Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: Розробка інформаційної системи контролю знань та оперативного обміну між групами студентів та викладачем кафедри з використанням СУБД HeidiSql у локальній мережі

Виконав(ла): студент 6 курсу, групи СПм-61
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)

(підпис) Біланик З.Б. (прізвище та ініціали)

Керівник (підпис) Петрик М.Р. (прізвище та ініціали)

Нормоконтроль (підпис) Стоянов Ю. М. (прізвище та ініціали)

Завідувач кафедри (підпис) Петрик М.Р. (прізвище та ініціали)

Рецензент (підпис) (прізвище та ініціали)

Тернопіль 2023р

РЕФЕРАТ

Кваліфікаційна робота магістра. Тернопільський національний технічний університет імені Івана Пулюя, кафедра програмної інженерії, спеціальність 121 «Інженерія програмного забезпечення». ТНТУ, 2023. Сторінок - 68, рисунків - 44 , додатків – 3, презентація. Тема: Розробка інформаційної системи контролю знань та оперативного обміну між групами студентів та викладачем кафедри з використанням СУБД HeidiSql у локальній мережі.

Метою виконання кваліфікаційної роботи магістра є проектування та реалізація десктопної версії програми для контролю якості освіти студентів та постійного зв'язку між користувачами, як однієї визначеної групи так і між окремими користувачами, а також надати інструменти обліку відвідуваності та графіку навчання по групах студентів. Робота демонструє результати проведеного аналізу предметної області для визначення вимог до функціонування та відображення елементів системи відповідно до обраної тематики системи, розглянуто інструменти які були залучені до розробки програми та особливості їх роботи та розгортання. ТАкож в роботі представлено опис спроектованої системи, визначено архітектуру, моделі бази даних, варіантів використання, для пояснення структури роботи системи та особливостей її розгортання та функціонування в залежності від обраної методології програмування. В результаті виконання кваліфікаційної роботи магістра створено інформаційну систему контролю знань з можливістю внутрішнього обміну повідомленнями та інформації між користувачами, а також відстеженням активності студентів та плану навчального процесу, з централізованим сервером даних для локальних мереж навчальної установи.

Ключові слова: C#, .NET, HeidiSQL, VIsual Studio, сервер,локальна мережа, СУБД.

ABSTRACT

Master's qualification work. Ivan Pulyuy Ternopil National Technical University, department of software engineering, specialty 121 "Software engineering". TNTU, 2023. Pages - 68, figures - 44, appendices - 3, presentation.

Topic: Development of an information system of knowledge control and operational exchange between groups of students and the teacher of the department using the HeidiSql DBMS in the local network.

The purpose of the master's qualification work is to design and implement a desktop version of the program to control the quality of student education and constant communication between users, both of one defined group and between individual users, as well as to provide attendance accounting tools and study schedules for groups of students. The work demonstrates the results of the analysis of the subject area to determine the requirements for the functioning and display of the system elements in accordance with the chosen subject of the system, the tools that were involved in the development of the program and the peculiarities of their operation and deployment are considered. Also, the work presents a description of the designed system, defines the architecture, database models, usage options, to explain the structure of the system and the features of its deployment and functioning depending on the selected programming methodology. As a result of the master's qualification work, an information system of knowledge control was created with the possibility of internal exchange of messages and information between users, as well as tracking of student activity and the plan of the educational process, with a centralized data server for local networks of the educational institution.

Keywords: C#, .NET, HeidiSQL, VIsual Studio, server, local network, DBMS.

ЗМІСТ

ВСТУП.....5
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ВИЗНАЧЕННЯ ВИМОГ.....7
1.1 Огляд систем аналогічної напрямленості..... 7
1.1.1 iStudiez pro.....7
1.1.2 Timetable.....9
1.2 Визначення функціональних та нефункціональних вимог до розробки програмного забезпечення.....12
2 ЗАСОБИ СТВОРЕННЯ ПРОГРАМНОГО ПРОДУКТУ.....14
2.1 Мова програмування C#.....15
2.2 Середовища програмування Visual Studio.....18
2.3 Фреймворк системних зв'язків. Entity Framework..... 20
2.4 Механізм адміністрації бази даних HeidiSQL..... 25
3 ПРОЕКТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ.....27
3.1 Моделі розробки програми.....27
3.2. Проектування варіантів використання та акторів програмної системи..... 29
3.3 Організація класів.....32
3.4 Метамодель системи та опис системи класів..... 35
3.5 Проектування бази даних.....45
3.6 Тестування програмного продукту..... 49
4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ.....59
4.1 Охорона праці..... 59
4.2. Забезпечення безпеки життєдіяльності при роботі з персональним комп'ютером.....62
ВИСНОВКИ.....67
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ..... 69
ДОДАТКИ.....71
Додаток А Технічне завдання..... 72
Додаток Б Слайди презентації.....73
Додаток В Публікація у науковому виданні..... 79

ВСТУП

З розвитком технологій та бази знань суспільство покращувати рівень власного комфорту в повсякденному житті включаючи всі сфери людської діяльності. У час зростання ролі інформаційних технологій очевидним є активне впровадження цифрових систем для покращення ефективності виконання практично будь-яких процесів та завдань, що дозволяє реформувати систему контролю та зробити її більш автоматизованою та скоротити ризики пов'язані з людським фактором, технологічними неточностями та інформаційним супроводом на виробництвах та підприємствах.

Як і для промислово-виробничого процесу цифровізація дозволяє якісно покращити контроль та виконання завдань для освітньо-культурних установ та організацій. Впровадження систем адміністрування в процес навчання надасть вирішення різноманітних ситуацій пов'язаних з управлінням та контролем освіти та рівня знань учнів, а також дозволить спростити документообіг та автоматизувати створення звітності проведених робіт.

Крім адміністрування використання інформаційних технологій дозволяє створювати різноманітні рішення, спрямовані на покращення навчального процесу, шляхом розробки інструментів для поглибленого ознайомлення та дослідження учнями освітніх матеріалів та оцінювання рівня засвоєння знань.

Отже, враховуючи актуальність та корисність використання цифровізації в освітньому процесі доцільно створити систему, яка виконуватиме завдання пов'язані з автоматизацією контролю та якості навчання.

Звідси, об’єктом дослідження роботи можна вважати способи використання інформаційних систем для вирішення проблеми з адмініструванням навчального процесу та похідних від нього завдань.

В якості предмета дослідження виступатиме програмна система автоматизації навчання, що виконуватиме контроль та якість освітнього процесу

для студентів, з можливістю подальшого аналізу отриманих результатів працівниками освітньої установи.

Отже, метою виконання кваліфікаційної роботи можна вважати проектування та подальшу розробку програмної системи, що виконуватиме автоматизацію адміністрації навчального процесу.

Для реалізації поставленої мети необхідно пройти наступні завдання:

- провести аналіз існуючих систем та програмних рішень для оцінки очікуваних функцій для системи, і визначення основних особливостей використання програм;
- спроектувати модель структури програми;
- реалізувати систему з використанням загально доступних інструментів та мереж розробки програмного забезпечення;
- провести тестування створеної системи.

Виконання поставлених завдань дозволить в кінцевому результаті створити робочу програмну систему для автоматизації освітнього процесу, для закладів будь якого розміру, направленості і спеціалізації. Кінцева система забезпечуватиме виконання необхідних функцій та їх коректної роботи для обробки та створення інформації пов'язаної з проведенням навчального процесу, що в результаті покращить швидкість аналізу і оцінювання прогресу навчання студентів та підвищить якість виконання шляхом скорочення адміністративно- організаційних обов'язків викладачів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ВИЗНАЧЕННЯ ВИМОГ

1.1 Огляд систем аналогічної напрямленості

Для визначення особливостей та вимог до розроблюваної системи доречно провести порівняльний аналіз на основі отриманих даних. З результатів яки можна визначити потреби в апаратному та програмному забезпеченні а також розробити проектні рішення які доречно залучити для покращення загального досвіду розробки та розгортання проекту. Тому було розглянуто декілька програм які реалізують такі функції що корисні для програми яку заплановано реалізувати.

1.1.1 iStudiez pro

Програма покликана допомогти студентові організувати власний навчальний процес, зробити календарні плани, створити робочі структури для ефективного навчального процесу. Іншими словами iStudiez pro представляє з себе систему для календарного планування на основі системи iOS.

ДЛя свого користувача програма надає максимальну свободу налаштувань програм з можливість залучення додаткових інструментів які встановлення на пристрої для спрощення налаштування та допомоги в відстеженні роботи програми, що до сповіщень та графіку який був розроблений без обмежень в рамках використання програми спрямованих на організацію процесу навчання. Будь які надбудови та коментарі можуть бути додані тільки в особливому порядку, тобто неможливо групувати та змінювати декілька елементів навіть якщо вони послідовно розміщені в розкладі планування за один раз, тільки для кожного індивідуально. Це компенсується великою кількістю полів, інструментів для деталізації кожного компонента та як найкращого вирізнення його за своєю ознакою що запобігає плутанині при наявності двох предметів зі схожою назвою

але різними напрямленостями, наприклад іноземна мова може бути як англійська, німецька чи французька та інші.

Крім того в глобальному масштабі програма дозволяє створювати маси календарних планів, які може змінюватися через задання особливих термінів та мов. Також система надає можливість створювати профілі викладачів, нотаток, коментарів, фотографій для розширення інформації про урок в графіку навчання, а також навіть створювати нагадування про проведення тої чи іншої пари чи заняття. Хорошим доповненням яке надає студентам система є можливість створювати графік екзаменів в окремому полі для наочності та запобіганню плутанини в розкладі.

Після налаштувань системи відповідно до навчального розкладу можна отримати повністю функціональний планувальний з можливістю динамічних змін при потребі. Крім того для тих осіб які люблять змінювати середовище відповідно до смаку, система дозволяє створювати та завантажувати власні іконки для розкладу, фон, ти фотографії в відповідні середовища для них, це дозволяє зробити програму максимально комфортним для користувача.

Крім планувальник пар можна відзначити наявність розділу домашніх завдань які дозволяють створювати записи, додаткові посилання та коментарі які будуть необхідні для виконання самостійної роботи з зазначенням інформації і в відповідності до заданих вимог. Проте ця вкладка не має можливості для синхронізації з основним розділом, тобто неможливо створити нагадування яке буде притягнуте до конкретного домашнього завдання без можливості конкретного визначення про що це завдання або до якого предмету воно відноситься, тільки день призначення і тоді не завжди він збігається з актуальним при додавання чи зміні контексту створеного завдання.

