацький інтерфейс реалізують за допомогою віконного інтерфейсу, змін кольору фону або керуючих елементів, їх розміру та розташування, а також видимості елементів або вікон. Також можна використовувати гнучкі налаштування керуючих елементів.

Графічний користувацький інтерфейс – це інтерфейс, який дає можливість користувачу взаємодіяти з комп'ютером та складається з різноманітних типів

меню та значків, кнопок і таке інше. Користувач може відкрити декілька вікон на одному екрані [12].

Створення максимально ефективного прототипу інтерфейсу є надзвичайно важливою задачею. Прототип повинен бути максимально повним, дешевим та повинен легко оновлюватися.

На рисунках 2.6 – 2.7 представлено прототип інтерфейсу основних форм програмного забезпечення для обліку комп’ютерної техніки Мішково-Погорілівської загальноосвітньої санаторної школи-інтернату.

Рисунок 2.6 – Прототип головного меню програмного забезпечення для обліку комп’ютерної техніки Мішково-Погорілівської загальноосвітньої санаторної школи-інтернату

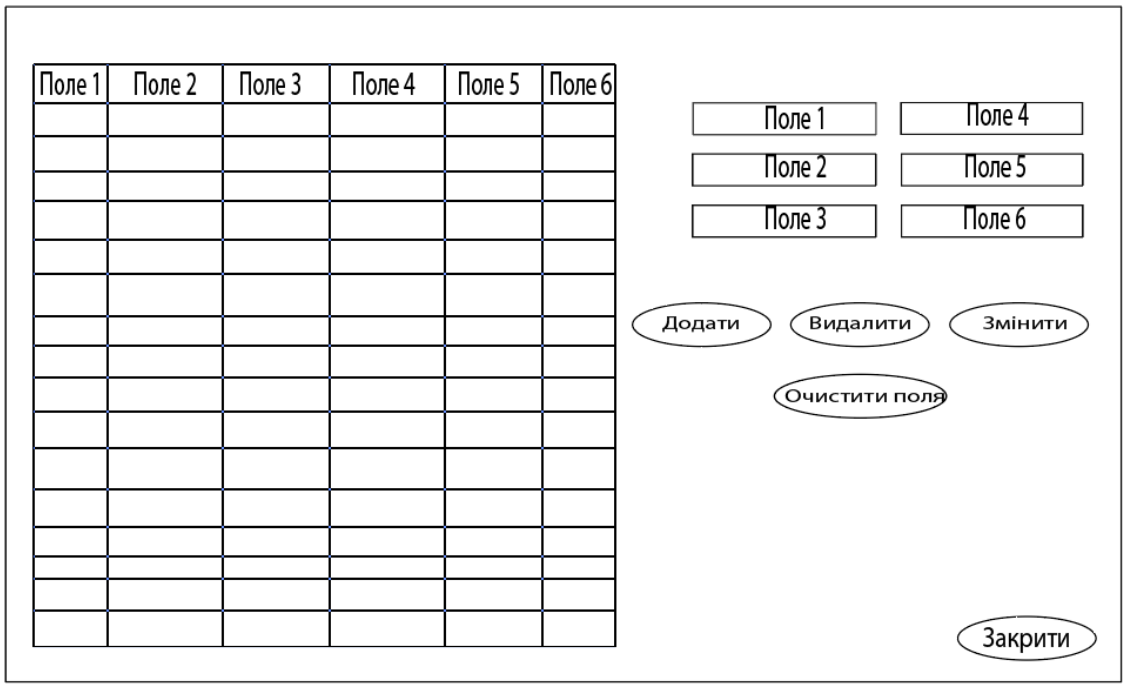

Рисунок 2.7 – Прототип форми обробки даних програмного забезпечення для обліку комп’ютерної техніки Мішково-Погорілівської загальноосвітньої санаторної школи-інтернату

Під час роботи над прототипом інтерфейсу користувача було створено простий, але повний інтерфейс для основних форм програми.

2.3.2 Технічний проект програмного забезпечення

Технічний проект програмного забезпечення для обліку комп’ютерної техніки Мішково-Погорілівської загальноосвітньої санаторної школи-інтернату включає в себе побудову статичної моделі у вигляді діаграми класів, розробку динамічної моделі у вигляді діаграми взаємодії, а також розробку логічної моделі бази даних.

Побудова діаграми класів

Діаграма класів – це статичне представлення структури програмного забезпечення, яке відображає його статичні елементи: класи, типи даних, їх зміст, відношення. Діаграма класів також може містити позначення для пакетів

та вкладених пакетів. Діаграма класів також може містити позначення поведінкових елементів, але їх динаміка представляється іншими типами діаграм [13].

Класи та зв`язки між класами програмного забезпечення для обліку комп’ютерної техніки Мішково-Погорілівської загальноосвітньої санаторної школи-інтернату наведені на рисунку 2.8.

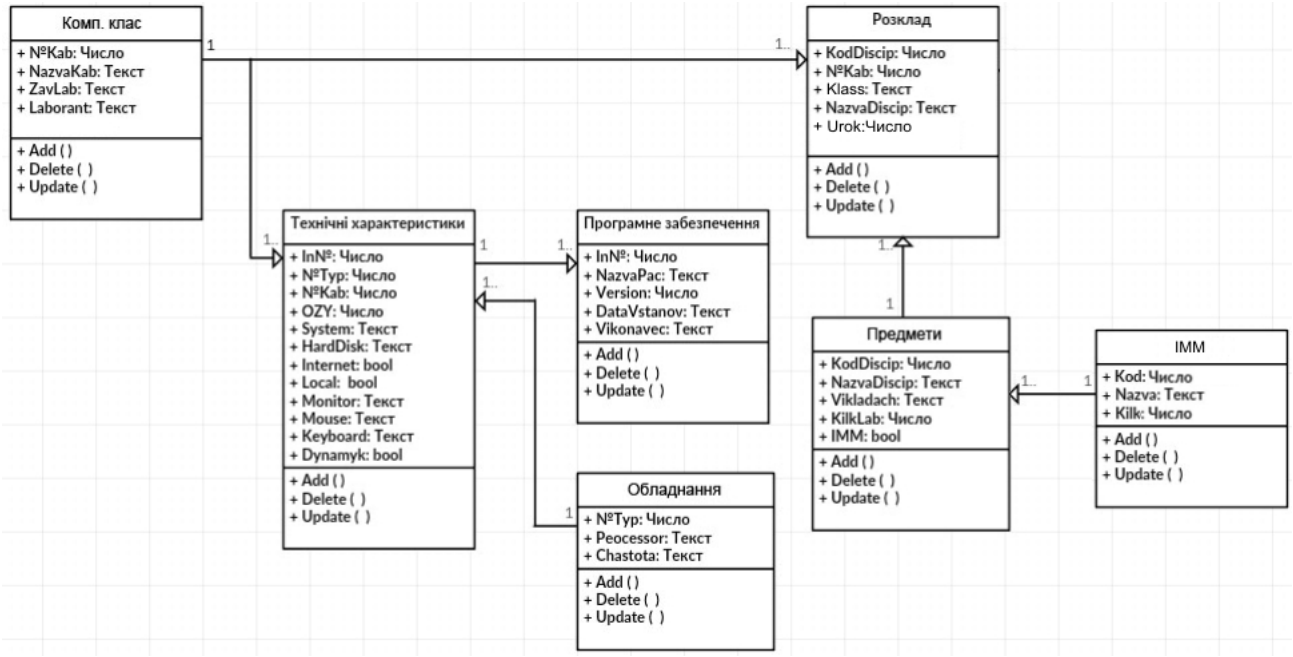

Рисунок 2.8 – Діаграма класів програмного забезпечення для обліку комп’ютерної техніки Мішково-Погорілівської загальноосвітньої санаторної школи-інтернату

Діаграма класів відображає статичну структуру в термінології класів об'єктно-орієнтованого проєктування. На діаграмі класів відображають класи, інтерфейси, об'єкти й кооперації, а також їхні відносини.

Побудова діаграми взаємодії

Діаграма взаємодії – це одна з UML-діаграм, яка описує поведінку груп об'єктів, що взаємодіють між собою. На цій діаграмі відображають екземпляри об'єктів та повідомлення, якими ці об'єкти обмінюються в рамках одного варіанта використання [14].

На рисунках 2.9 – 2.10 зображені діаграми взаємодії для декількох варіантів використання програмного забезпечення для обліку комп’ютерної техніки Мішково-Погорілівської загальноосвітньої санаторної школи-інтернату.

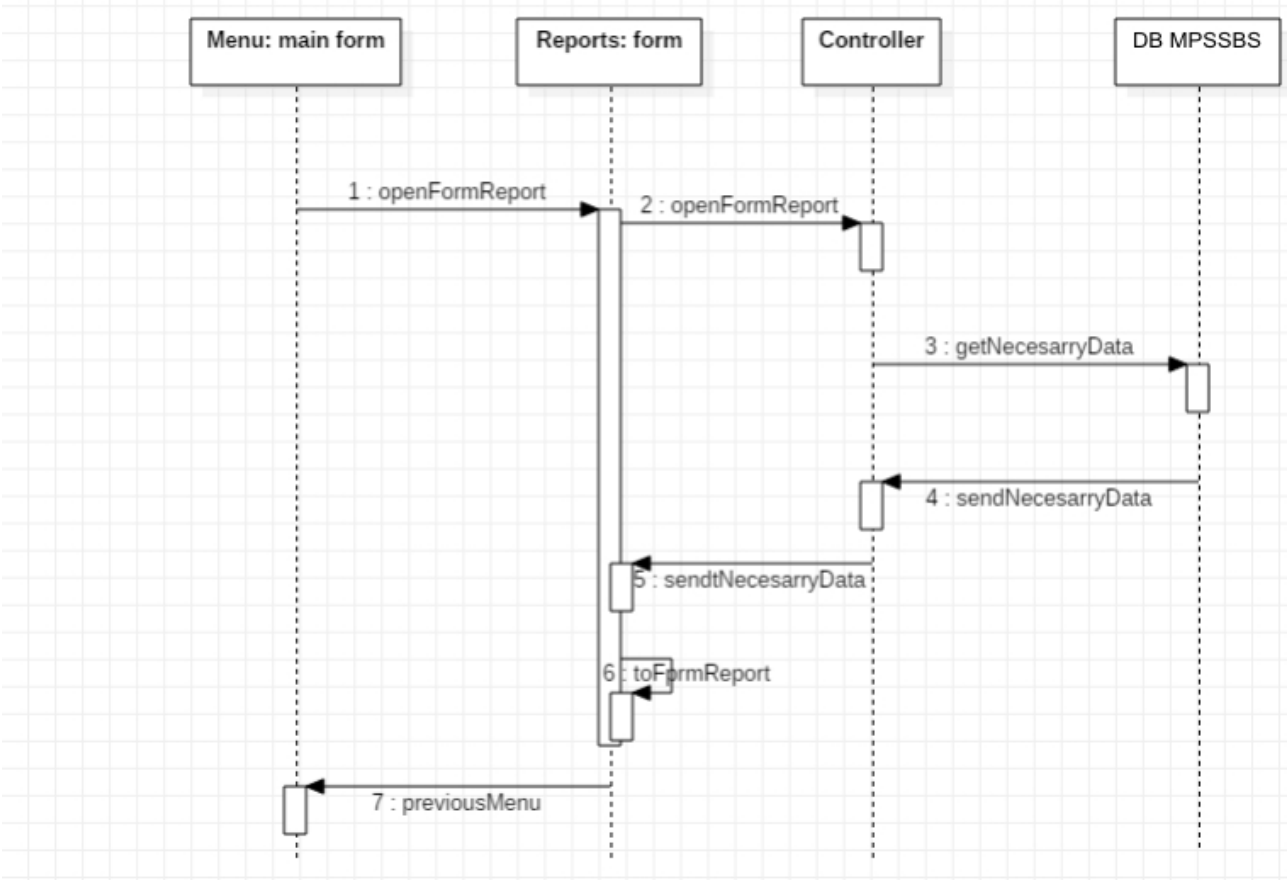

Рисунок 2.9 – Діаграма взаємодії для варіанту використання "Формування звітів"