Іншим критичним моментом є поширенням планувальника на інші пристрої чи синхронізацію з ними. така проблема виникає коли користувач хоче перенести створений розклад чи поділитися із одногрупниками. В такі ситуації часто втрачаються дані в процесі, і не менш рідше збиваються налаштування для частини налаштованого плану. Це призводить до надмірного витрачання часу та

зусиль для відстеження втрат та відновлення даних, які можуть бути критичні для користувача в подальшому навчанні та призвести до можливих проблем через відсутність тих чи інших елементів інформації що були внесені в планувальник чи налаштовані сповіщення які були запущені . Це сильно підриває довіру до програми через постійну невпевненість в цілісності збереженої чи введеної в систему інформації. Через складність налаштування кожного окремого елемента

В останніх оновленнях система дозволяє створювати форму схожу на журнал балі для відстеження можливого кінцевого результату після завершення дисципліни. Як і розклад все налаштовується індивідуально для кожного елементу, як і введення. Проте система надає спеціальний калькулятор для обчислення остаточного балу за допомогою різних буквенно-численных градацій та формул які можуть задаватися і користувачем при потребі.

Програма надає вільний доступ до основних свої функцій планування безкоштовно проте для перенесення і синхронізації з іншими пристроями де є це система необхідний зв'язок з хмарним сховищем з якого і буде передаватися інформації, для його використання необхідна підписка яка є відповідно плантою. Проте з її використанням система може добре працювати за будь яких системах IOS які підтримують пересилання та поділ інформацією між собою.

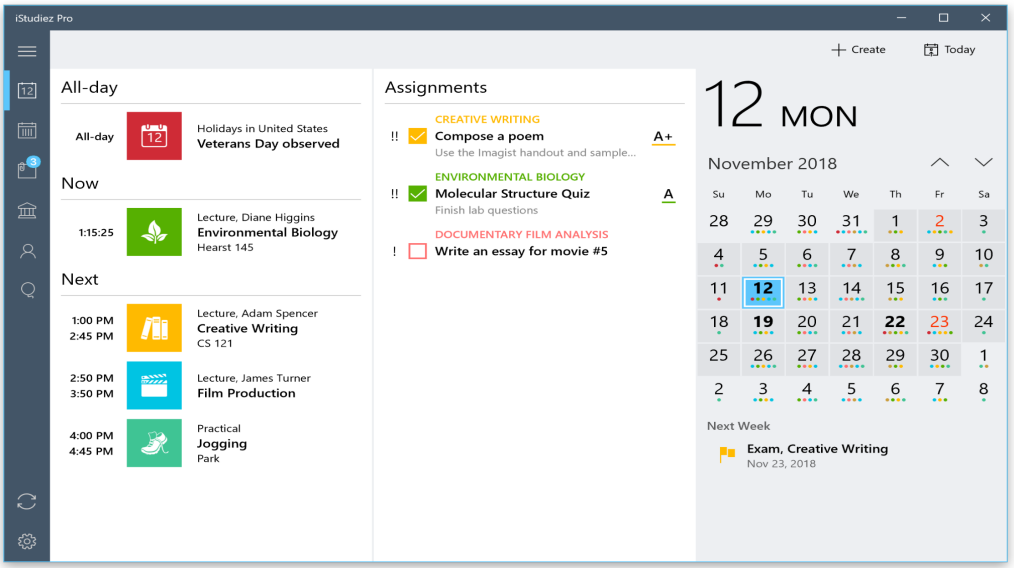


Рисунок 1.1.1.1 – Вигляд роботи програми iStudiez pro загальне вікно програми

1.1.2 Timetable

Ця програма реалізує систему електронного розкладу для систем Android. З її використанням користувача можу створювати власний план на тиждень, або навіть розробити власний графік навчання на весь семестр. Застосунок призначений для особистого використання в сферах навчання для учнів та студентів, тим самим дозволяючи оптимізувати розклад, для зручного відстеженя навчального дня.

Великою складністю використання програми, є нагромадження деталей, які необхідно відзначити для коректного відображення уроків в календарі, також відзначається неможливість створення мітки уроку, для більш ніж одного елемента для масового чи послідовного вводу даних. Це призводить до значних витрат часу користувача на заповнення на форматування вмісту графіку та складності відслідковування коректності заповнення, що може призвести до накладок в розкладі після затвердження створеного плану. Вагомим недоліком є відсутність коригування часу початку занять. Дуже зручним є наявність профілю предмету, який визначає ряд додаткової інформації для розширення інформації про предмет. Крім цього додаток визначає набір інструментів для створення сповіщень та налаштувань частоти звукових відтворень.

Система також не визначає більше ніж 5-денний робочий тиждень, шо не дозволяє створювати плани та розклад при потребі на вихідні дні. Програма має окремий вміст для самостійних та домашніх завдань дає можливість відповідно створювати для них сповіщення та редагувати з додатковими інструментами розробки. ПРоте відсутня синхронізація між основним розкладом і списком завдань, що змішує окремо заповнювати кожен з них. Також наявний список екзаменів який зберігає інформацію про іспити у вигляді списку сповіщень.

Крім цього система надає широкий спектр інструментів персоніфікації для користувача з метою найкомфортнішого використання всіх можливостей програми. Застосунок має можливість виконувати синхронізацію між пристроями,

для обміну даних, проте ця функція часто є причиною втрати даних або часткового затирання при переносі, також при її використанні виникають проблеми зі встановленням зв'язку що призводить до критичних закриттів програми в останніх версія системи.

Корисним є наявність інструментів віджетів програми які спрощують доступність інформації для перегляду студентові, та зменшують залежність програми, від необхідності активної роботи перевівши частину відображення для фонового виконання.

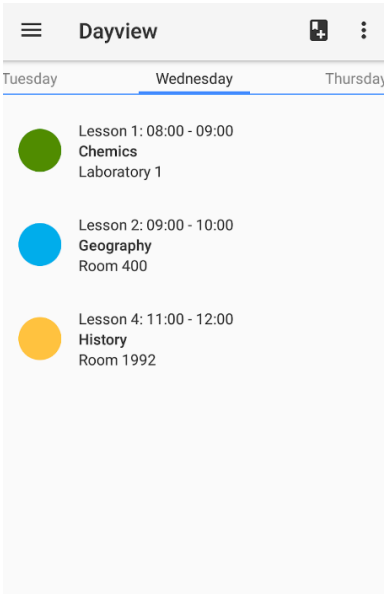


Рисунок 1.1.2.1 – Вигляд роботи програми Timetable в режимі денного графіку

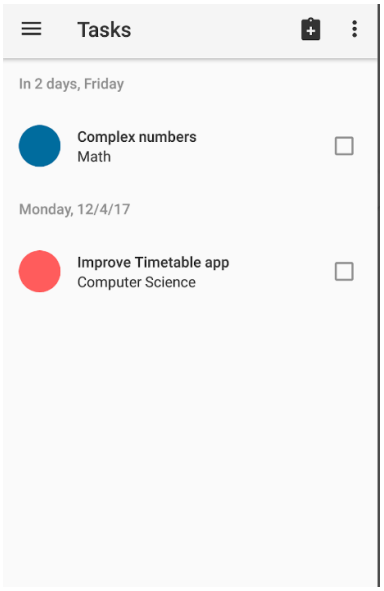


Рисунок 1.1.2.2 – Вигляд роботи програми Timetable з вкладкою завдань

1.2 Визначення функціональних та нефункціональних вимог до розробки програмного забезпечення

В результаті проведеного дослідження було визначено особливості, структуру, основні цілі та функції вирішення яких представляється в предметній області. Основна ідея роботи системи полягає в тому, щоб проводити контроль над процесом навчання, шляхом відстеження якості освіти, відвідуваності та обміну інформації між користувачами програми. Звідси можна означити функціональні вимоги що необхідно реалізувати та розглянути цільові інструменти для визначення інтерфейсу системи для користувачів. В результаті проведення досліджень, щодо схожих програмних систем було складено список вимог. Функціональні вимоги:

- виконання функції авторизації та реєстрація користувачів;
- проведення користувачем функцій додавання, видалення користувачів зміни їхньої інформації в системі про них;
- створення можливості затверджувати, видаляти та змінювати інформацію про оцінки користувачів які позначені як учні в системі;
- надання можливості користувачам які виконують роль адміністратора вносити зміни до навчальних планів, видаляти чи створювати нові екземпляри;
- для користувачів студентів дозволяти відстужувати власну відвідуваність, для викладачів вести облік відвідуваності студентів;
- адміністратору надається можливість створювати інформацію та проводити маніпуляція над ними;
- для всіх користувачів надання функцій для перегляду власного профілю та мати можливість змінювати дані в ньому;
- адміністратор має можливість змінювати доступність користувачів до інформації про групи та предмети відповідно до користувача;
- всі користувачі можуть обмінюватися повідомлення в мережі системи.

Нефункціональні вимоги

Цей тип вимог визначає особливості реалізації інтерфейсного вигляду програми для користувачі, способи розгортання, роботи, та обміну сповіщень від програми. Нефункціональні вимоги включають в себе апаратні потреби до роботи програми, визначає мінімальні вимоги які вимагаються від девайсу для коректного функціонування системи. До таких потреб включається:

- вибрана програма розробляється для виконання роботи на системах типу Windows для версії 10;
- передбачається реалізація інтерфейсу з легким інтуїтивним керуванням, з наявністю детальних підписів, для спрощення керуванням системи для користувача;
- програма повинна мати чіткі обмеження щодо використання елементів сторонніх програм або елементів з надмірними функціональними налаштуваннями, тобто зробити виконання функцій більш однозначними для спрощення розуміння команд на полі інтерфейсів, а також для простого визначення особливостей інформаційного супроводу системи;

Отже враховуючи визначені вимоги до програмного продукту, буде визначено основний функціонал, який виконуватиме програма, для користувачів, а також спроектовано відповідну структуру системи, для найоптимальнішого їх реалізації та виконання.

2 ЗАСОБИ СТВОРЕННЯ ПРОГРАМНОГО ПРОДУКТУ

Реалізація програми спрощується завдяки введення в проект різноманітних засобів та інструментів, які дозволяють автоматично генерувати або надають готові рішення для вирішення різних завдань які можуть виникнути в результаті процесу розробки. Завдяки цьому значно спрощується налаштування проектних файлів, системних конфігурацій, також звільняється час для розробника на відлагодження проекту шляхом зменшення зайнятості з налагодження системної сумісності програми з комп'ютерною та операційною системами.