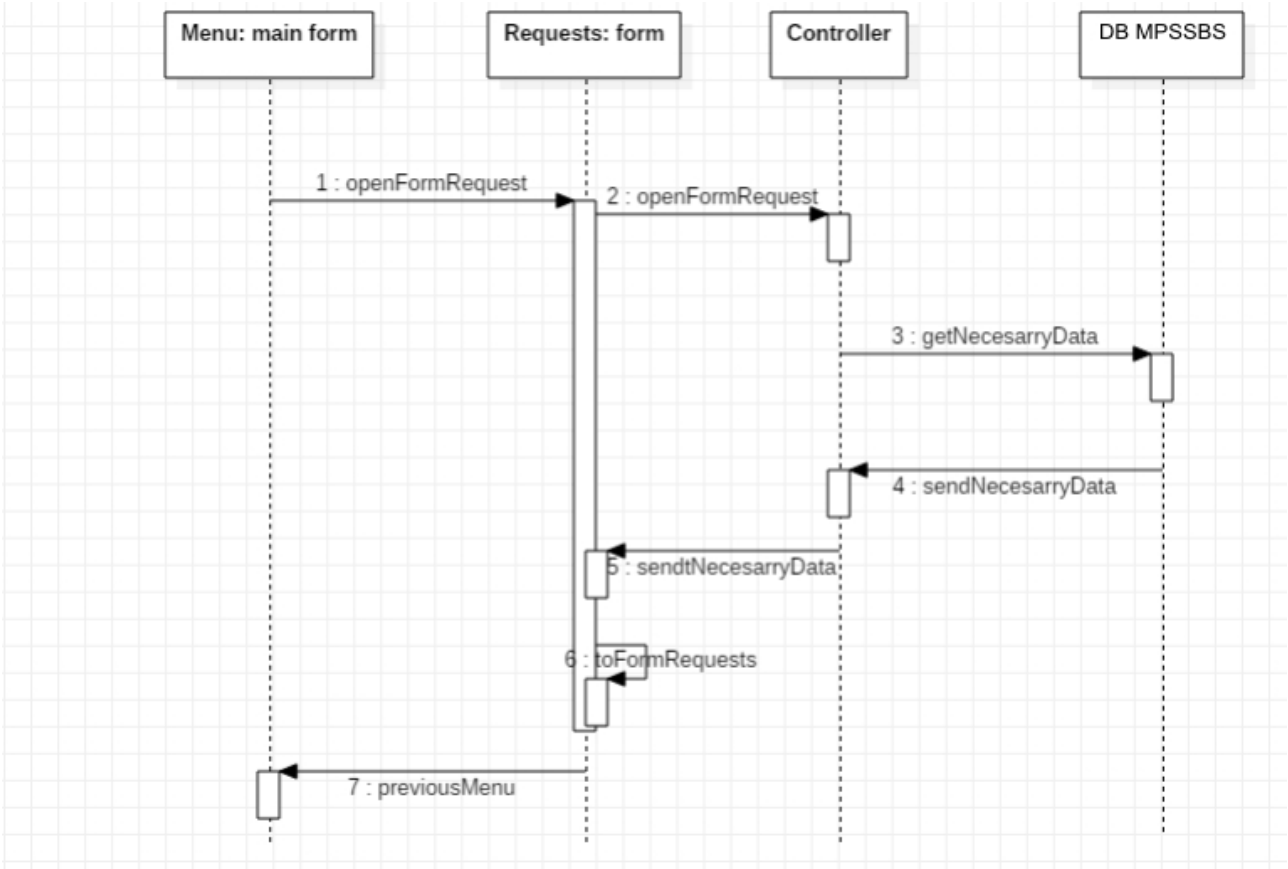

Рисунок 2.10 – Діаграма взаємодії для варіанту використання запиту «Характеристики комп`ютера»

Як правило, кожна окрема діаграма взаємодії описує поведінку тільки в межах одного варіанта використання.

Розробка логічної моделі бази даних

Для того щоб побудувати логічну модель бази даних використаємо концептуальну модель даних, що була розроблена раніше та наведена на рисунку 2.5. Мета створення логічної моделі бази даних – більш точне уявлення про склад та типи даних, які будуть використовуватися в програмному забезпеченні.

Логічна модель бази даних програмного забезпечення для обліку комп’ютерної техніки Мішково-Погорілівської загальноосвітньої санаторної школи-інтернату наведена на рис.2.11.

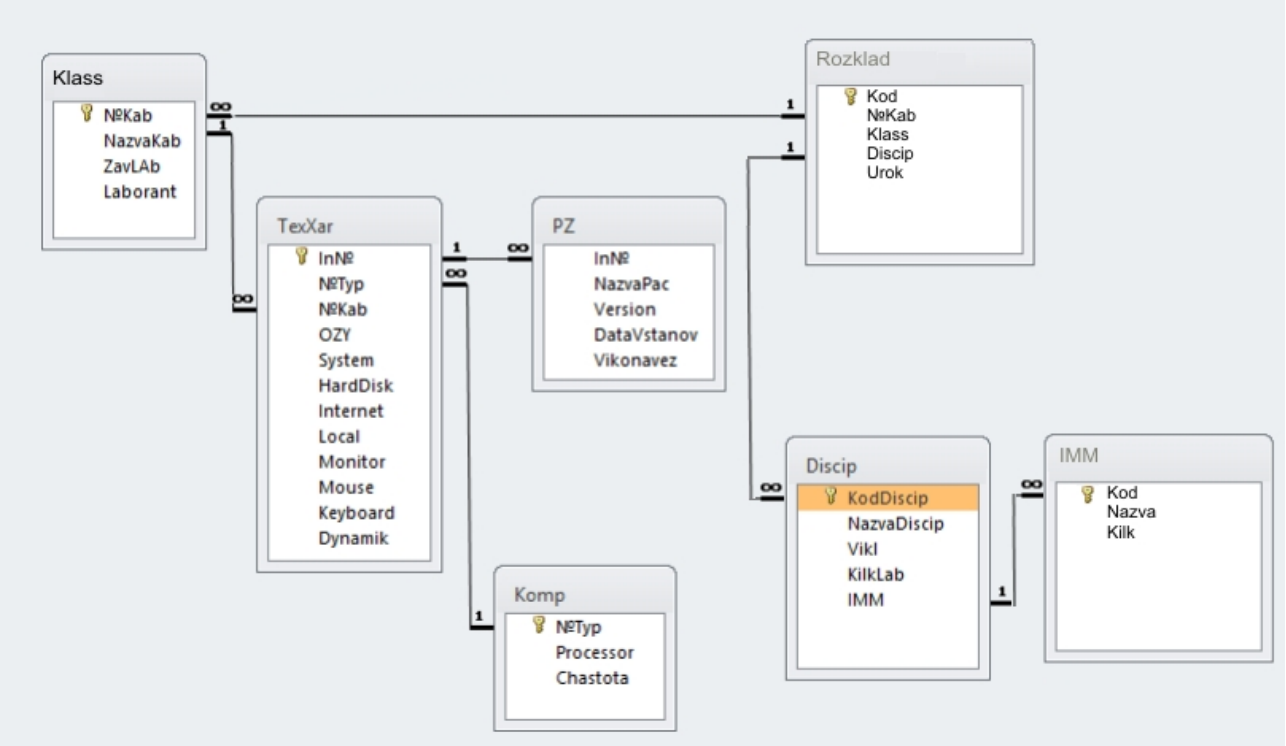

Рисунок 2.11 – Логічна модель бази даних програмного забезпечення для обліку комп’ютерної техніки Мішково-Погорілівської загальноосвітньої санаторної школи-інтернату

В таблицях 2.5 – 2.11 представлені інформаційні об’єкти бази даних.

Таблиця 2.5 – Інформаційний об’єкт «Предмет»

Поле	Тип	Первинний ключ
Код предмета	Числовий	+
Назва	Текстовий	
Викладач	Текстовий	
Кількість занять	Числовий	
IMM	Логічний	

Таблиця 2.6– Інформаційний об’єкт «Обладнання»

Поле	Тип	Первинний ключ
№ Типу	Числовий	+
Процесор	Текстовий	
Частота	Текстовий	

Таблиця 2.7– Інформаційний об’єкт «Комп`ютерний клас»

Поле	Тип	Первинний ключ
№ комп`ютерного класу	Числовий	+
Назва комп`ютерного класу	Текстовий	
Завідувач комп`ютерного класу	Текстовий	
Лаборант	Текстовий	

Таблиця 2.8– Інформаційний об’єкт «Програмне забезпечення»

Поле	Тип	Первинний ключ
Інв №	Числовий	+
Назва пакету	Текстовий	
Версія	Числовий	
Дата встановлення	Текстовий	
Виконавець	Текстовий	

Таблиця 2.9– Інформаційний об’єкт «Технічні характеристики»

Поле	Тип	Первинний ключ
Інв №	Числовий	+
№ Типу	Числовий	
№ комп`ютерного класу	Числовий	
ОЗУ	Числовий	
Система	Текстовий	
Жорсткий диск	Текстовий	
Інтернет	Логічний	
Локальна мережа	Логічний	
Монітор	Текстовий	
Миша	Текстовий	
Клавіатура	Текстовий	
Динаміки	Логічний	

Таблиця 2.10– Інформаційний об’єкт «ІММ»

Поле	Тип	Первинний ключ
Код	Числовий	+
Назва	Текстовий	
Кількість	Числовий	

Таблиця 2.11– Інформаційний об’єкт «Розклад»

Поле	Тип	Первинний ключ
Код	Числовий	+
№ комп`ютерного класу	Числовий	
Клас	Текстовий	
Предмет	Текстовий	
Урок	Числовий	

Таким чином, у наведених таблицях представлено інформацію про основні об`єкти логічної моделі майбутньої бази даних та їх характеристики.

2.3.3 Робочий проект програмного забезпечення

Робочий проект програмного забезпечення для обліку комп’ютерної техніки Мішково-Погорілівської загальноосвітньої санаторної школи-інтернату включає в себе вибір та обґрунтування інструментів розробки, реалізацію та тестування програмного забезпечення.

Вибір та обґрунтування інструментів розробки

Програмне забезпечення для обліку комп’ютерної техніки Мішково-Погорілівської загальноосвітньої санаторної школи-інтернату повинно реалізовувати функції створення, збереження, пошуку даних в масивах інформації. Інформаційні масиви повинні утримувати в собі довідкові та оперативні дані. За запитами користувача потрібно формувати та видавати йому конкретні дані оперативного та довідкового характеру.

Для програмної реалізації була обрана мова об'єктно-орієнтованого програмування C++, використовуючи інтегроване середовище візуального програмування NetBeans IDE, та мова структурованих запитів SQL у СУБД Microsoft Access для розробки бази даних.

C++ – це високорівнева мова програмування, яка підтримує декілька парадигм програмування, зокрема об'єктно-орієнтовану та процедурну. Вона має ряд переваг порівняно з іншими об’єктно-орієнтованими мовами:

– швидкість роботи програм на C++ найвища порівняно з більш популярними об'єктно-орієнтованими мовами, такими як C#, Java;

– мовою C++ розробляють програми для платформ і систем різних типів;

– для різних типів даних можливо створювати як узагальнені алгоритми, так і враховувати їх спеціалізацію;

– можливе обчислення на етапі компіляції з використанням шаблонів;

– програма мовою C++ може як використовувати компоненти VCL, так і працювати напряму з WinAPI [15].

В якості середовища розробки було обрано середовище NetBeans IDE – безкоштовне інтегроване середовище розробки з відкритим вихідним кодом для розробників програмного забезпечення. Середовище надає всі засоби, необхідні для створення професійних десктоп додатків, корпоративних, мобільних і веб-додатків на платформі Java, а також C/C++, PHP, JavaScript, Groovy і Ruby. Робоча область середовища IDE повністю настроюється – існує можливість настройки для користувача дій, які виконуються за допомогою панелі, призначення "гарячих" клавіш і т.д. [16].