Відповідно до поставлених умов доцільно використовувати програмні інструменти з широким діапазоном використання, та можливістю швидкого рефакторингу існуючих рішень. Система передбачатиме використання розгалуженої клієнтської системи з централізованим користувачем-сервером в локальній системі. Крім того варто врахувати необхідність застосування простих сторонніх програм які можуть застосовувати можливості фонових процесів для цільової операційної системи.

Також оскільки програма значною мірою пов'язана на обмін інформації доцільно залучити до проекту інструменти для адміністрування та збереження інформації, які можуть бути легко імплементовані в середовище розробки з мінімальними вимогами від системи розгортання. Крім доцільно враховувати те, що передбачається централізоване збереження інформації з різних пристроїв які є носіями однієї програми в одній локальній безпровідній мережі. Звідси встановлюється можливість створення аналогу архітектури типу клієнт-сервер який реалізовуватиметься через клієнт-сервер та окремий клієнт з урахуванням можливості багатопотокового підключення при різній кількості та часових тривалостях, відповідно до потреб користувачів системі в визначеній системою мережі.

Враховуючи визначені особливості системи що розробляється вирішено включити наступні інструменти в процес створення програмного продукту, зокрема:

- Створення програмного коду відбуватиметься мовою програмування C# з включенням розширення .Net;
- Використання інструментів середовища реалізації програмних рішень Visual Studio, версії 2019;
- Включення основних інструментів ADO.NET з розширенням пакетами розробки Entity Framework;
- Для центрального сховища даних використовуватиметься система управління базами даних HeidiSql яка використовує принципи MySQL;
- Створення і тестування системи відбуватиметься з використанням інструментів операційної системи Microsoft Windows 10;

2.1 Мова програмування C#

Відповідно до вибраних інструментів код програми розроблятиметься з нахилом на використання алгорітмів що найкраще працюватимуть з особливостями мови C#. Враховуючи її широку поширеність та наявність великої кількості різноманітних програмних рішень різних алгоритмів, було залучено в процес розробки і пакетне розширення фреймворку .Net Framework. Цей пакет значно спрощує побудову функціональних залежностей та реалізує найбільш поширені структури для легкої реалізації багатьох програмних рішень та ситуацій з якими може стикатися розробник. Застосування розширених функцій .Net дозволяє також зменшити змістовне навантаження та запобігти дублювання вже існуючих механізмів, що добре відзначиться на швидкості виконання реалізацій та програмуванню запланованих функцій системи. Крім цього цей пакет дозволяє

залишити достатньо простору для майбутнього розширення чи реорганізації проекту шляхом використання динамічних елементів, які легко застосовуватимуться з наявними рішеннями.

Рисунок 2.1.1 – Логотип C#

До особливості мови можна віднести строгу типізацію елементів проекту, поліморфізм, перевантаження операторів, використання вказівників на елементи класів та типів, це та інші особливості було унаслідував від синтаксису та логіки мов програмування C++ і Java. Крім цього C# в своїй структурі змогла уникнути деякі проблеми своїх попередників які пов'язані з можливістю виникнення критичних виключень через наявність чи створення дестабілізуючих елементів в коді програми цим самим запобагіючи краху системі в моментах їх виникнення.

Реалізація процесу розгортання проектів в мові C# проводиться з використанням компонентів реалізації системи Common Language Runtime вне залежності від середовища розробки програми. Вона представляє собою механізм емулятор віртуальної машини яка в своїй основі комбінує структурує протоколи реалізації програмних рішень для будь якої мови визначеної пакетом .NET Framework за власне визначеними шаблонами. Така поведінка можлива через те що система Common Language Runtime проводить процес перетворення готового коду програми в байт-код мови IL. Цей код в подальшому комплюється і з його допомогою написані функції директиви та команди можуть застосовуватися в будь

якому середовищі та системі що використовує байт-код через цим самим уніфікуючі та розширюючи можливості розробника для кросплатформенності власної програми. Крім цього така реалізація системи розгортання необхідна для коректної синхронізації з пакетами .NET Framework які реалізовані та працюють винятково в такому середовищі. Завдяки цьому також є можливість цим самим роблячи програмні реалізації сумісні з іншими компонентами які можуть бути підключені з використанням вищевказаного пакету та її класів бібліотек.

Програмування з використанням мови C# позиціонується як мова творення коду на для розробки прикладного рівня в системі Common Language Runtime звідси виникає необхідність коректного налаштування самої програми для правильного процесу реалізації та розгортання програми в визначеному користувачем середовищі. Структура кінцевого результату конвертації вхідного коду через використання Common Language Runtime залежать від особливостей мови програмування на якій проводиться розробка та від додаткових функціональних пакетів які були підключені користувачем для вирішення тих чи інших завдань програми. Звідси можна судити про високу залежність системи розгортання від рівня розвитку її у відношенні до складності та насиченості самої програми проектними реалізаціями та складністю алгоритмів програми що застосовуватиметься в ній. Це дозволяє збалансувати ефективність розробки та зменшити залежність від використання надлишкових елементів в системі для встановлення оптимального часу для роботи самої програми та визначити найкращу тривалість розгортання системи на цільовій машині.

Переваги використання C# :

- Велика перевага мови C# в її об’єктно-орієнтованій, це дозволяє створювати складні програмні реалізації для багатоцільового використання без повторень та надмірних деталізацій які корисні для створення будь-яких програм;
- Властивість строгої типізації дозволяє забезпечити безпечне розгортання програми та уникнуть критичних результатів роботи програми при

некоректній роботі з програмою та встановити передбачувані повідомлення що допоможуть визначити неполадки та виправити їх ;

- В своїй структурі C# володіє інструментами для автономного контролю вивільнення пам'яті за убезпечить розробників від нагромадження великої кількості “сміття” шляхом видалення його з певним проміжком часу. Цей механізм надає можливості зменшити складність звільнення пам'яті та знімає відповідальність за її реструктуризації з розробника;
- Мова програмування має багато створених пакетів які дають можливість використовувати готові програмні рішення та пришвидшити утворення більш складних програмних реалізацій;
- ДОзволяє просто змінювати ключі компіляції програми в середовищі та створює виконувані файли та бібліотеки компонентів як .NET через які можу відбуватися виклик раніше згенерованих файлів без необхідності створювати нові ланцюжки програмного розгортання та дозволяють проводити певне керування кодом як в ActiveX;

2.2 Середовища програмування Visual Studio

Для використання мови програмування C# існують різні програмні реалізації середовища в якому розробник можу просто створити власний програмний проект своєї програми, якість підходять краще які гірше, в залежності від потреб, спеціалізації та платформи самої програми та досвіду розробника. ДЛя реалізації програми було вирішено вибрати середовище програмування Visual Studio версії 2019.

Рисунок 2.2.1 – Логотип Visual Studio

Ця програма володіє значною серією програм, яка значно покращилася в плані інструментального наповнення та репутаційних піднесень з часу її першого релігу. Середовище призначене для реалізації різноманітних програмних рішень з великими проектними можливостями для програм різної специфіки. Система в своїй основі включає набір інструментів для створення, реалізації та розгортання проектів різного типу відповідно до наявних налаштувань. До них належать різноманітні драйвери, розширення, доповнення для комфорту та розширення основних функцій, також програма дозволяє створювати та використовувати систему паралельної розробки між пристроями, таких як Git як для локальних мереж так і для веб версій. Крім того система надає широкий набір методів для швидкої встановлення і швидкої роботи з базами даних, що надає просту та ефективну роботу для багатьох інформаційно орієнтованих систем, підтримується системи різних типів, як локальних так і віддалених серверів. Середовище володіє широким спектром інструментів для відлагодження, відстеження помилок та системи синтаксичних перевірок спрямованих на підтримку коректності наз елементів в системі. Visual Studio надає підтримку в розробці та розгортанні програм різного типу консольні, віконні програми(WinForm, Xamarin), створення веб програм, служб та сторінок, проектування та становлення служб для Windows, та багато інших. Широкий інструментарій доступних типів проектів дозволяє легко створювати кросплатформенні додатки зі збереженням логіки та звязку з програмними компонентами, незважаючи та різноманітність цільових машин

розгортання. Середовище вибрано враховуючи хорошу надійність, простота використання, доступність для власних проектів, наявність багатьох інструментів, наявність інструментів контролю версій та сумісність в роботі з базами даних та службами системи Windows. Також значною перевагою системи є можливість створювати поєднанні проекти для об'єднання роботи програм в залежності від шляху виклику в програмах, можливість відклагоджувати програми, а також наявність ефективного редагування коду та інструментів контролю критичних шляхів завершення роботи програми та деталізований генератор шляху помилок.

2.3 Фреймворк системних зв’язків. Entity Framework

Розширений пакет функцій програми ADO.NET для мови програмування C# дозволяє в поєднанні з Entity Framework створювати в програмних застосунках об'єктно-орієнтованого типу, інструменти для встановлення та контролю зв'язків з сховищами інформації, зокрема з базами даних як в локальній мережі так і з віддаленим доступом. Використання цього інструментів Entity Framework з поєднанням потужностей системи Visual Studio створює можливість відображення структури баз даних, додатковими можливостями, щодо маніпуляцій та розширення можливостей роботи із сховищем інформації. Це призводить до створення зв’язків з різними джерелами забезпечуючи безперебійний контакт програмної системи з самою базою, її елементами та об'єктами без використання інструментів відображення та роботи інтерфейсного застосунку цільового сховища інформації. Цим самим зменшуючи нагрузку розробника на контроль та адміністрування бази даних окремо від самого програмного проекту.

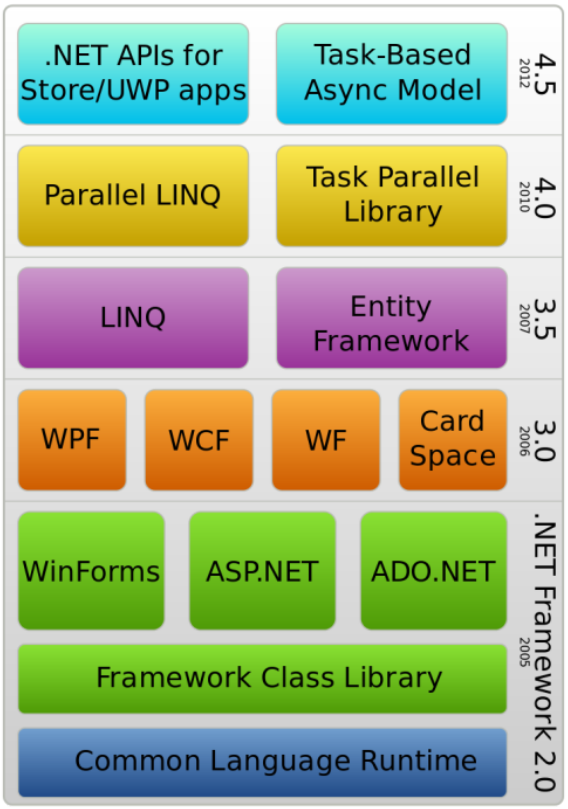


Рисунок 2.3.1 — Схема градації компонентів .NET Framework

Система .NET Framework включає в себе безліч програмних доповнень та функціональних розширень, які якісно збільшують ефективність та швидкість розробки будь-якого програмного продукту. Кожний новий елемент включається в систему з виходом нового розширення версій системи .NET. Наступні елементи складають частину компонентів фреймворку:

– Середовище виконання, розгортання та компіляції програмних систем, виключно для фреймворку .NET, такого типу як Common Language Runtime, включають також додаткові драйвера для .NET Core а саме системи розгортання CoreCLR і CoreRT ;

– Набір пакетів функціональних рішень, класів бібліотек з розширеннями, що реалізують класи системи типів для NETStandard, з додатковими можливостями для розширення, доповнення і модернізації нових чи існуючих елементів системи. Ці пакети дають можливість розширити можливості розробки програмних алгоритмів з використанням поглиблених інструментів роботи з кодами та типами даних, включаючи розробку інтерфейсів та систем швидкого

розгортання, та можливістю додавання додаткових сторонніх елементів до розробки системи;

− Набір платформ для розробки графічних рішень для користувацького відображення системи, для різноманітних платформ як веб систем так і персональних комп'ютерів та типів девайсів;

Рисунок 2.3.2 – Логотип методології Entity Framework

Особливості роботи методології Entity Framework представляється через реалізацію відображення представлень компонентів сховища даних в вигляді об'єкти зі складеними атрибутами або множини об'єктів. Враховуючи те що система побудована на взаємодії з елементами тобто об'єктами то для них справедливі ті самі обмеження і властивості як і для функціональних елементів та компонентів програм об'єктно-орієнтованого типу. Зокрема наступні твердження є справедливими для визначення структури EF:

- При описі будь-якої сутності що застосовується в процесі програмування системи справедливе визначення ряду елементів які описують характеристики сутності що будуть ідентичні об'єкта який існує в реальному світі
- Будь-який об’єкт і його елементи що визначають його характеристики можуть виписуватися з використанням як і простих типів даних так і з допомогою створених розробником складених структур типів.

- Структури сутності можуть бути описані різними типами та кількістю характеристик з допомогою яких можна відрізнити унікальність того чи іншого об'єкта від схожих елементів наявних в системі.
- Елементи системи мають можливість створювати різноманітні зв'язки між собою та іншими сутністями, наперед визначеними розробником та обмежені наявністю посилання на об'єкт розміщення або іншими можливостями відстеження місця компонента.

Корисним в системі Entity Framework є наявність реалізації системи створення та використання скорочених функціональних програмних реалізацій конструкцій LINQ які використовують для спрощення ініціалізації обробки та вибірки даних за різними специфіками, джерелами та іншими особливостями розробки програмних систем без виключення їх подальшої модернізації, включаючи роботу з базами даних, посиланнями, тощо.

Як особливість в системі Entity Framework відносять можливість створення та відображення моделей суб'єктів даних що дозволяє відтворювати фізично дані та змінювати через використання інструментів системи. Entity Framework надає можливість створювати моделі порівняння для вибраних класів об'єктів, таблиць та інших структурних елементів з цільової бази даних яка містить інформацію необхідну для якісного функціонування програми. Кожна модель формується з поєднання декількох робочих рівнів які формують всі особливості для коректної роботи об'єкта в середовищі системи. Виділяють наступні опорні рівні функціонування моделей даних: концептуальний, рівень сховища і рівень зіставлення.

Концептуальний рівень дозволяє проводити маніпуляції що відповідають пошуку та визначення необхідних даних з визначених класів або сутностей з використанням можливостей системи яка розробляється.

Рівень сховища визначає простір який буде відведено для збереження інформації, також формує робочий буфер який може легко змінити дані після внесення змін.

Рівень зіставлення виступає як проміжний між двома попередніми та визначає шлях та напрямку взаємодії з інформацією через визначені дані введені чи отримані в концептуальномі рівні а саме створюючи відповідності з властивостями класу суті і відповідною для неї таблицею. Цим самим рівнем і визначається спосіб взаємодії між класом сутності програми та відповідною таблицею даних з їх властивостями.

Варто зазначити що Entity Framework надає розробникам можливості вибирати шляхи реалізації та роботи з інформацією баз даних через наступними методами:

- Database first це метод що включає в себе попередньо визначення особливостей системи що розробляється та відповідно до неї створювати об'єкти та сутності бази даних, далі з використанням механізмів Entity Framework генерується EDMX-модель яка в подальшому використовуватиметься для розробки програми. Цей підхід застосовується проектувальниками програм та база жанихвикористовується в основному, проектувальниками баз даних. Підхід вимагає реалізації чіткого визначення обмежень бази даних, коректності синтаксису та визначення строгого порядку виконання SQL запитів, цей метод не впливає на написання коду програми;
- Model first передбачає попереднє створення EDMX-моделі з використанням засобів середовища розробки і з використанням отриманої структури генерує проектну структуру бази даних. З використанням цього підходу обмежується вплив SQL запитів та вплив програмних реалізацій на початкову структуру бази даних та встановлює максимальну достовірність до визначених властивостей в моделі та процесу розробки структури в цілому. Цей метод має популярність серед програмних архітекторів.
- Code first зосереджується на програмних реалізаціях класів звязку з базою даних включаючи використання інструментів автоматичної генерації класів сутностей та зв'язків між об'єктами. Поширено використовується між фахівцями програмістами.

2.4 Механізм адміністрації бази даних HeidiSQL

Враховуючи необхідність зовнішнього відстеження змін інформації та постійного моніторингу коректності результатів обробки інформації що була отримані чи введена через програму доцільно використовувати незалежну систему сервер, що буде суміжна з середовищем розробки та буде представляти кінцеву мету для отримання і переміщення даних. Оскільки передбачено створення стільникової програми в локальній мережі яка не передаватиме даны за її межі доцільно використати інструмент який буде застосовуватися для закритих мереж а також простий у своєму використанню. Тому було обрано продукт, що може працювати з різними конфігураціями як MySQL так і SQL серверами та її аналогами, а саме систему управління базами даних HeidiSQL. Ця програма сумісна з багатьма представленнями контролю баз даних і можу створювати багато представлень в один і той самий час не впливаючи на несподівану зміну результатів для іншого користувача. Система дозволяє працювати в двох режимах через інтерфейс і консольно. В інтерфейсі є багато інструментів з адміністрування як інформації так і структури бази, будь-які зміни що містять проблемний синтаксис будуть або відмінені або вимагатимуть перегляду доки не буде їх виправлено.

Система також може формувати власні зєднання для створення резервних шляхів для убезпечення даних, проте це потребуватиме деяких налаштувань. Простота реалізації функціональних рішень програми дозволяє легко синхронізувати потік інформації між середовищем розробки та самою базою, а також надає можливості для створення власних винятків для підвищення якості безпеки передачі даних, але це вже залежить від майстерності архітекторів бази даних.

HeidiSQL використовується як база даних реляційного типу, що включає наявність всіх базових функцій що підтримуватимуть роботу з таким типом структури роботи. Також варто зазначити що через те що ця система спеціалізується на локальних базах даних, вона надає більший простір для

створення більш громістких але добре розподілених таблиць-сутностей, які дозволяють просто проводить адміністрування бази та провести тестування та збільшити межі навантаження під час тестування програмного продукту. Враховуючи вище зазначені переваги і напрямленість само програми доречно включити такий інструмент в процес розробки функціональних та інформаційних зв'язків та формування структури збереження інформації для програми.

Рисунок 2.4.1 – Система HeidiSQL

3 ПРОЕКТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ

3.1 Моделі розробки програми

Розробляючи програмний продукт виникає необхідність створювати додатки з певною послідовністю та з використанням перевірених практик для спрощення процесу розробки програмних застосунків та приведення процесу роботи до певного стандарту. Розробка програмного забезпечення передбачає процес проектування та реалізації програмних рішень та алгоритмів для побудови життєздатної системи, цей процес по іншому називають моделлю життєвого циклу. Структура такої моделі переважно складає порядковість виконання певних етапів спрямованих на дослідження предметної області і подальше перетворення отриманих результатів в проектні рішення та програмну реалізацію з подальшим тестуванням для формування життєздатного програмного продукту. В своїй суті ця модель складає певну структуру процесів, робіт, задач і розрахунків, що відображатимуть весь шлях для створення робочої програми, визначивши її сильні та слабкі сторони та передбачивши найоптимальніший шлях для її використання в кінцевому результаті.

Враховуючи набір функціоналу, напрямленість програми та середовище роботи для системи що розробляється доцільно використати методологію що забезпечуватиме можливість повернення до розробки різних частин програми, при оновленні чи доповненні даних отриманих від зовнішнього дослідження, для оновлення чи вилучення функціонального коду програми. Така система життєвого циклу також буде корисна при вирішенні проблем специфікації які можуть призвести до конфлікту логіки роботи програми.

Враховуючи потреби розробки програми було вирішено використати еволюційну модель життєвого циклу для розробки цільової програми та впровадження її в робочий стан.

Еволюційна модель за своїм визначення дозволяє проводити процеси програмування таким чином щоб, мати можливість на будь-якому етапі

повертатися до попереднього циклу розробки програми та в разі потреби доповнити загальну програмну структуру для покращення кінцевого продукту. Ця модель реалізує в своїй основі принцип інкрементної методології програмування принцип якої полягає в повторенні ланцюгів створення етапів розробки програмного продукту для досягнення виконання кінцевих вимог до програмного забезпечення, та розширює її. Доповнення полягає в більш точковій розробці блоків окремих блоків а не всієї структури, а й доповнити специфікацію та уточнити вимоги до остаточної роботи кінцевої програми. Це спрощує початок роботи над розробкою програми, та дозволяє пришвидшити початок роботи розробки через проведення менш детальних досліджень перед створенням початкових шаблонів програмних структур та реалізацій. При цьому специфікація може доповнювати в кожній ітерацій в залежності від отриманих результатів від розробки та досліджень роботи попередньо реалізованих компонентів програмної системи. В результаті після дослідження ефективності роботи цієї моделі для розробки систем з нестабільними компонентами як бездротові мережі, було вирішено що ця методологія найкраще підійде для реалізації поставлених завдань до програмної системи, та дозволить виправить можливі недоліки які можуть виникнути в результаті недостатнього визначення обмежень програми в її специфікації що убезпечить від надмірної роботи над виправленням помилок, що призводять до виконання надмірних прилавок в коді та значно продовжують процес розробки застосунку.