Середовище розробки NetBeans IDE має в своєму складі розширений багатомовний редактор для різних мов програмування – Java, C/C++, Ruby, Groovy, PHP, JavaScript, CSS, XML, HTML, RHTML, JSP, документацію Javadoc. Існує можливість розширення функцій редактора з метою підтримки будь-якої іншого мови. Редактор NetBeans робить відступи рядків, перевіряє відповідність дужок і слів, підсвічує синтаксис вихідного коду. Проводиться перевірка помилок під час введення, відображення варіантів для автозавершення коду і фрагментів документації по обраній мові програмування.

Розширені засоби для виконання контекстно-залежного пошуку по всьому середовищу IDE, довідкових матеріалах і всіх відкритих проектах та файлах. Існує можливість створення проектів у вільному форматі або починати роботу з проектом з шаблону. У комплекті з середовищем IDE поставляються шаблони і приклади проектів для додатків Java SE, мобільних, веб-додатків і додатків рівня підприємства, додатків JavaFX, модулів NetBeans, додатків Groovy, PHP,

C/C++, Ruby і Ruby on Rails. NetBeans має вбудовану підтримку CVS, Mercurial і Subversion. Для перегляду змін використовується редактор з колірними позначеннями.

SQL – це мова програмування для роботи з базами даних, здійснення запитів та внесення змін, а також управління базами даних. Різні бази даних підтримують мову SQL з розширеннями до стандартної мови. Ядро SQL формує командна мова, яка дозволяє здійснювати пошук, вставку, оновлення та вилучення даних, використовуючи при цьому систему управління базами даних та адміністративні функції.

Microsoft Office Access або Microsoft Access – це реляційна система управління базами даних, яка розроблена Microsoft Corporation. Вона входить до офісного пакету Microsoft Office. Вона має широкий спектр функцій, включаючи відповідні запити, зв'язок із зовнішніми таблицями та базами даних. Завдяки вбудованій мові VBA, безпосередньо в самому Microsoft Access можливо писати застосунки, що працюють з базами даних [17].

Розробка фізичної модель бази даних

Для побудови фізичної моделі використовуємо уже розроблену логічну модель даних. Побудуємо фізичні моделі таблиць зі значеннями: Назви поля, Типу даних та Довжини. Для реалізації БД вибрано СУБД MSAccess.

В таблицях 2.12 – 2.18 представлені інформаційні об’єкти розробленої бази даних.

Таблиця 2.12 – Фізична модель таблиці «Дисципліни» (Discip)

Логічна назва поля	Фізична назва поля	Тип	Довжина поля
Код Дисципліни	KodDiscip	Числовий	Довге ціле
Назва Дисципліни	NazvaDiscip	Текстовий	25
Викладач	Vikl	Текстовий	25
Кількість занять	KilkLab	Числовий	Довге ціле
ІММ	ІММ	Логічний	

Таблиця 2.13 – Фізична модель таблиці «Обладнання» (Komp)

Логічна назва поля	Фізична назва поля	Тип	Довжина поля
№Типу	№Typ	Числовий	Довге ціле
Процесор	Processor	Текстовий	15
Частота	Chastota	Числовий	Довге ціле

Таблиця 2.14 – Фізична модель таблиці «Комп`ютерний клас» (Klass)

Логічна назва поля	Фізична назва поля	Тип	Довжина поля
Номер комп`ютерного класу	№Kab	Числовий	Довге ціле
Назва комп`ютерного класу	NazvaKab	Текстовий	15
Зав комп`ютерного класу	ZavLab	Текстовий	20
Лаборант	Laborant	Текстовий	20

Таблиця 2.15 – Фізична модель таблиці «Програмне забезпечення» (PZ)

Логічна назва поля	Фізична назва поля	Тип	Довжина поля
Інвентарний номер	Inv№	Числовий	Довге ціле
Назва пакету	NazvaPac	Текстовий	20
Версія	Version	Числовий	Довге ціле
Дата Встановки	DataVstanov	Текстовий	15
Виконавець	Vikonavez	Текстовий	20

Таблиця 2.16 – Фізична модель таблиці «Технічні характеристики» (TexXar)

Логічна назва поля	Фізична назва поля	Тип	Довжина поля
1	2	3	4
Інвентарний номер	In№	Числовий	Довге ціле
Номер типу	№Typ	Числовий	Довге ціле
Номер комп`ютерного класу	№Kab	Числовий	Довге ціле
ОЗУ	OZY	Числовий	Довге ціле
Система	System	Текстовий	20
Жорсткий Диск	HardDisk	Текстовий	20
Інтернет	Internet	Логічний	
Локальна мережа	Local	Логічний	

Продовження таблиці 2.16

1	2	3	4
Монітор	Monitor	Текстовий	25
Миша	Mouse	Текстовий	25
Клавіатура	Keyboard	Текстовий	25
Динаміки	Dynamik	Логічний	

Таблиця 2.17 – Фізична модель таблиці «ІММ»

Логічна назва поля	Фізична назва поля	Тип	Довжина поля
Код	Kod	Числовий	Довге ціле
Назва	Nazva	Текстовий	20
Кількість	Kilk	Числовий	Довге ціле

Таблиця 2.18–Фізична модель таблиці «Розклад»

Логічна назва поля	Фізична назва поля	Тип	Довжина поля
Код	Kod	Числовий	Довге ціле
Номер комп`ютерного класу	№Kab	Числовий	Довге ціле
Клас	Klass	Текстовий	10
Предмет	Discip	Текстовий	20
Урок	Urok	Числовий	Довге ціле

3 РЕЗУЛЬТАТИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОБЛІКУ КОМП'ЮТЕРНОЇ ТЕХНІКИ МІШКОВО-ПОГОРІЛІВСЬКОЇ ЗАГАЛЬНООСВІТНЬОЇ САНАТОРНОЇ ШКОЛИ-ІНТЕРНАТУ

В представленій кваліфікаційній роботі була розв'язана практична задача інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов, а саме розроблено програмне забезпечення для обліку комп'ютерної техніки Мішково-Погорілівської загальноосвітньої санаторної школи-інтернату. Зовнішній вигляд головного вікна програмного забезпечення наведено на рисунку 3.1.

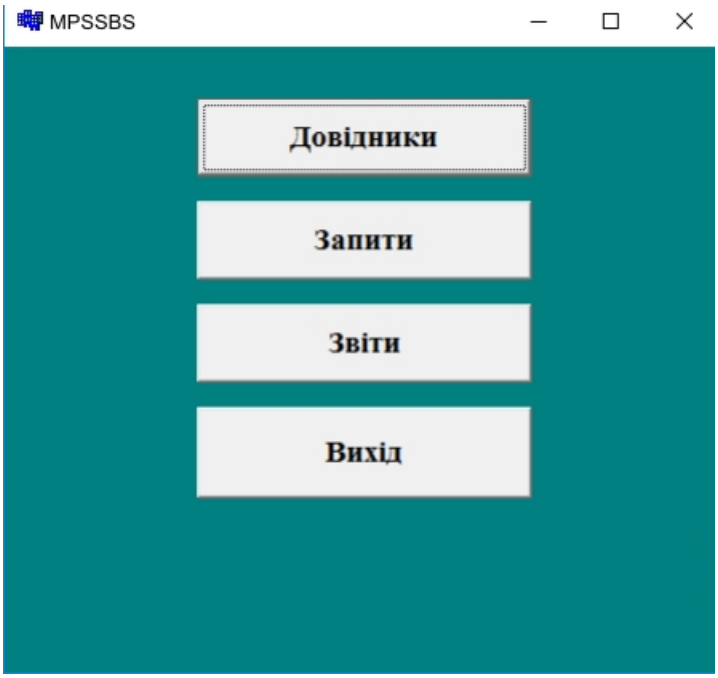

Рисунок 3.1 – Зовнішній вигляд головного вікна програмного забезпечення для обліку комп'ютерної техніки Мішково-Погорілівської загальноосвітньої санаторної школи-інтернату

Програмне забезпечення дозволяє зберігання, перегляд, пошук, редагування та видалення інформації про комп'ютерну техніку, пакети

прикладних програм та інструктивно-методичні матеріали, формування розкладу, а також інвентаризації комп’ютерної техніки.

Проведено аналіз вимог до програмного забезпечення, розроблено архітектуру програмного забезпечення, розроблено проєкт програмного забезпечення, виконані реалізація та тестування програмного забезпечення.

В рамках аналізу вимог до програмного забезпечення побудована модель варіантів використання та розроблена специфікація варіантів використання.

В якості архітектури програмного забезпечення була обрана архітектура MVC: Model-View-Controller, або Модель-Уявлення-Контролер.

В рамках проєкту програмного забезпечення виконані ескізний, технічний, робочий проєкти. В рамках ескізного проєкту виконана побудова сценаріїв варіантів використання у вигляді діаграми діяльності варіантів використання, розроблена концептуальна моделі у вигляді ER-діаграми, розроблено прототип інтерфейсу користувача.

В рамках технічного проекту побудована статична модель у вигляді діаграми класів, розроблена динамічна модель у вигляді діаграми взаємодії, розроблена логічна модель бази даних.

В рамках робочого проекту виконано вибір та обґрунтування інструментів розробки, реалізацію та тестування програмного забезпечення. В якості мови програмування була обрана об'єктно-орієнтована мова програмування C++, в якості середовища розробки – інтегроване середовище візуального програмування NetBeans IDE, для роботи з базою даних була обрана мова структурованих запитів SQL, в якості СУБД – СУБД Microsoft Access.

Розроблене програмне забезпечення використовує незначну кількість ресурсів на комп’ютері, а впровадження програмного забезпечення дозволить оперативно отримувати необхідну інформацію щодо комп’ютерної техніки та аналітичні звіти.

Було виконано випробування програмного забезпечення, яке показало повну працездатність та відповідність вимогам, які представлені у підрозділі 1.3 Постановка задачі на розробку програмного забезпечення для обліку

комп’ютерної техніки Мішково-Погорілівської загальноосвітньої санаторної школи-інтернату, а також у Додатку А Технічне завдання даної кваліфікаційної роботи.

Також була розроблена уся необхідна програмна документація, перелік якої наведено у Додатку А Технічне завдання даної кваліфікаційної роботи:

– технічне завдання представлено у Додатку А;

– текст програми представлено у Додатку Б;

– опис програми представлено у Додатку В;

– інструкцію користувача представлено у Додатку Г;

– програму та методика випробування представлено у Додатку Д.

4 ОХОРОНА ПРАЦІ

4.1 Вступ

Охорона праці – це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних та лікувально-профілактичних заходів і засобів, спрямованих на збереження здоров’я та працездатності людини в процесі трудової діяльності [18]. Охорона праці є комплексною науковою дисципліною, яка базується на теоретичних положеннях природничих (фізика, хімія, біологія, медицина), математичних (математичне моделювання, теорія ймовірностей, математична статистика), суспільних (економіка, соціологія, психологія, право) і технічних наук (електротехніка, електроніка, автоматика, системологія, ергономіка, інженерна психологія, технічна естетика, наукова організація праці). Особливо тісно охорона праці пов'язана з безпекою життєдіяльності, екологією, науковою організацією праці, ергономікою, інженерною психологією та технічною естетикою. У них єдина мета – сприяти збереженню здоров'я та працездатності людини, підвищенню її продуктивності праці, усуненню або зменшенню впливу на неї шкідливих і небезпечних виробничих чинників. У той же час усі вони підходять до досягнення поставленої мети з різних боків і на різних рівнях, а саме:

1. Безпека життєдіяльності вивчає загальні закономірності виникнення небезпек, їх властивості та особливості впливу на людину, наслідки такого впливу, а також способи та засоби захисту життя та здоров'я людини й середовища її проживання від реальних та потенційних небезпек.

2. Наукова організація праці займається вивченням, розробкою та впровадженням у практику раціональної побудови трудового процесу, за якої забезпечується висока продуктивність праці, створюються умови для збереження здоров'я працівників, збільшується період їх трудової діяльності.

3. Ергономіка досліджує, розробляв та дає рекомендації щодо конструювання, виготовлення та експлуатації технічних засобів, які забезпечують людині в процесі праці необхідні зручності, зберігають її сили, працездатність та здоров'я.