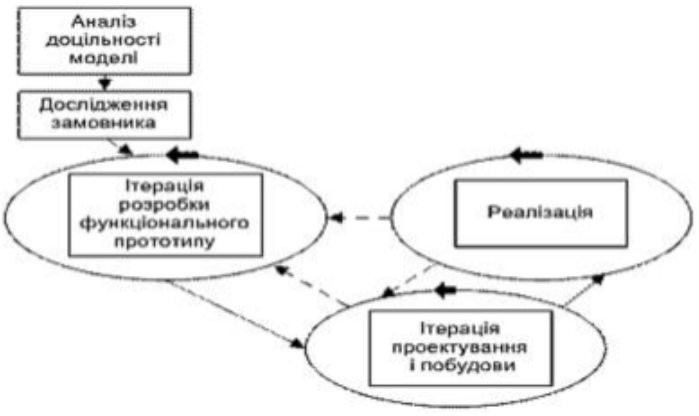


Рисунок 3.1.1 – Структура еволюційної моделі розробки програмного забезпечення

3.2. Проектування варіантів використання та акторів програмної системи

В результаті визначення вимог до системи які були визначені після дослідження схожих по направленості систем можна формувати варіанти використання та визначити їх основних актантів, що визначатимуть користувачів що будуть використовувати їх для своєї діяльності в системі.

Варіант використання представляє опис того як програма реагуватиме на певні дії користувачів, способи їх виконання, очікувані реакції, способи відображення затвердження реалізованих дій чи їх виділення у системі що розробляється. Також варіанти використання відображають всі функціональні дії, в результаті яких можна отримати зміну інформації, чи будь-яку її модифікацію. Для кожного варіанту може бути призначений один або декілька акторів, що буде використовувати його, звідси виникає необхідність визначити актанти та дії які вони виконуватимуть в програмній системі. До акторів відносяться наступні ролі в програмі:

- Учень;
- Вчитель;
- Адміністратор;
- Система-сервер;

Учень представляє користувача йому в системі надаються можливості що дозволяють переглядати дані системи пов'язані ним безпосередньо. А також проводити пошук, виконувати авторизацію і реєстрацію, обмінюватися листами та взаємодіяти з профілем користувача. Зокрема він може:

- Виконувати функції пошуку по навчальних здобутків;
- Змінювати інформації в профілі користувача;
- Перевіряти журнал відвідуваності, та освітній планан;

- Проводити листування між користувачами;
- Виконувати реєстрацію та авторизацію.

Вчитель виконує дії користувача з вищим ступенем можливостей ніж в студента, із включенням можливостей для змін щодо навчального плану, відвідуваності. Має можливість вносити бали студента та визначати напрямленість їх отримання. Підтримує також ті самі функції що визначені для учня, проте з більш ширшим рівнем видимості інформації, визначеним адміністратором попередньо. В результаті можна виділити наступні функції, що будуть доступні для ролі вчителя в системі:

- Можливість вносити зміни в перелік навчальних здобутків студентів, які виключно призначені для конкретного вчителя;
- Змінювати дані доступні в профілі;
- Обмежений доступ до редагування навчальних планів, призначених даному користувачеві;
- Змінювати вміст в розкладі відвідуваності студентів які призначені для конкретного викладача;
- Обмінюватися повідомленням в системі;
- Виконувати авторизацію та реєстрацію користувача;

Адміністратор це основна функціональна ролі, що виконує більшість дій пов'язаних на структуризацію доступності даних в системі для студентів та викладачів, визначає перелік осіб, що можуть користуватися системою після реєстрації. Надає можливість створювати нові навчальні плани, предмети. Дозволяє переглядати всю інформацію але омежує рівень доступу до змін в графіку відвідуваності, та журналі оцінок. Може проводити внутрішньосистемне листування. Для нього передбачені наступні функції Функцій що може виконувати адміністратор:

- Доступ до можливості модифікувати реєстр користувачів в системі;
- Дозвіл на маніпуляції даних про групи в програмі;
- Змінювати інформацію про предмети;

- Генерувати та модифікувати навчальні плани;
- Призначати доступність до інформації в системі для кожного окремого користувача;
- Доступ до перегляду всіх даних в системі та можливість пошуку в базі програми;
- Обмінюватися повідомленнями між користувачами;

Система-сервер не є представленням роботи користувача, але визначає відповідальність яку покладено на програму. Вона визначає способи надання відповідей на запити що отримуються від інших клієнтів, запити формуються автоматично з урахуванням поточних потреб від системи користувача. Вона повинна:

- Надавати дані з бази для поточних потреб користувача
- Зберігати дані користувача, пов'язані з ним інформацію про навчальний процес;
- Реалізовувати оновлення системи після внесення змін;

В результаті визначення варіантів використання стає можливим створення діаграми для візуального представлення функціональних завдань які необхідно реалізувати для коректної роботи програми. В результаті можна організувати весь функціонал у вигляді сутностей, що дозволяє візуалізувати всі взаємозв'язки між ролями які передбачаються в системі та командами які передбачено створити в програмі.

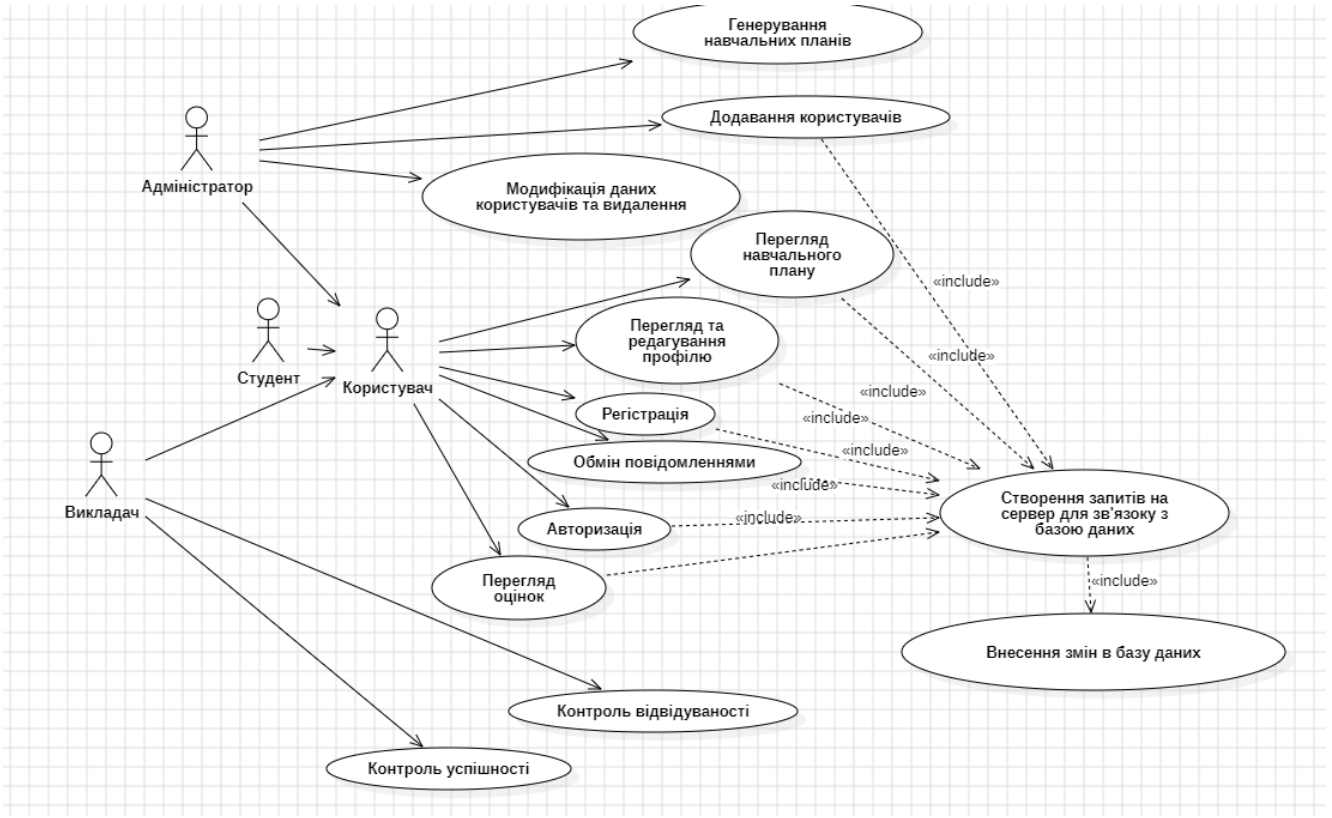


Рисунок 3.2.1 – Структура еволюційної моделі розробки програмного забезпечення

3.3 Організація класів

Відповідно до об'єктно-орієнтованого методу програмування, система передбачає реалізацію своїх функцій з використанням класів, які визначатимуть програмну реалізацію визначених вимог та відповідних варіантів реалізації. Крім того враховуючи те що система передбачає обмін інформації в формі клієнт-сервер то доцільно також створити підпрограму яка буде приймати, та обробляти, відповідати на виклики з пристроїв клієнтів. Таким чином доцільно зробити під субпрограму яка виконувати фонові функції для плавної роботи системи між девайсами в одній мережі. Тож для створення і коректної роботи програми передбачається наявність двох програм а саме клієнтської програми і фонової системи відстеження запитів, вона буде створюватися із використанням можливостей windows служб.

Основна програма StudentSheduler:

- StartWindow створює програмний вигляд інтерфейсу вікна входу в систему;
- UserRegister визначає інтерфейс деталей для реєстрації користувача в системі;
- TableView формує дизайн таблиць даних для вікна програми та інтерфейсу в системі;
- ObjectInterpretator визначає внутрішньосистемні програмні реалізації, які покликані для спрощення вибірки цільової інформації в програмному середовищі, без надмірного навантаження пам'яті пристрою;
- ServerExplover генерує підключення між пристроями та сервером для обміну даних для функціонування програми в локальній мережі;
- DataFormating визначає набір інструментів які містять в собі різні комбінації методів для ініціалізації взаємодії з запитами виклику інформації чи її модифікації;
- PreLoadProgram генерує вікно завантаження перевірки завантаження системи;
- ChatView створює інтерфейсні конструкції для роботи користувача з поштою в середовищі програми;
- Chats реалізує функціональну логіку команд для роботи функції чату в програмній системі;
- AlertWindow визначає вигляд та поведінку вікон сповіщень які виникатимуть для повідомлення користувача щодо статусу виконання його запитів;
- ValueChecker визначає набір інструментів, для перевірки коректності введених даних користувачем, та запобіганню їх збереження;
- AdminWindow реалізує вікно функціональних інструментів для роботи адміністратора з даними програмної системи;

- AdminFunctions визначає команди для коректного виконання роботи для визначених полів вікні інструментів адміністратора;
- CollorMixer реалізує механізми для генерації нових змін в середовищі програми;
- MainWorkWindow визначає функціонал та поведінку на дії для головного робочого вікна;
- DataManager містить реалзації для генерації вибірок з бази даних
- StudyShedule визначає поведінку вікна системи для генерації та взаємодії з навчальним планом для груп студентів;
- MarksBase визнчає інтерфейс та способи взаємодії користувачів з інформаційною базою оцінок;
- UserInfo реалізує взаємодію користувача з інформацією про себе яка була внесена в систему дозволяє створювати вікно профілю користувача;
- UserPhoto створює вікно для маніпуляції з фотографіями в системі;
- DatadaseDetail визначає способи підключення системи до бази даних на сервері системи;
- InstallerEdit містить додаткові команди розгортання які викликаються в процесі чи після інсталяції клієнтської програми на пристрої;

Програма серверу на основі служби Windows:

- ServerService реалізує процес запуску програми згідно з визначеними форматами служб Windows в якості програми сервера в форматі;
- TCPConectionListener генерує підключення, визначає та виконує дії які виконуватимуться в залежності від переданого через підключення запиту;
- TCPServerController виконує менеджмент доступних підключень в доступній локальній мережі за визначеними портами з'єднань;
- ServerInstaller містить інструменти для встановлення сервісу-служби

які використовують при встановленні основної програми;

Ці класи визначають як, програмні логіку, так і інтерфейсні реалізації для користувачів як WindowsForms об'єкти та системні установки як класи розгортання системи, для клієнтської програми та служби Windows. Відповідно вони реалізують як і елементи форм що застосовуються в системі для роботи інтерактивних дій із системою так і алгоритми автоматичної роботи для роботи сервера та обробки команд від користувача. Усі компоненти підтримуються за необхідності бібліотеками алгоритмів для відтворення рішень що підтримують оптимізацію та структурованість виконання команд програмного застосунку.

3.4 Метамодель системи та опис системи класів

В результаті створення програми виникає необхідність поділу функціональних систем на компоненти які мають різні направленості та кінцевий результат своєї діяльності. Через це часто кожна програма по своїй структурі набуватиме певного ієрархічного вигляду який дозволить краще пояснити залежності між програмними компонентами системи, визначити вплив одних на інші та передбачити кінцевий результат такої взаємодії. Тому доцільно оформити модель розробки які розподілятимуть діаграму класів на різні складові що в поєднанні виражатимуть метамодель програмної системи.

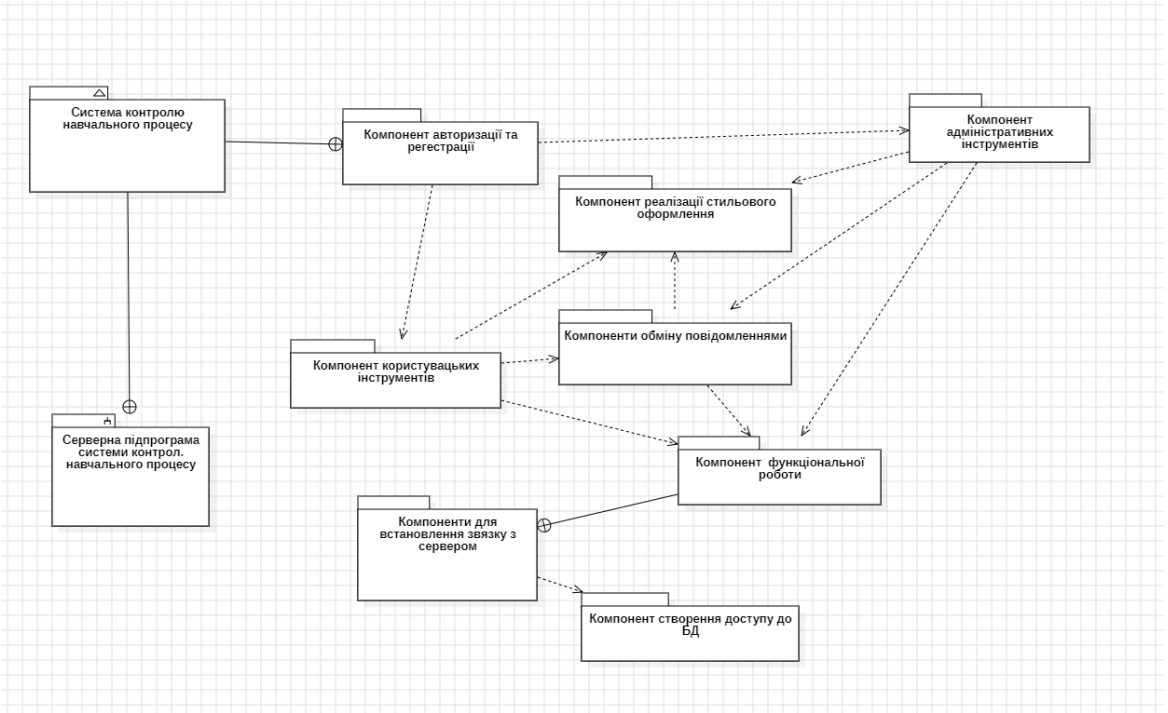


Рисунок 3.4.1 – Діаграма мета моделі системи

Ці всі компоненти визначають певні функції в системі що призводять до виконання вимог, що були визначені перед створення програми. В результаті чого створюється продукт який містить інструменти, що можуть виконати поставлені до системи цілі не використовуючи багато системних ресурсів для свого функціонування.

Компонент авторизації та реєстрації реалізує сценарії для авторизації та реєстрації користувачів в системі та визначення цільової ролі відповідно до ступеня доступності користувача. Компонент визначає групу класів, що реалізують інтерфейс для проведення дій, що спрямовані на визначення особи користувача та його можливостей в системі. Класи компонента:

StartWindow реалізує інтерфейс для виконання входу в систему, або ініціалізації реєстрації в системі. Наслідує системний клас створення інтерфейсів Form, має зв’язок з класом UserRegister

UserRegister визначає графічну реалізацію вікна для проведення реєстрації в системі. Наслідує клас графічного вікна Form.

PreLoadProgram визначає вікно загрузки для програми. Наслідує клас графічного вікна Form.

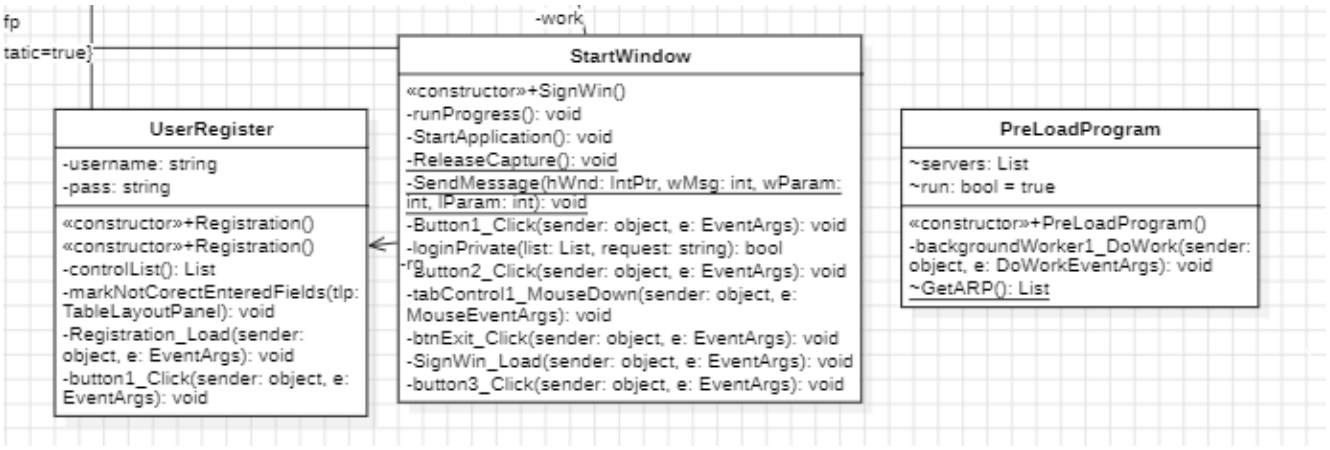


Рисунок 3.4.2 – Діаграма класів що відображає компонент авторизації реєстрації

Компонент функціональної роботи містить програмні інструменти та методи для контролю за даними що вводяться користувачем, проводять та ініціалізація їх обробку для подальшої роботи з ними в системі. В компоненті реалізований патерн посередник для оптимізації зв’язків в системі між елементами. Класи компонента:

DataFormating – містить реалізації функцій спрямованих на зміну даних системи та обробки даних системі, визначає способи сповіщення що попереджає користувача про процеси в системі. Наслідує DatadaseDetail, має зв'язок з класами AlertWindow і ValueChecker.

AlertWindow виконує генерацію вікна для повідомлень про результати виконання дій та стан роботи системи. Наслідує системний клас створення графічних інтерфейсів Form.

ValueChecker визначає інструменти для виконання перевірки введеної інформації від користувача.

ObjectInterpretator містить набір методів для виконання системного обміну та вибору інформації в програмному середовищі.