4. Інженерна психологія вивчає взаємодію людини з новою технікою і з'ясовує функціональні можливості людини в трудових процесах з метою створення таких умов праці, за яких зберігаються високі психофізіологічні можливості людини.

5. Технічна естетика встановлює залежність умов та результатів праці від архітектурного, конструктивного та художнього вирішення знарядь праці, робочих місць, дільниць, цехів, санітарно-побутових та інших допоміжних приміщень, усього, що оточує людину на виробництві.

Інваріантною основою змісту вищеназваного є система “людина – середовище”, де під середовищем розуміють як природне середовище (біосфера), так і штучне середовище (техносфера, соціальна сфера). Охорона праці розглядає штучне середовище – виробництво, об’єкти виробничої діяльності, що конкретизує техносферу [19].

4.2 Санітарно-гігієнічні вимоги до виробничого середовища програміста

Санітарно-гігієнічні вимоги до процесу роботи людини у приміщені з персональним комп`ютером визначають: мікроклімат; степінь освітленості; рівень шуму і вібрації; іонізуюче та неіонізуюче випромінювання [20].

Мікроклімат

Мікроклімат на робочих місцях з персональним комп`ютером нормується ДЕСТ 12.1.005-88 і описаний у таблиці 4.1.

Таблиця 4.1 – Нормовані параметри мікроклімату для приміщень з персональним комп`ютером

Пора року	Категорія робіт	Температура	Відносна вологість, %	Швидкість руху повітря, м/с, не більше
Холодна пора року	Легка – Іа Легка – Іб	22-24 21-23	40-60 40-60	0,1 0,1
Тепла пора року	Легка – Іа Легка – Іб	23-25 22-24	40-60 40-60	0,1 0,1

Освітлення

1) Робочі місця з персональними комп`ютерами повинні мати природне і штучне освітлення.

2) Вікна приміщень з персональними комп`ютерами повинні мати орієнтацію на північ або на північний схід, з КПО 1,5%. Дозволяється експлуатація персональних комп`ютерів у приміщеннях без природного освітлення за узгодженням з органами Держнаглядохоронпраці та санітарно-епідеміологічними установами.

3) Рівень освітленості на робочому місці в зоні розміщення документів має бути від 300 лк до 500лк;

4) Штучне освітлення виконується, як правило, люмінесцентними лампами. Допускається для місцевого освітлення використовувати лампи розжарювання.

5) Місцеве освітлення не повинно створювати блисків на екрані персональних комп`ютерів, а його освітленість повинна не перевищувати 300 лк.

Рівні шуму

Рівні шуму на робочих місцях з персональними комп`ютерами визначені ДсанПіН 3.3.2-007-98 в залежності від його тональності та виду трудової

діяльності користувачів персональних комп`ютерів (програмісти, оператори, оператори комп'ютерного набору). Рівні звукового тиску описані у таблиці 4.2.

Таблиця 4.2 – Рівні звукового тиску в дБ в октавних смугах частот, Гц

Вид трудової діяльності / Робочі місця	Рівні звукового тиску в дБ в октавних смугах із середньогеометричними частотами, Гц									
	31,5	63	125	250	500	1000	2000	4000	8000	Рівні звуку, еквівалентні рівні звуку, дБА/дБА екв.
Програмісти	86	71	61	54	49	45	42	40	38	50
Оператори в залах обробки інформації на ПК та оператори комп'ютерного набору	96	83	74	68	63	60	57	55	54	65
В приміщеннях для розташування шумних агрегатів	103	91	83	77	73	70	68	66	64	75

Для нормування шуму застосовуються шумопоглинальні засоби, визначені спеціальними інженерно-акустичними розрахунками.

Рівні вібрації

Рівні вібрації не повинні перевищувати допустимих норм, визначених ДсанПіН 3.3.2-007-98.

Рівні іонізуючого випромінювання

Рівні іонізуючого випромінювання повинні відповідати НРБУ–97. Потужність експозиційної дози рентгенівського випромінювання на відстані 5 см від екрану ПК повинна бути не більше 0,1 мбер/година (100 мкР/година).

Рівні неіонізуючого випромінювання

Рівні неіонізуючого випромінювання повинні відповідати вимогам ДЕСТ 12.1.006, СН №3206-85 «Гранично допустимі рівні магнітного поля частотою 50 Гц, СН №4557-88 «Санітарні норми ультрафіолетового випромінювання у виробничих приміщеннях».

4.3 Ергономічні вимоги до робочого місця програміста

Проектування робочих місць, забезпечених відеотерміналами, відноситься до числа важливих проблем ергономічного проектування в області обчислювальної техніки.

Робоче місце користувача персонального комп`тера повинно відповідати вимогам ДЕСТ 12.2.032-78.

Робоче місце і взаємне розташовує всіх його елементів повинне відповідати антропометричним, фізичним і психологічним вимогам. Велике значення має також характер роботи. Зокрема, при організації робочого місця програміста повинні бути дотримані наступні основні умови: оптимальне розміщення устаткування, що входить до складу робочого місця і достатній робочий простір, що дозволяє здійснювати всі необхідні рухи і переміщення.

Ергономічними аспектами проектування відеотермінальних робочих місць, зокрема, є: висота робочої поверхні, розміри простору для ніг, вимоги до того, що розташовує документів на робочому місці (наявність і розміри підставки для документів, можливість різного розміщення документів, відстань від очей користувача до екрану, документа, клавіатури і т.д.), характеристики робочого крісла, вимоги до поверхні робочого столу, можливість регулювання елементів робочого місця . Головними елементами робочого місця програміста є стіл і крісло. Основним робочим положенням є положення сидячи.

Робоча поза сидячи викликає мінімальне стомлення програміста. Раціональне планування робочого місця передбачає чіткий порядок і

постійність розміщення предметів, засобів праці і документації. Те, що потрібне для виконання робіт частіше, розташоване в зоні легкої досяжності робочого простору.

Моторне поле – простір робочого місця, в якому можуть здійснюватися рухові дії людини.

Максимальна зона досяжності рук – це частина моторного поля робочого місця, обмеженого дугами, описуваними максимально витягнутими руками при русі їх в плечовому суглобі.

Оптимальна зона – частина моторного поля робочого місця, обмеженого дугами, що описуються передпліччями при русі в ліктьових суглобах з опорою в точці ліктя і з відносно нерухомим плечем.

Оптимальне розміщення предметів праці і документації в зонах досяжності:

– дисплей розміщується в центрі;

– системний блок розміщується в передбаченій ніші столу;

– клавіатура – в зоні попереду;

– «миша» – в зоні справа;

– сканер в зоні зліва;

– принтер знаходиться в зоні справа.

Документація: необхідна при роботі – в зоні легкої досяжності долоні, а у висувних ящиках столу – література, невживана постійно.

Для комфортної роботи стіл повинен задовольняти наступним умовам :

– висота столу повинна бути вибрана з урахуванням можливості сидіти вільно, в зручній позі, при необхідності спираючись на підлокітники;

– нижня частина столу повинна бути сконструйована так, щоб програміст міг зручно сидіти, не був вимушений підтискати ноги;

– поверхня столу повинна володіти властивостями, що виключають появу відблисків в полі зору програміста;

– конструкція столу повинна передбачати наявність висувних ящиків (не менше 3 для зберігання документації, лістингів, канцелярських обладнань).

– висота робочої поверхні рекомендується в межах 680-760 мм. Висота поверхні, на яку встановлюється клавіатура, повинна бути біля 650 мм.

Велике значення надається характеристикам робочого крісла. Так, висота сидіння над рівнем підлоги, що рекомендується, знаходиться в межах 420-550мм. Поверхня сидіння м'яка, передній край закруглює, а кут нахилу спинки – регульований. безпека оператор виробниче приміщення

Необхідно передбачати при проектуванні можливість різного розміщення документів: збоку від відеотерміналу, між монітором і клавіатурою і т.п. Крім того, у випадках, коли відеотермінал має низьку якість зображення, наприклад помітні мигтіння, відстань від очей до екрану роблять більше (біля 700 мм), ніж відстань від ока до документа (300-450 мм). Взагалі при високій якості зображення на відеотерміналі відстань від очей користувача до екрану, документа і клавіатури може бути рівним.

Положення екрану визначається:

– відстанню прочитування (0,6-0,7 м);

– кутом прочитування, напрямом погляду на 20 нижче горизонталі до центру екрану, причому екран перпендикулярний цьому напряму.

Повинна також передбачатися можливість регулювання екрану:

– по висоті +3 см;

– по нахилу від -10 до +20 щодо вертикалі;

– в лівому і правом напрямах.

Велике значення також надається правильній робочій позі користувача. При незручній робочій позі можуть з'явитися болі в м'язах, суглобах і сухожиллях. Вимоги до робочої пози користувача відеотерміналу наступні:

– голова не повинна бути нахилена більш ніж на 20;

– плечі повинні бути розслаблені;

– лікті – під кутом 80-100;

– передпліччя і долоні рук – в горизонтальному положенні.

Причина неправильної пози користувачів обумовлена наступними чинниками: немає хорошої підставки для документів, клавіатура знаходиться

дуже високо, а документи – низько, нікуди покласти руки і кисті, недостатній простір для ніг.

В цілях подолання вказаних недоліків даються загальні рекомендації: краще пересувна клавіатура; повинні бути передбачені спеціальні пристосування для регулювання висоти столу, клавіатури і екрану, а також підставка для рук.

Істотне значення для продуктивної і якісної роботи на комп'ютері мають розміри знаків, густину їх розміщення, контраст і співвідношення яскравості символів і фону екрану. Якщо відстань від очей оператора до екрану дисплея складає 60-80 см, то висота знака повинна бути не менше 3 мм, оптимальне співвідношення ширини і висоти знака складає 3:4, а відстань між знаками – 15-20% їх висоти. Співвідношення яскравості фону екрану і символів – від 1:2 до 1:15.

Під час користування комп'ютером медики радять встановлювати монітор на відстані 50-60 см від очей. Фахівці також вважають, що верхня частина відеодисплея повинна бути на рівні очей або трохи нижче. Коли людина дивиться прямо перед собою, його очі відкриваються ширше, ніж коли він дивиться вниз. За рахунок цього площа огляду значно збільшується, викликаючи обезводнення очей. До того ж якщо екран встановлений високо, а очі широко відкриті, порушується функція моргання. Це значить, що очі не закриваються повністю, не омиваються слізною рідиною, не одержують достатнього зволоження, що приводить до їх швидкої стомлюваності.

Створення сприятливих умов праці і правильне естетичне оформлення робочих місць на виробництві має велике значення як для полегшення праці, так і для підвищення його привабливості, позитивно впливаючою на продуктивність праці.

ВИСНОВКИ

У рамках кваліфікаційної роботи було розв`язано практичну задачу інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов, а саме розроблено програмне забезпечення для обліку комп’ютерної техніки Мішково-Погорілівської загальноосвітньої санаторної школи-інтернату.

Було сформульовано об`єкт роботи. Було виконано аналіз практичної задачі інженерії програмного забезпечення: розглянуті питання автоматизації обліку комп’ютерної техніки, пакетів прикладних програм та інструктивно-методичних матеріалів, виконано аналіз існуючих програмних продуктів. Здійснено постановку задачі на розробку програмного забезпечення. Розроблено програмне забезпечення для обліку комп’ютерної техніки Мішково-Погорілівської загальноосвітньої санаторної школи-інтернату, для чого були виконані наступні завдання: проведено аналіз вимог до програмного забезпечення, розроблена архітектура програмного забезпечення, розроблено проєкт програмного забезпечення, виконані реалізацію та тестування програмного забезпечення.

Також у рамках кваліфікаційної роботи представлено розділ з охорони праці.

Мета кваліфікаційної роботи досягнута, усі завдання виконані в повному обсязі.

Впровадження розробки забезпечило отримання необхідної інформації щодо комп’ютерної техніки за запитами користувачів та оперативне отримання аналітичних звітів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Освітні технології. Учні в комп'ютерізованому класі. URL: http://osvita.ua/school/method/5894/ (дата звернення: 20.03.2023).