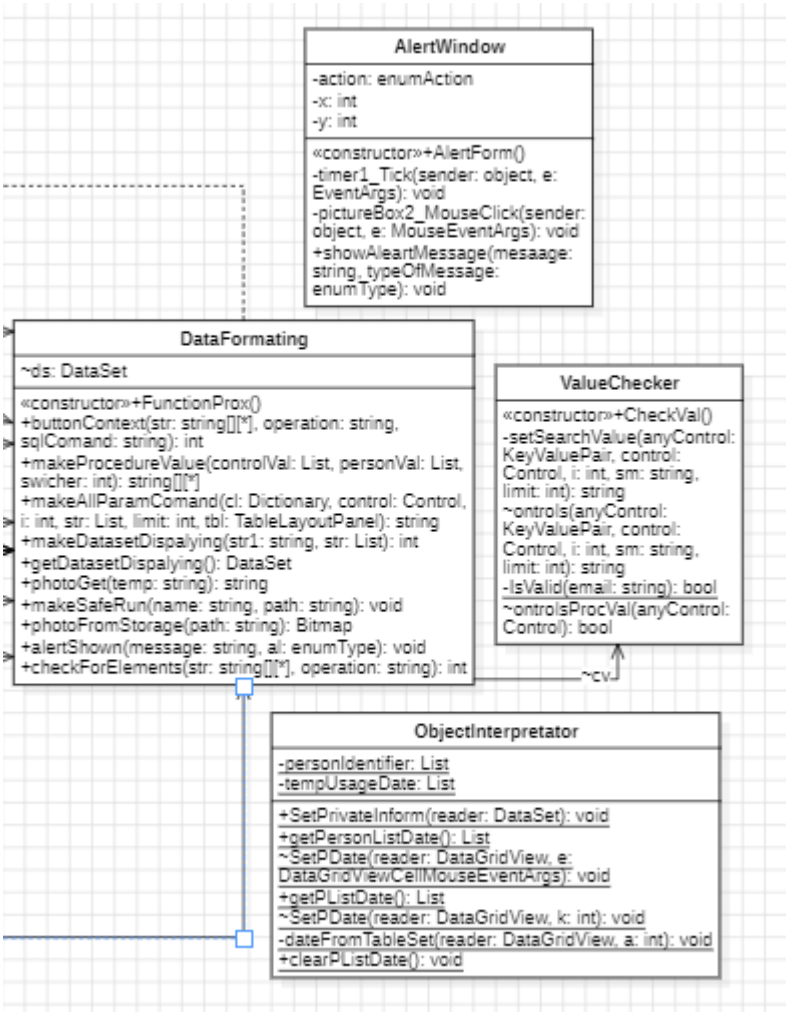


Рисунок 3.4.3 – Діаграма класів що відображає компонент функціональної роботи системи

Компонент адміністративних інструментів, містить реалізації для роботи інтерфейсу адміністративного вікна та методів обробки інформації отримані від адміністратора програми. Компонент реалізує графічне представлення для роботи з даними користувачів та відповідні функції їх обробки та підтримання. Вміст компонента:

- AdminWindow визначає інтерфейс для виконання адміністрування системи. Наслідує системний клас створення інтерфейсів Form, має зв’язок з класом DataFormating
- AdminFunctions реалізує інструменти контролю та обробки інформації що вводиться адміністратором. Має зв’язок з класом DataFormating.

Рисунок 3.4.4 – Діаграма класів що відображає компонент функціональної роботи системи

Компонент реалізації стильового оформлення системи визначає допустимі програмні представлення вигляду основного робочого поля системи, коричування та модифікації візуальних частин вікон програми. Компонент включає статичний клас що реалізує методи що впливатимуть на загальний вид системи та функції для табличних представлень програми. Класи компонента:

CollorMixer – статичний клас, що працює в усьому проекті та призначений внесення змін в вигляд робочого простору системи.

TableView статичний клас який визначає зміну вигляду табличних представлень системи

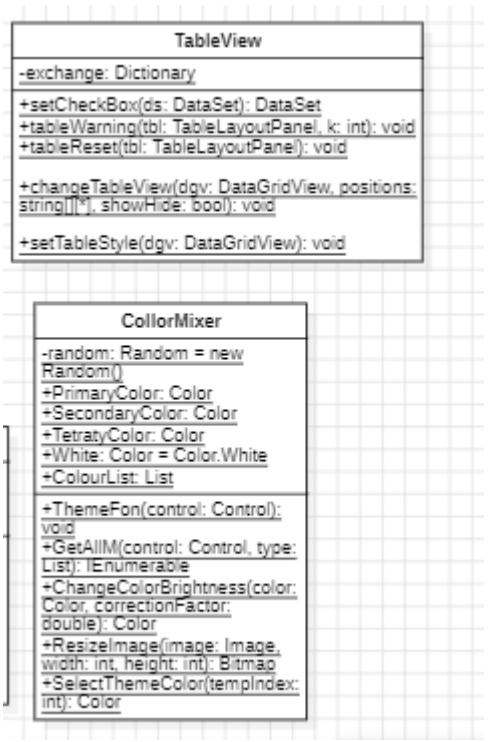


Рисунок 3.4.5 – Реалізація компонента зміни стилю

Компонент користувацьких інструментів визначає інтерфейси для використання функцій ведення обліку успішності та контролю навчального процесу, та інформаційну підтримку для них. Компонент включає класи що реалізують паттерн декоратор для впорядкування зв’язків між інструментами та різними типами користувачів які передбачені в програмі. Класи компонента:

MainWindow створює інтерфейс та реалізує механізми патерну декоратора який представляє набір інструментів призначений для виконання програми згідно запланованих завдань. Наслідує системний клас Form має зв'язок з класами StudyShedule, UserInfo, MarkBase.

StudyShedule генерує інструменти роботи користувача пов'язані з відображенням механізму створення та роботи з даними про навчальні плани та відвідуваність студентів користувачів. Наслідує системний клас UserControl і має зв'язок з класами MainWindow , FunctionProxy та UserPhoto..

MarkBase визначає способи створення та відображення даних про успішність студентів користувачі та відповідну інтерфейсну підтримку для них. Наслідує системний клас UserControl і має зв'язок з класами MainWindow, DataFormating та UserPhoto..

UserInfo містить інструменти роботи з дани про користувачів та підтримує віконне представлення функціональної області для маніпуляції ними. Наслідує системний клас UserControl і має зв'язок з класами MainWindow , DataFormating та UserPhoto.

UserPhoto визначає представлення для відображення фотографій користувача в системній мережі. Наслідує системний клас UserControl і має зв'язок з класами UserInfo, MarkBase, StudyShedule.

DataManager визначає інформаційну надбудову для кожного класу компоненту користувацьких інструментів, для визначення типів даних які використовуватимуться в системі відповідно до визначеної ролі користувача. Має зв'язок з класами MarkBase, StudyShedule.

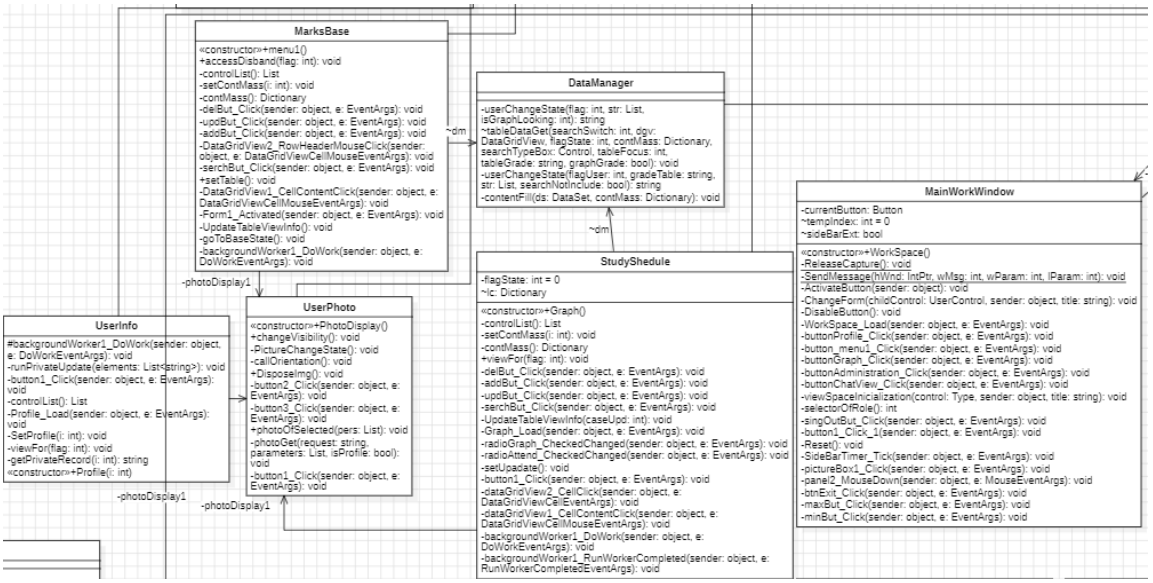


Рисунок 3.4.6 – Діаграма класів що відображає компонент контролю даних про персоналу

Компонент створення з'єднання до БД відповідає формування з'єднання до алгоритмів доступу бази даних, формування запитів на видобуто даних з сервера системи. Компонент включає клас що реалізує паттерн синглетон для зменшення нагромадження об’єктів доступу сховища даних, для оптимізації роботи програми та звільнити обчислювальні ресурси комп'ютера на виконання завдання. Клас компонента:

DatadaseDetail – клас представляє методи формування зв'язків з базою даних системи. Наслідується класом DataFormating.

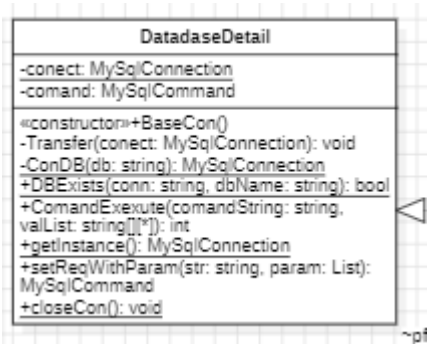


Рисунок 3.4.7 – Діаграма класів що відображає компонент створення доступу до БД

Компонент відображення створення з'єднання з сервером програми яка містить необхідну інформацію для роботи програми. Компонент включає наступний клас:

ServerExplover– клас представляє методи створення контакту з сервером програми. має зв'язок з класом DataFormating.