2. Освітня технологія Росток. Навчальні програми. URL: https://rostok.org.ua/navchalni-prohramy/ (дата звернення: 20.03.2023).

3. Комп’ютери в школах – якими мають бути для кращих результатів навчання? URL: https://dixi.education/computers-in-schools-new-standards/ (дата звернення: 20.03.2023).

4. Hardware Inspector. Программы для учета компьютеров и автоматизации работы IT-отдела. URL: http://www.hwinspector.com/ (дата звернення: 07.04.2023).

5. Network Inventory Advisor. Professional Network Inventory Software. URL: https://www.network-inventory-advisor.com/ (дата звернення: 07.04.2023).

6. HPE OpenView. Integrated IT infrastructure management software. URL: https://www.hpe.com/us/en/integrated-systems/software.html (дата звернення: 07.04.2023).

7. OCS Invetory. Open computers and software inventory is an asset management solution. URL: https://ocsinventory-ng.org/ (дата звернення: 07.04.2023).

8. Фаулер М. UML. Основи. Короткий посібник по стандартній мові об'єктного моделювання. М.: Символ-плюс, 2018. 192 с.

9. Соммервилл И. Инженерия программного обеспечения. Вильямс, 2002. 624 с.

10. Ларман К. Применение UML и шаблонов проектирования. 2-е издание: Пер. с англ. М: Издательский дом "Вильямс", 2004. 620 с. ил.

11. Елементи моделі «сутність-зв'язок». URL: http://easy-code.com.ua/2012/09/elementi-modeli-sutnist-zvyazok-integraciya-dodatkiv-i-danix-bazi-danix-statti/ (дата звернення: 25.04.2023).

12. Інтерфейс. URL: http://sven.ua/ua/service/glossary/ukr/11/ (дата звернення: 28.04.2023).

13. Діаграма класів. URL: http://moodle.ipo.kpi.ua/moodle/mod/resource/view.php?id=28088 (дата звернення: 28.04.2023).

14. Діаграма взаємодії. URL: http://moodle.ipo.kpi.ua/moodle/mod/resource/view.php?r=16538 (дата звернення: 28.04.2023).

15. News, Status & Discussion about Standard C++. URL: https://isocpp.org/ (дата звернення: 30.04.2023).

16. Офіційний сайт NetBeans. URL: https://netbeans.apache.org/ (дата звернення: 30.04.2023).

17. Дані в центрі уваги. URL: https://microsoftaccess.com (дата звернення: 30.04.2023).

18. Про охорону праці: Закон України від 21.11.2002 р. № 229-IV. Дата оновлення: 04.02.2021. URL: https://zakon.rada.gov.ua/laws/main/2694-12#Text. (дата звернення: 17.05.2023).

19. Поняття, мета і завдання охорони праці. URL: http://ncpn.net.ua/oxrana_truda.html (дата звернення: 17.05.2023).

20. Санітарні норми мікроклімату виробничих приміщень (ДСН 3.3.6.042-99). URL: https://zakon.rada.gov.ua/rada/show/va042282-99#Text (дата звернення: 17.05.2023).

ДОДАТОК А

Технічне завдання на розробку програмного забезпечення для обліку комп’ютерної техніки Мішково-Погорілівської загальноосвітньої санаторної школи-інтернату

Вступ

Повна назва програмного продукту: "Програмне забезпечення для обліку комп’ютерної техніки Мішково-Погорілівської загальноосвітньої санаторної школи-інтернату"

Сфера застосування: Мішково-Погорілівська загальноосвітня санаторна школа-інтернат.

1. Підстави для розробки

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

2. Призначення розробки

2.1 Експлуатаційне призначення

Експлуатаційним призначенням є зберігання, перегляд, пошук, редагування та видалення інформації про комп’ютерну техніку, її програмне забезпечення, інструктивно-методичні матеріали, використання комп’ютерного часу, а також інвентаризації комп’ютерної техніки у базі даних.

2.2 Функціональне призначення

Функціональним призначенням є автоматизація обліку комп’ютерної техніки Мішково-Погорілівської загальноосвітньої санаторної школи-інтернату, яка покращить та полегшить обробку потрібної інформації.

3 Вимоги до програмного забезпечення

3.1 Вимоги до функціональних характеристик

3.1.1 Вимоги до складу виконуваних функцій

– авторизація користувача;

– введення, перегляд та редагування даних про комп’ютерну техніку;

– введення, перегляд та редагування даних про програмне забезпечення;

– введення, перегляд та редагування даних про інструктивно-методичні матеріали;

– введення, перегляд та редагування даних про використання комп’ютерного часу;

– формування інвентаризаційних відомостей;

– формуванні звітів.

3.1.2 Вимоги до організаційних вхідних та вихідних даних

Введення даних здійснюється за допомогою клавіатури та миши. Дані вводяться вручну або обираються з випадаючих списків.

Обмін інформацією між програмним забезпеченням і базою даних відбувається за допомогою файлів з довільним доступом типу .mdb.

Виведення даних здійснюється за допомогою пристроїв виведення даних (монітор та принтер).

3.2 Вимоги до надійності

Програмний продукт повинен нормально функціонувати при безперебійній роботі ПК. При виникненні збоїв в роботі, відновлення нормальної роботи повинне проводитися після перезавантаження програми.

Для забезпечення надійності інформації повинна використовуватися СКБД, що забезпечує цілісність транзакцій і несе відповідальність за цілісність інформації.

Система повинна продовжувати коректне функціонувати при втраті частини інформації. У випадку неможливого продовження коректної роботи має бути повідомлення про це.

У разі введення користувачем некоректної інформації система повинна повідомити про помилку і надати можливість виправити її.

3.3 Вимоги до умов експлуатації

Необхідний рівень підготовки користувачів: мінімальні навики в користуванні комп’ютером. Комп’ютер призначений для роботи в закритому опалювальному приміщені.

3.4 Вимоги до складу та параметрів технічних засобів

Система повинна задовольняти вказаним вимогам на комп’ютері наступної мінімальної комплекції:

– CPU: Intel(R) i3;

– RAM: 4 GB;

– HDD: 10 GB вільного місця.

3.5 Вимоги до інформаційної та програмної сумісності

Для повноцінного функціонування системи необхідно використовувати ОС Windows версії не нижче 7.

Для роботи системи необхідно СУБД Access не нижче ніж Access 2013.

3.6 Вимоги до маркування та пакування

Продукт буде упакований в електронний архів і мати найменування mpssbs.rar

4 Вимоги до програмної документації

Склад програмної документації повинен включати в себе:

– технічне завдання на розробку програмного забезпечення;

– текст програми;

– опис програми;

– інструкцію користувача;

– програму і методику випробувань програмного забезпечення.

5 Техніко-економічні показники

Не розраховуються в зв’язку з тим, що продукт розробляється в рамках кваліфікаційної роботи.

6 Стадії та етапи розробки

Стадії та етапи розробки представлені у таблиці А.1.

Таблиця А.1 – Стадії та етапи розробки

№ п/п	Назва етапів роботи	Терміни виконання
1.	Аналіз практичної задачі інженерії ПЗ	31.03.2023
2.	Аналіз вимог до ПЗ	07.04.2023
3.	Розробка архітектури ПЗ	21.04.2023
4.	Розробка проекту ПЗ	05.05.2023
5.	Реалізація та тестування ПЗ	12.05.2023

7 Порядок контролю та приймання

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

Приймання 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 протягом не більше ніж двох тижнів виправити зазначені помилки і оповістити замовника про повторне проведення перевірки.

ДОДАТОК Б

Текст програми

Головний модуль

```
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
#include "Unit2.h"
#include "Unit3.h"
#include "Unit4.h"
#include "Unit5.h"
#include "Unit10.h"
#include "Unit11.h"
#include<Dblogdlg.hpp>
//---------------------------------------------------------------------
-----------
#pragma package(smart_init)
#pragma resource "*.dfm"
TFMain *FMain;
//---------------------------------------------------------------------
-----------
__fastcall TFMain::TFMain(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------
-----------
//---------------------------------------------------------------------
-----------
void __fastcall TFMain::BtE1Click(TObject *Sender)
{
Close();
}
//---------------------------------------------------------------------
-----------
void __fastcall TFMain::BtDovClick(TObject *Sender)
{
FDov->Show();
}
//---------------------------------------------------------------------
-----------
void __fastcall TFMain::BtReqClick(TObject *Sender)
{
Form6->Show();
}
//---------------------------------------------------------------------
-----------
void __fastcall TFMain::BtReportClick(TObject *Sender)
{
```

```
FReport->Show();
}
//---------------------------------------------------------------------
-----------
void __fastcall TFMain::FormShow(TObject *Sender)
{
FMain->Top=261;
FMain->Left=474;
AnsiString una, pwd;
label: if (LoginDialog("Base", una, pwd))
{
if ((una=="Admin")&& (pwd=="1234"))
FMain->Show();

else{
ShowMessage("Невірна інформація");
goto label;
};
}
else Close();
}
//---------------------------------------------------------------------
-----------
Модуль довідники
#include <vcl.h>
#pragma hdrstop
#include "Unit2.h"
#include "Unit1.h"
#include "Unit3.h"
#include "Unit4.h"
//---------------------------------------------------------------------
-----------
#pragma package(smart_init)
#pragma resource "*.dfm"
TFDov *FDov;
//---------------------------------------------------------------------
-----------
__fastcall TFDov::TFDov(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------
-----------
void __fastcall TFDov::BtE2Click(TObject *Sender)
{
FMain->Show();
Hide();
}
//---------------------------------------------------------------------
-----------
//---------------------------------------------------------------------
-----------
void __fastcall TFDov::BtAddClick(TObject *Sender)
```

```
{
{
if (( TedNtyp->Text=="")||(TedProc->Text=="")||(TedChas-
>Text==""))
{ShowMessage("Якщо поля пусті");
TedNtyp->SetFocus();
}
else
if(MessageDlg("Додати
поле?",mtInformation,TMsgDlgButtons()<<mbYes<<mbNo,0)==mrYes)
ADOTable2->InsertRecord(ARRAYOFCONST((StrToInt(TedNtyp->Text),
       TedProc->Text, TedChas->Text)));
}
TedNtyp->Clear();
TedProc->Clear();
TedChas->Clear();
}
//---------------------------------------------------------------
-----------

void __fastcall TFDov::DBGrid2CellClick(TColumn *Column)
{
TedNtyp->Text=ADOTable2->Fields->Fields[0]->AsString;
TedProc->Text=ADOTable2->Fields->Fields[1]->AsString;
TedChas->Text=ADOTable2->Fields->Fields[2]->AsString;
}
//---------------------------------------------------------------
-----------
void __fastcall TFDov::BtDeleteClick(TObject *Sender)
{
 if(MessageDlg("Видалити
поле?",mtInformation,TMsgDlgButtons()<<mbYes<<mbNo,0)==mrYes)
ADOTable2->Delete();
}
//---------------------------------------------------------------
-----------
void __fastcall TFDov::BtEditClick(TObject *Sender)
{
ADOTable2->Edit();
       ADOTable2-> SetFields(ARRAYOFCONST((StrToInt(TedNtyp-
>Text),
       TedProc->Text, TedChas->Text)));
ADOTable2->Post();
}
//---------------------------------------------------------------
-----------
void __fastcall TFDov::BtClearClick(TObject *Sender)
{
 for(int i=0; i<ComponentCount; i++)
{ TEdit*TedNtyp=dynamic_cast<TEdit*>(Components[i]);
if (TedNtyp)
TedNtyp->Text=" ";
}
```

```
}
//---------------------------------------------------------------
-----------
void __fastcall TFDov::BtAddLabClick(TObject *Sender)
{
{
if (( TedNkab->Text=="")||(TedNazvaKab->Text=="")||(TedZavLb-
>Text==""||
(Edit1->Text=="")||Edit2->Text==""))
{ShowMessage("Не всі поля заповненні");
TedNkab->SetFocus();
}
else
if(MessageDlg("Додати
поле?",mtInformation,TMsgDlgButtons()<<mbYes<<mbNo,0)==mrYes)
ADOTable1->InsertRecord(ARRAYOFCONST((StrToInt(TedNkab->Text),
       TedNazvaKab->Text, TedZavLb->Text,Edit1->Text,Edit2-
>Text)));
}
TedNkab->Clear();
TedNazvaKab->Clear();
TedZavLb->Clear();
Edit1->Clear();
Edit2->Clear();
}
//---------------------------------------------------------------
-----------
void __fastcall TFDov::DBGrid1CellClick(TColumn *Column)
{
TedNkab->Text=ADOTable1->Fields->Fields[0]->AsString;
TedNazvaKab->Text=ADOTable1->Fields->Fields[1]->AsString;
TedZavLb->Text=ADOTable1->Fields->Fields[2]->AsString;
Edit1->Text=ADOTable1->Fields->Fields[3]->AsString;
Edit2->Text=ADOTable1->Fields->Fields[4]->AsString;
}
//---------------------------------------------------------------
-----------
void __fastcall TFDov::BtDeleteLAbClick(TObject *Sender)
{
if(MessageDlg("Видалити
поле?",mtInformation,TMsgDlgButtons()<<mbYes<<mbNo,0)==mrYes)
ADOTable1->Delete();
}
//---------------------------------------------------------------
-----------
void __fastcall TFDov::BtEditLabClick(TObject *Sender)
{
ADOTable1->Edit();
       ADOTable1-> SetFields(ARRAYOFCONST((StrToInt(TedNkab-
>Text),
       TedNazvaKab->Text, TedZavLb->Text,Edit1->Text,Edit2-
>Text)));
ADOTable1->Post();
```

```
}
//---------------------------------------------------------------------
-----------
void __fastcall TFDov::BtCleanLabClick(TObject *Sender)
{
for(int i=0; i<ComponentCount; i++)
{ TEdit*TedNkab=dynamic_cast<TEdit*>(Components[i]);
if (TedNkab)
TedNkab->Text=" ";
}
}
//---------------------------------------------------------------------
-----------
void __fastcall TFDov::DBGrid3CellClick(TColumn *Column)
{
TedINv->Text=ADOTable3->Fields->Fields[0]->AsString;
TedNazvPac->Text=ADOTable3->Fields->Fields[1]->AsString;
TedVersion->Text=ADOTable3->Fields->Fields[2]->AsString;
TedDataVst->Text=ADOTable3->Fields->Fields[3]->AsString;
TedVikon->Text=ADOTable3->Fields->Fields[4]->AsString;
}
//---------------------------------------------------------------------
-----------
void __fastcall TFDov::BtAddPZClick(TObject *Sender)
{
{
if (( TedINv->Text=="")||(TedNazvPac->Text=="")||(TedVersion-
>Text=="")
 ||(TedDataVst->Text=="") ||(TedVikon->Text==""))
{ShowMessage("Заповніть пусті поля");
TedINv->SetFocus();
}
else
if(MessageDlg("Додати
поле?",mtInformation,TMsgDlgButtons()<<mbYes<<mbNo,0)==mrYes)
ADOTable3->InsertRecord(ARRAYOFCONST((StrToInt(TedINv->Text),
       TedNazvPac->Text, TedVersion->Text,TedDataVst-
>Text,TedVikon->Text)));
}
TedINv->Clear();
TedNazvPac->Clear();
TedVersion->Clear();
TedDataVst->Clear();
TedVikon->Clear();
}
//---------------------------------------------------------------------
-----------
void __fastcall TFDov::BtDeletePZClick(TObject *Sender)
{
if(MessageDlg("Видалити
поле?",mtInformation,TMsgDlgButtons()<<mbYes<<mbNo,0)==mrYes)
ADOTable3->Delete();
}
```

```
//---------------------------------------------------------------
-----------
void __fastcall TFDov::BtEditPZClick(TObject *Sender)
{
ADOTable3->Edit();
       ADOTable3-> SetFields(ARRAYOFCONST((StrToInt(TedINv->Text),
       TedNazvPac->Text, TedVersion->Text,TedDataVst-
>Text,TedVikon->Text)));
ADOTable3->Post();
}
//---------------------------------------------------------------
-----------
void __fastcall TFDov::BtCleanPZClick(TObject *Sender)
{
for(int i=0; i<ComponentCount; i++)
{ TEdit*TedINv=dynamic_cast<TEdit*>(Components[i]);
if (TedINv)
TedINv->Text=" ";
}
}
//---------------------------------------------------------------
-----------
void __fastcall TFDov::Button1Click(TObject *Sender)
{
{
if (( EDinvN->Text=="")||(EDNtyp->Text=="")||(EDNkab->Text=="")
 ||(EdOZY->Text=="") ||(EDSystem->Text=="") ||(EDHD->Text=="")
 ||(EDInt->Text=="")||(EDLoc->Text=="") ||(EDMon->Text=="")
 ||(EDMouse->Text=="") ||(EDKey->Text=="")
 ||(EDDyn->Text==""))
{ShowMessage("Не всі поля заповненні");
EDinvN->SetFocus();
}
else
if(MessageDlg("Додати
поле?",mtInformation,TMsgDlgButtons()<<mbYes<<mbNo,0)==mrYes)
ADOTable4->InsertRecord(ARRAYOFCONST((StrToInt(EDinvN-
>Text),EDNtyp->Text,
EDNkab->Text,StrToInt(EdOZY->Text),EDSystem->Text,EDHD-
>Text,EDInt->Text,
EDLoc->Text,EDMon->Text,EDMouse->Text,EDKey->Text, EDDyn->Text)));
}
EDinvN->Clear();
EDNtyp->Clear();
EDNkab->Clear();
EdOZY->Clear();
EDSystem->Clear();
EDHD->Clear();
EDInt->Clear();
EDLoc->Clear();
EDMon->Clear();
EDMouse->Clear();
EDKey->Clear();
```

```
EDDyn->Clear();
}
//---------------------------------------------------------------------
-----------
void __fastcall TFDov::DBGrid4CellClick(TColumn *Column)
{
EDinvN->Text=ADOTable4->Fields->Fields[0]->AsString;
EDNtyp->Text=ADOTable4->Fields->Fields[1]->AsString;
EDNkab->Text=ADOTable4->Fields->Fields[2]->AsString;
EdOZY->Text=ADOTable4->Fields->Fields[3]->AsString;
EDSystem->Text=ADOTable4->Fields->Fields[4]->AsString;
EDHD->Text=ADOTable4->Fields->Fields[5]->AsString;
EDInt->Text=ADOTable4->Fields->Fields[6]->AsString;
EDLoc->Text=ADOTable4->Fields->Fields[7]->AsString;
EDMon->Text=ADOTable4->Fields->Fields[8]->AsString;
EDMouse->Text=ADOTable4->Fields->Fields[9]->AsString;
EDKey->Text=ADOTable4->Fields->Fields[10]->AsString;
EDDyn->Text=ADOTable4->Fields->Fields[11]->AsString;
}
//---------------------------------------------------------------------
-----------
void __fastcall TFDov::DBGrid5CellClick(TColumn *Column)
{
EDKodDis->Text=ADOTable5->Fields->Fields[0]->AsString;
EDNazvaDis->Text=ADOTable5->Fields->Fields[1]->AsString;
Edit8->Text=ADOTable5->Fields->Fields[2]->AsString;
EDKilLab->Text=ADOTable5->Fields->Fields[3]->AsString;
EDIMM->Text=ADOTable5->Fields->Fields[4]->AsString;
}
//---------------------------------------------------------------------
-----------
void __fastcall TFDov::Button5Click(TObject *Sender)
{
{
if (( EDKodDis->Text=="")||(EDNazvaDis->Text=="")||(EDKilLab-
>Text=="")
 ||(Edit8->Text=="")||(EDIMM->Text==""))
{ShowMessage("Не всі поля заповенні");
EDKodDis->SetFocus();
}
else
if(MessageDlg("Додати
поле?",mtInformation,TMsgDlgButtons()<<mbYes<<mbNo,0)==mrYes)
ADOTable5->InsertRecord(ARRAYOFCONST((StrToInt(EDKodDis->Text),
       EDNazvaDis->Text, Edit8->Text, EDKilLab->Text,EDIMM-
>Text)));
}
EDKodDis->Clear();
EDNazvaDis->Clear();
Edit8->Clear();
EDKilLab->Clear();
EDIMM->Clear();
}
```

```
//---------------------------------------------------------------
-----------

void __fastcall TFDov::Button6Click(TObject *Sender)
{
if(MessageDlg("Видалити
поле?",mtInformation,TMsgDlgButtons()<<mbYes<<mbNo,0)==mrYes)
ADOTable5->Delete();
}
//---------------------------------------------------------------
-----------
void __fastcall TFDov::Button7Click(TObject *Sender)
{
ADOTable5->Edit();
       ADOTable5-> SetFields(ARRAYOFCONST((StrToInt(EDKodDis-
>Text),
       EDNazvaDis->Text,Edit8->Text, EDKilLab->Text,EDIMM-
>Text)));
ADOTable5->Post();
}
//---------------------------------------------------------------
-----------
void __fastcall TFDov::Button8Click(TObject *Sender)
{
for(int i=0; i<ComponentCount; i++)
{ TEdit*EDKodDis=dynamic_cast<TEdit*>(Components[i]);
if (EDKodDis)
EDKodDis->Text=" ";
}
}
//---------------------------------------------------------------
-----------
void __fastcall TFDov::Button2Click(TObject *Sender)
{
if(MessageDlg("Видалити
поле?",mtInformation,TMsgDlgButtons()<<mbYes<<mbNo,0)==mrYes)
ADOTable4->Delete();
}
//---------------------------------------------------------------
-----------
void __fastcall TFDov::Button4Click(TObject *Sender)
{
for(int i=0; i<ComponentCount; i++)
{ TEdit*EDinvN=dynamic_cast<TEdit*>(Components[i]);
if (EDinvN)
EDinvN->Text=" ";
}
}
//---------------------------------------------------------------
-----------
void __fastcall TFDov::Button3Click(TObject *Sender)
{
ADOTable4->Edit();
```

```
       ADOTable4-> SetFields(ARRAYOFCONST((StrToInt(EDinvN-
>Text),EDNtyp->Text,
EDNkab->Text,StrToInt(EdOZY->Text),EDSystem->Text,EDHD-
>Text,EDInt->Text,
EDLoc->Text,EDMon->Text,EDMouse->Text,EDKey->Text, EDDyn->Text)));
ADOTable4->Post();
}
//---------------------------------------------------------------
-----------
void __fastcall TFDov::Button9Click(TObject *Sender)
{
{
if (( Edit3->Text=="")||(Edit4->Text=="")||(Edit5->Text=="")||
(Edit6->Text=="")||(Edit7->Text==""))
{ShowMessage("Не всі поля заповенні");
Edit3->SetFocus();
}
else
if(MessageDlg("Додати
поле?",mtInformation,TMsgDlgButtons()<<mbYes<<mbNo,0)==mrYes)
ADOTable6->InsertRecord(ARRAYOFCONST((StrToInt(Edit3->Text),
       Edit4->Text, Edit5->Text,Edit6->Text,Edit7->Text)));
}
Edit3->Clear();
Edit4->Clear();
Edit5->Clear();
Edit6->Clear();
Edit7->Clear();
}
//---------------------------------------------------------------
-----------
void __fastcall TFDov::DBGrid6CellClick(TColumn *Column)
{
Edit3->Text=ADOTable6->Fields->Fields[0]->AsString;
Edit4->Text=ADOTable6->Fields->Fields[1]->AsString;
Edit5->Text=ADOTable6->Fields->Fields[2]->AsString;
Edit6->Text=ADOTable6->Fields->Fields[3]->AsString;
Edit7->Text=ADOTable6->Fields->Fields[4]->AsString;
}
//---------------------------------------------------------------
-----------
void __fastcall TFDov::Button10Click(TObject *Sender)
{
if(MessageDlg("Видалити
поле?",mtInformation,TMsgDlgButtons()<<mbYes<<mbNo,0)==mrYes)
ADOTable6->Delete();
}
//---------------------------------------------------------------
-----------
void __fastcall TFDov::Button11Click(TObject *Sender)
{
ADOTable6->Edit();
       ADOTable6-> SetFields(ARRAYOFCONST((StrToInt(Edit3->Text),
```

```
       Edit4->Text, Edit5->Text,Edit6->Text,Edit7->Text)));
ADOTable6->Post();
}
//---------------------------------------------------------------------
-----------
void __fastcall TFDov::Button12Click(TObject *Sender)
{
for(int i=0; i<ComponentCount; i++)
{ TEdit*Edit3=dynamic_cast<TEdit*>(Components[i]);
if (Edit3)
Edit3->Text=" ";
}
}
//---------------------------------------------------------------------
-----------
```

Модуль звіти

```
#include <vcl.h>
#pragma hdrstop
#include "Unit4.h"
#include "Unit1.h"
#include "Unit2.h"
#include "Unit3.h"
//---------------------------------------------------------------------
-----------
#pragma package(smart_init)
#pragma resource "*.dfm"
TFReport *FReport;
//---------------------------------------------------------------------
-----------
__fastcall TFReport::TFReport(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------
-----------
void __fastcall TFReport::BtE4Click(TObject *Sender)
{
FMain->Show();
Hide();
}
//---------------------------------------------------------------------
-----------
void __fastcall TFReport::Button2Click(TObject *Sender)
{
QuickRep1->Preview();
}
//---------------------------------------------------------------------
-----------
void __fastcall TFReport::Button1Click(TObject *Sender)
{
  QuickRep2->Preview();
}
//---------------------------------------------------------------------
```

```
-----------
void __fastcall TFReport::Button3Click(TObject *Sender)
{
FDov->QuickRep2->Preview();
}
//---------------------------------------------------------------
-----------
Модуль запити
#include <vcl.h>
#pragma hdrstop
#include "Unit11.h"
#include "Unit10.h"
#include "Unit1.h"
#include "Unit2.h"
#include "Unit12.h"
#include "Unit13.h"
#include "Unit14.h"
#include "Unit15.h"
//---------------------------------------------------------------
-----------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm6 *Form6;
//---------------------------------------------------------------
-----------
__fastcall TForm6::TForm6(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------
-----------
void __fastcall TForm6::Button1Click(TObject *Sender)
{
Form5->Show();
Hide();
}
//---------------------------------------------------------------
-----------
void __fastcall TForm6::Button6Click(TObject *Sender)
{
FMain->Show();
Hide();
}
//---------------------------------------------------------------
-----------
void __fastcall TForm6::Button2Click(TObject *Sender)
{
Form7->Show();
Hide();
}
//---------------------------------------------------------------
-----------
void __fastcall TForm6::Button3Click(TObject *Sender)
```

```
{
Form8->Show();
Hide();
}
//---------------------------------------------------------------------
-----------
void __fastcall TForm6::Button4Click(TObject *Sender)
{
Form9->Show();
Hide();
}
//---------------------------------------------------------------------
-----------
void __fastcall TForm6::Button5Click(TObject *Sender)
{
Form15->Show();
Hide();
}
//---------------------------------------------------------------------
-----------
Модуль запиту “Дані по класу ”
#include <vcl.h>
#pragma hdrstop
#include "Unit10.h"
#include "Unit11.h"
//---------------------------------------------------------------------
-----------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm5 *Form5;
//---------------------------------------------------------------------
-----------
__fastcall TForm5::TForm5(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------
-----------
void __fastcall TForm5::Button1Click(TObject *Sender)
{
ADOQuery1->Active=false;
ADOQuery1->Parameters->ParamByName("p")->Value=Edit1->Text;
ADOQuery1->Active=true;
ADOQuery1->Open();
if (!ADOQuery1->RecordCount)
        {
                ShowMessage("Введіть існуючий номер кабінету ");
        }
}
//---------------------------------------------------------------------
-----------
void __fastcall TForm5::Button2Click(TObject *Sender)
{
```

```
Form6->Show();
Hide();
}
//---------------------------------------------------------------------
-----------
Модуль запиту " Наявність НМК "
include <vcl.h>
#pragma hdrstop
#include "Unit12.h"
#include "Unit11.h"
//---------------------------------------------------------------------
-----------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm7 *Form7;
//---------------------------------------------------------------------
-----------
__fastcall TForm7::TForm7(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------
-----------
void __fastcall TForm7::Button1Click(TObject *Sender)
{
Form6->Show();
Hide();
}
//---------------------------------------------------------------------
-----------
void __fastcall TForm7::Button2Click(TObject *Sender)
{
ADOQuery1->Active=false;
ADOQuery1->Parameters->ParamByName("d")->Value=Edit1->Text;
ADOQuery1->Active=true;
ADOQuery1->Open();

if (!ADOQuery1->RecordCount)
        {
                ShowMessage("Введіть існуючий код класу ");
        }
}
//---------------------------------------------------------------------
-----------
Модуль запиту " Перелік ПК по класам "
#include <vcl.h>
#pragma hdrstop
#include "Unit13.h"
#include "Unit11.h"
//---------------------------------------------------------------------
-----------
#pragma package(smart_init)
#pragma resource "*.dfm"
```

```
TForm8 *Form8;
//---------------------------------------------------------------------
-----------
__fastcall TForm8::TForm8(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------
-----------
void __fastcall TForm8::Button1Click(TObject *Sender)
{
Form6->Show();
Hide();
}
//---------------------------------------------------------------------
-----------
void __fastcall TForm8::Button2Click(TObject *Sender)
{
ADOQuery1->Active=false;
ADOQuery1->Parameters->ParamByName("a")->Value=Edit1->Text;
ADOQuery1->Active=true;
ADOQuery1->Open();
if (!ADOQuery1->RecordCount)
        {
                ShowMessage("Введіть існуючий номер кабінету ");
        }
}
//---------------------------------------------------------------------
-----------
Модуль запиту ” ПЗ класу ”
#include <vcl.h>
#pragma hdrstop
#include "Unit14.h"
#include "Unit11.h"
//---------------------------------------------------------------------
-----------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm9 *Form9;
//---------------------------------------------------------------------
-----------
__fastcall TForm9::TForm9(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------
-----------
void __fastcall TForm9::Button1Click(TObject *Sender)
{
Form6->Show();
Hide();
}
//---------------------------------------------------------------------
```

```
----------
void __fastcall TForm9::Button2Click(TObject *Sender)
{
ADOQuery1->Active=false;
ADOQuery1->Parameters->ParamByName("g")->Value=Edit1->Text;
ADOQuery1->Active=true;
ADOQuery1->Open();
if (!ADOQuery1->RecordCount)
        {
                ShowMessage("Введіть існуючий номер кабінету ");
        }
}
//---------------------------------------------------------------------
----------
Модуль запиту“Характеристики ПК”
#include <vcl.h>
#pragma hdrstop
#include "Unit15.h"
#include "Unit11.h"
//---------------------------------------------------------------------
----------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm15 *Form15;
//---------------------------------------------------------------------
----------
__fastcall TForm15::TForm15(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------
----------
void __fastcall TForm15::Button1Click(TObject *Sender)
{
Form6->Show();
Hide();
}
//---------------------------------------------------------------------
----------
void __fastcall TForm15::Button2Click(TObject *Sender)
{
if (ADOQuery1->Active==true )
ADOQuery1->Active=false;
ADOQuery1->Parameters->ParamByName("p")->Value=StrToInt(Edit1-
>Text);
ADOQuery1->Active=true;
if (ADOQuery1->RecordCount==0)
        {
                ShowMessage("Введіть існуючий інвентарний номер
"); }}
```

Додаток В

Опис програми

В даній кваліфікаційній роботі було розроблено програмне забезпечення для обліку комп’ютерної техніки Мішково-Погорілівської загальноосвітньої санаторної школи-інтернату. Програмне забезпечення дозволяє зберігання, перегляд, пошук, редагування та видалення інформації про комп’ютерну техніку, пакети прикладних програм та інструктивно-методичні матеріали, формування розкладу, а також інвентаризації комп’ютерної техніки.

В якості мови програмування була обрана об'єктно-орієнтована мова програмування C++, в якості середовища розробки – інтегроване середовище візуального програмування NetBeans IDE, для роботи з базою даних була обрана мова структурованих запитів SQL, в якості СУБД – СУБД Microsoft Access.

Система повинна задовольняти вказаним вимогам на комп’ютері наступної мінімальної комплекції:

– CPU: Intel(R) i3;

– RAM: 4 GB;

– HDD: 10 GB вільного місця.

Для повноцінного функціонування системи необхідно використовувати ОС Windows версії не нижче 7.

Для роботи системи необхідно СУБД Access не нижче ніж Access 2013.

Експлуатаційним призначенням є зберігання, перегляд, пошук, редагування та видалення інформації про комп’ютерну техніку, її програмне забезпечення, інструктивно-методичні матеріали, використання комп’ютерного часу, а також інвентаризації комп’ютерної техніки у базі даних.

Функціональним призначенням є автоматизація обліку комп’ютерної техніки Мішково-Погорілівської загальноосвітньої санаторної школи-інтернату, яка покращить та полегшить обробку потрібної інформації.

Функції, які виконує програмне забезпечення:

– авторизація користувача;

– введення, перегляд та редагування даних про комп’ютерну техніку;

– введення, перегляд та редагування даних про програмне забезпечення;

– введення, перегляд та редагування даних про інструктивно-методичні матеріали;

– введення, перегляд та редагування даних про використання комп’ютерного часу;

– формування інвентаризаційних відомостей;

– формуванні звітів.

Виконання цих функцій забезпечують відповідні модулі. Програмне забезпечення має зручний та зрозумілий інтерфейс, що дуже важливо тому, що працювати з програмою будуть працівники, в яких знання комп’ютера може бути на рівні звичайного користувача.

Для входу в систему користувач повинен підтвердити право доступу шляхом авторизації: введення логіну та паролю. Після отримання дозволу для входу в програму відкривається головна форма, в якій користувачу пропонується вибрати необхідну дію.

Додаток Г

Інструкція користувача

Запуск програми відбувається при завантаженні виконуваного файлу програми mpssbs.exe, який знаходиться на робочому столі комп`ютера.

Для входу в програму користувач повинен підтвердити право доступу шляхом авторизації (див. рис. Г.1).

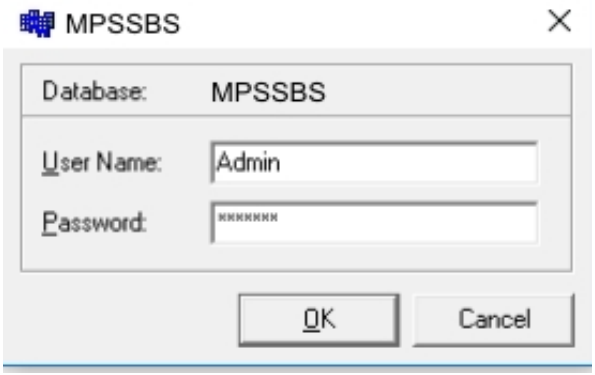

Рисунок Г.1 – Авторизація користувача

При успішній авторизації відкривається форма, яка містить головне меню програми (див.рис. Г.2).

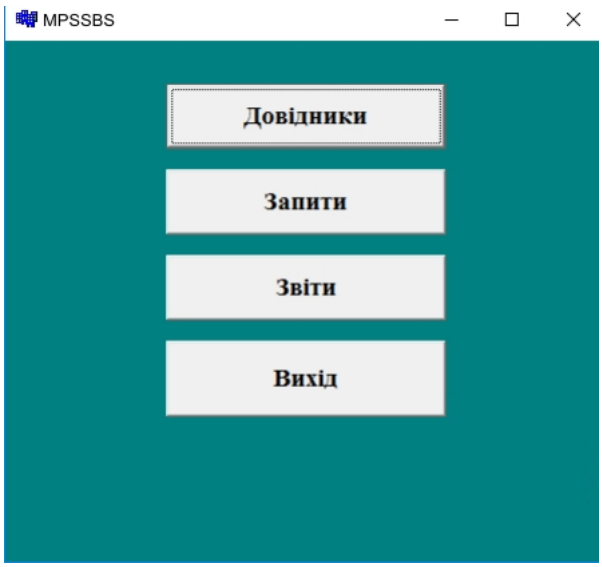

Рисунок Г.2 – Головне меню програми

Якщо вибрано кнопку «Довідники», з’являється форма вибору довідників (див.рис. Г.3).

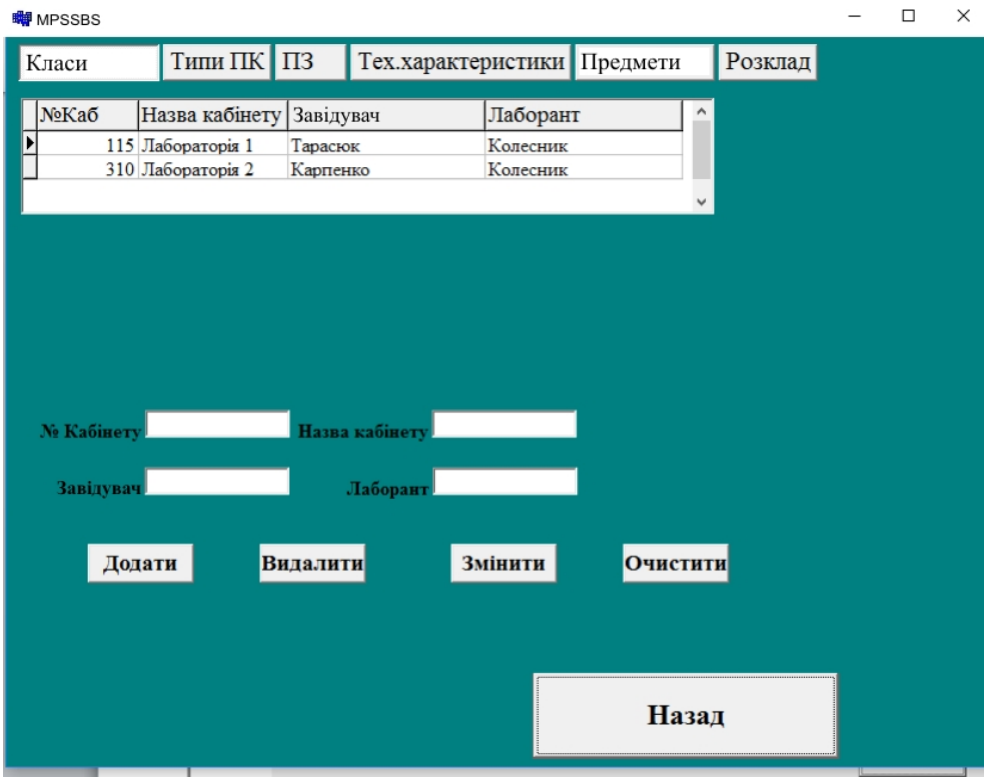

Рисунок Г.3– Форма вибору довідників.

При спробі занести в базу даних неповну інформацію видається повідомлення про помилку (див.рис. Г.4).

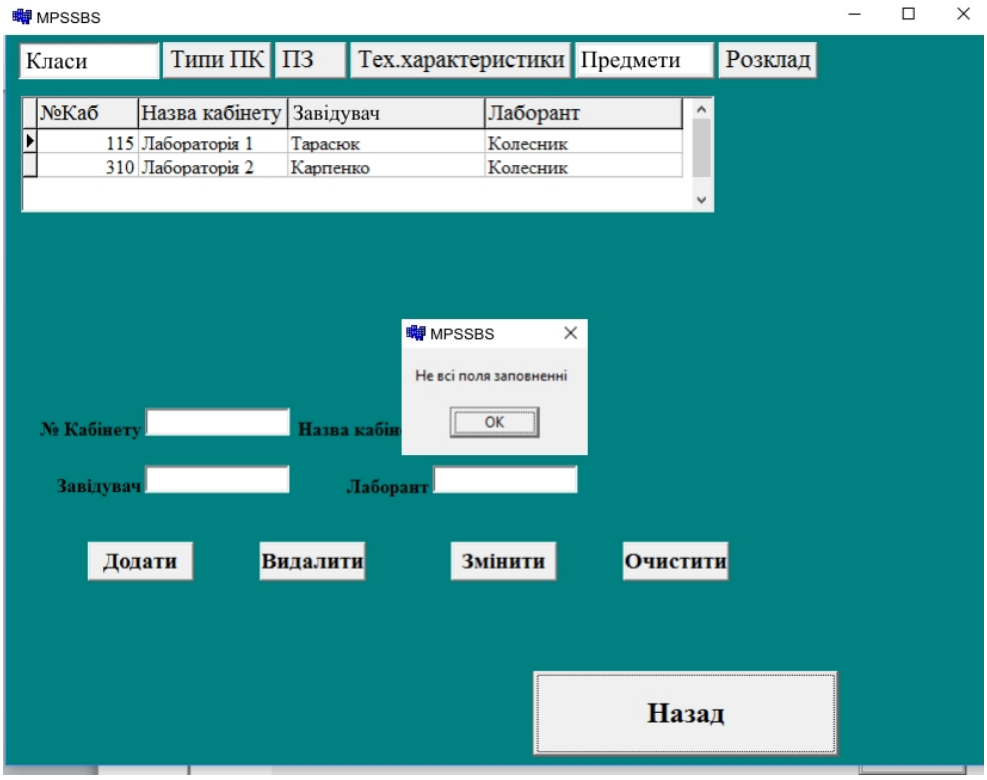

Рисунок Г.4 – Повідомлення про помилку вводу

Робота з іншими довідниками виконується аналогічно.

Якщо вибрано кнопку «Запити», з’являється форма вибору запитів (див. рис. Г.5).

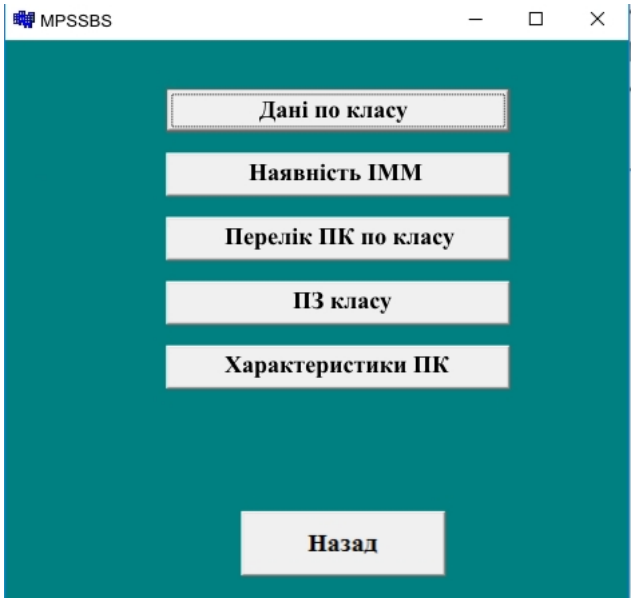

Рисунок Г.5 – Форма вибору запитів

При виборі запиту «Перелік ПК по класу» відкривається форма, в якій необхідно ввести номер кабінету (див. рис. Г.6).

Рисунок Г.6. – Запит «Перелік ПК по класу»

Робота з іншими запитами виконується аналогічно.

Якщо вибрано кнопку «Звіти», відкривається форма вибору звітів. Фрагмент звіту «Технічні характеристики ПК» наведено на рис. Г.7.

Інв№	ОЗУ	Система	Жорсткий диск	Інтернет	Монітор	Миші	Клавіатура	Динаміки
700	321	Windows 7	DB 13		Samsung Pro	Atech x5	Genius Pro	Так
701	222	Windows 7	DB 26	Так	Samsung Dig	Atech 51231B	Genius 3412A	
702	221	Windows 7	DB 621		LG Pro	Logitech 52123N	Genius	Так
703	333	Windows 7	DB 266	Так	LG Pro	Logitech 52123N	Genius	
709	231	Windows 7	DB 621	Так	Samsung 6	Logitech 52123N	Genius 3412A	
710	255	Windows 7	DB 321		Samsung 412D	Logitech 34213G	Genius 3234K	Так

Рисунок Г.7 – Фрагмент звіту «Технічні характеристики ПК»

Завершення роботи з програмою виконується при виборі кнопки «Вихід» головного вікна програми.

Додаток Д

Програма та методика випробування програмного забезпечення

Об'єкт випробування

Об’єктом випробування є програмне забезпечення для обліку комп’ютерної техніки Мішково-Погорілівської загальноосвітньої санаторної школи-інтернату.

Мета випробування

Метою проведення випробування є перевірка працездатності даного програмного продукту, перевірка відповідності характеристик та вимог розробленої програми, викладених у постановці задачі, приведення прикладу роботи та отримання відповідних навичок.

Вимоги до програми

При проведенні випробувань функціональні характеристики (можливості) програми підлягають перевірці на відповідність вимогам, викладеним у пункті «Вимоги до функціональних характеристик» постановки задачі.

Вимоги до програмної документації

1) Склад програмної документації, пропонованої на випробування

Програмна документація повинна включати в себе наступні документи:

а) Технічне завдання;

б) Текст програми;

в) Опис програми;

г) Інструкція користувача;

д) Програма та методика випробування програмного забезпечення.

2) Спеціальні вимоги

Спеціальні вимоги до програмної документації не висуваються.

Засоби і порядок випробувань

1) Технічні засоби, що використовуються під час випробувань

Склад технічних засобів, що використовувалися під час випробувань:

– CPU: Intel(R) i3;

– RAM: 4 GB;

– HDD: 10 GB вільного місця.

– Монітор з роздільною здатністю екрану – 1400 x 900 пікселів;

– Клавіатура 101/102-х клавішна рус / лат;

– Маніпулятор «миша».

2) Програмні засоби, що використовувалися під час випробувань

– ОС Windows версії 7;

– СУБД Access 2013.

3) Порядок проведення випробувань

Порядок проведення випробувань складається з перевірки вимог до функціональних характеристик програмного продукту та перевірки вимог до програмної документації.

Методи випробувань

1) Методика проведення перевірки вимог до функціональних характеристик програмного продукту

Перевірка працездатності програми виконується згідно з пунктом «Вимоги до функціональних характеристик» постановки задачі.

Перевірка вважається завершеною у разі відповідності складу і послідовності дій, при виконанні даної перевірки, вказаною в пункту постановки задачі.

В процесі тестування була перевірена функціональність програмного забезпечення для обліку комп’ютерної техніки Мішково-Погорілівської загальноосвітньої санаторної школи-інтернату. У таблиці Д.1, що наведена

нижче, вказано перелік випробувань основних функціональних можливостей.

Таблиця Д.1 – Перелік випробувань основних функціональних можливостей

№	Дія	Очікуваний результат	Результат перевірки	Зауваження
1	Перевірка правильності авторизації у програмі	Авторизацію пройдено при вводі правильного логіну та паролю	Виконано	
2	Перевірка правильності переходів на вкладку довідників	Відкривається вікно з довідниками	Виконано	
3	Перевірка правильності додання запису про клас	Програма повинна додати запис про клас	Виконано	
4	Перевірка правильності редагування даних класу	Програма повинна редагувати потрібний запис про клас	Виконано	
5	Перевірка правильності видалення даних класу	Програма повинна видалити потрібний запис про клас	Виконано	
6	Перевірка правильності формування звітів	Програма повина сформувати звіт з даними на тему, яку обрав користувач	Виконано	

2) Методика проведення перевірки вимог до програмної документації

Перевірка дотримання вимог програмної документації на програмний продукт виконується візуально представником служби, відповідальної за експлуатацію. У ході перевірки зіставляється склад і комплектність програмної документації, представленої розробником, з переліком програмної документації, наведеним у пункті «Склад програмної документації, пропонованої на випробування» цього документа.

Перевірка вважається завершеною у випадку відповідності складу та комплектності програмної документації, представленої розробником, переліком програмної документації, наведеному у зазначеному вище пункті.