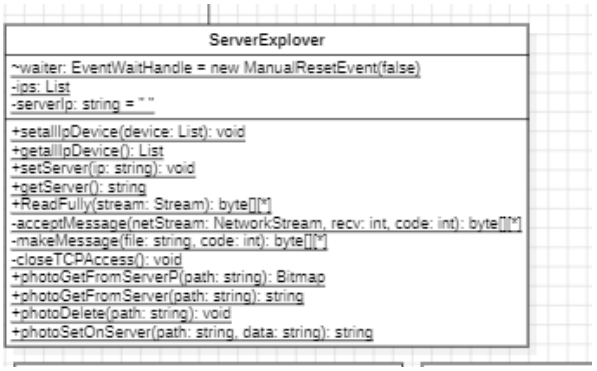


Рисунок 3.4.8 – Діаграма класів що відображає компонент створення доступу до БД

Компонент обміну повідомлень визначає функції для передачі інформації між користувачами визначеними в програмній системі та інтерфейс для їх реалізації. Компонент включає наступні класи:

Chats створює функціональні реалізації для визначеної системи обміну інформації між користувачами. Має зв'язок з класом DataFormating.

ChatView визначає інтерфейсне представлення для роботи системи листування та представляє елементи ініціалізації функцій для комфортної роботи з поштою програми. Має зв'язок з класом Chat.

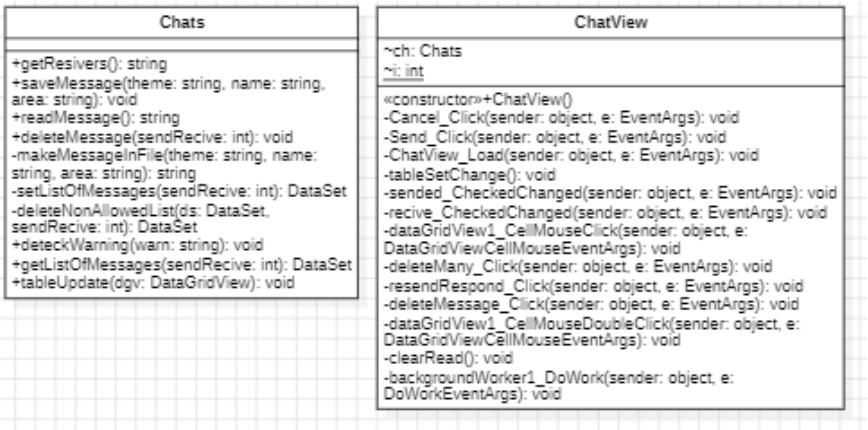


Рисунок 3.4.9 – Діаграма класів що відображає компонент створення доступу до БД

Обєднавши всі вищевказані компоненти в результаті виходить наступна діаграма класів:

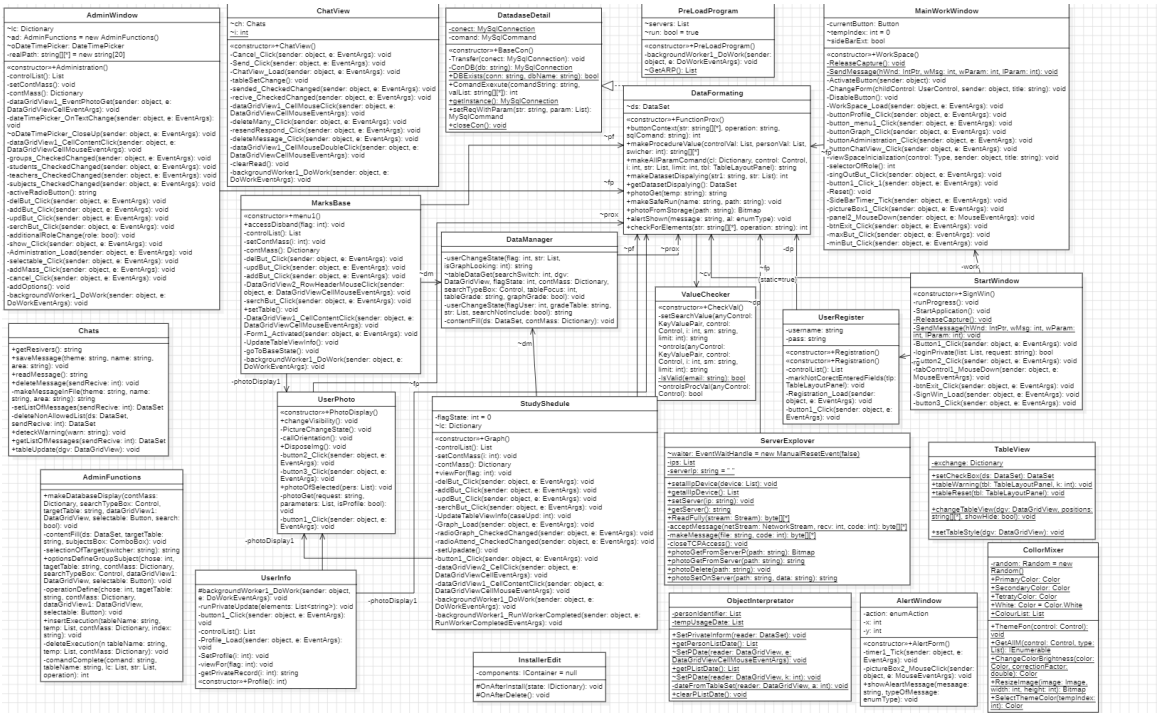


Рисунок 3.4.10 – Діаграма класів для клієнтської частини програми

Також в системі існує підпрограма яка виконує серверну частину роботи програми через використання інструментів Windows служб, які мають можливість використовувати фонові потоки для постійної підтримки роботи серверу без необхідності запуску з боку користувача програми. Цей пакет містить наступні класи:

ServerService визначає спосіб запуску на припинення роботи служби

Windows в фоновому режимі роботи програмної системи.Має зв'язок з класом TCPServerController

TCPConectionListener створю програмні визначення для роботи з командами які надаються серверу з клієнтської програми та надсилає відповіді після завершення роботи відповідно до визначених запитів.

TCPServerController визначає механізми розподілу відповідальності щодо встановлених з'єднань між клієнтами та серверами у визначеному порядку підключень.

ServerInstaller має значення для автоматичного встановлення служби Windows без необхідності залучення користувача на серверній частині.

Звідси, діаграма класів підпрограми виглядатиме так:

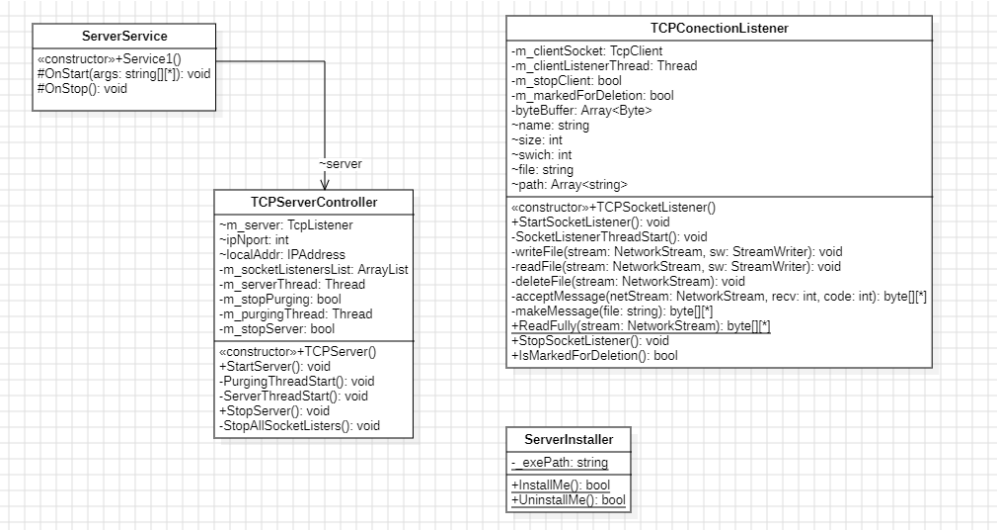


Рисунок 3.4.11 – Діаграма класів для частини сервера

Всі визначені звязку між компонентами залишаються дійсні і до отриманих представлень через діаграми класів. Завдяки формуванню класів в якому вигляді вдається отримати низьку зв'язність між елементами, це дозволить п подальшому розшири чи замінити компоненти та дозволить залишатися програмі мобільною щодо встановлених функціональних реалізацій. В результаті отримані діаграми класів, визначатимуть запланований функціонал системи відповідно до спроектованої метамоделі що будуть реалізувати зв’язків між компонентами та механізми використання їх елементів для виконання поставлених завдань які будуть ініціалізовані користувачем.

3.5 Проектування бази даних

Враховуючи необхідність використання бази даних для програми яка розробляється через потребу обміну та маніпуляції інформацію, доцільно створити структуру, що визначатиметься як сховище даних. Сама структура відображатиме окремі елементи навчального процесу, компоненти які впливають на кінцеві результати, та вирізняють їх з поміж інших в системі. Звідси вирішено створити структури для збереження інформації про користувачів системи, успішність студентів, навчальні плани та відвідуваність а також зберігати значення для обміну повідомленнями між користувачами в системі та даних для контролю входу в систему.

Проектування сховища даних потребує виконання наступних етапі для коректної роботи:

- Концептуальне проектування.
- Логічне проектування.
- Фізичне проектування.

База даних для як найточнішої роботи системи потребує включення в сферу знань опис реальних сутностей або виключення різних математичних формул та функцій з яких дотримуватимуться актуальні для коректної роботи даних на основі яких відбуватиметься функціонування програми.

В процесі логічного проектування створюються особливості інструментів програмування баз даних для визначення зв'язків між сутностями предметної області для визначення необхідних для роботи системи залежностей, які впливатимуть на кінцевий результат роботи програми. Для системи що розробляється вирішено створити структуру бази даних на основі методу проектування «сутність-зв'язок» або як його ще називають «ER-діаграм». В використання входить включення наступного набору елементів побудови логіки сховища інформації: сутність, атрибути, ключі, зв'язки між

елементами-сутностями, ступінь зв'язку, екземпляри класів сутностей, діаграми ER-типу.

Елементи-сутності дозволяє визначити елемент предметної області як набір компонентів-характеристик програми, яка дотримується певних логічних обмежень та подається у вигляді кінцевої інформації в базі даних.

В результаті виконання проектування бази даних розробник отримує життєздатну структуру об'єктів що є необхідними для роботи предметної області в вигляді таблиць, з визначеними між ними зв'язками для розширення даних про елементи системи які поясніть взаємодію між сутностями, визначають тип доступу інформації між ними. Створена інформаційно-логічна модель зображена на на рисунку 3.5.1

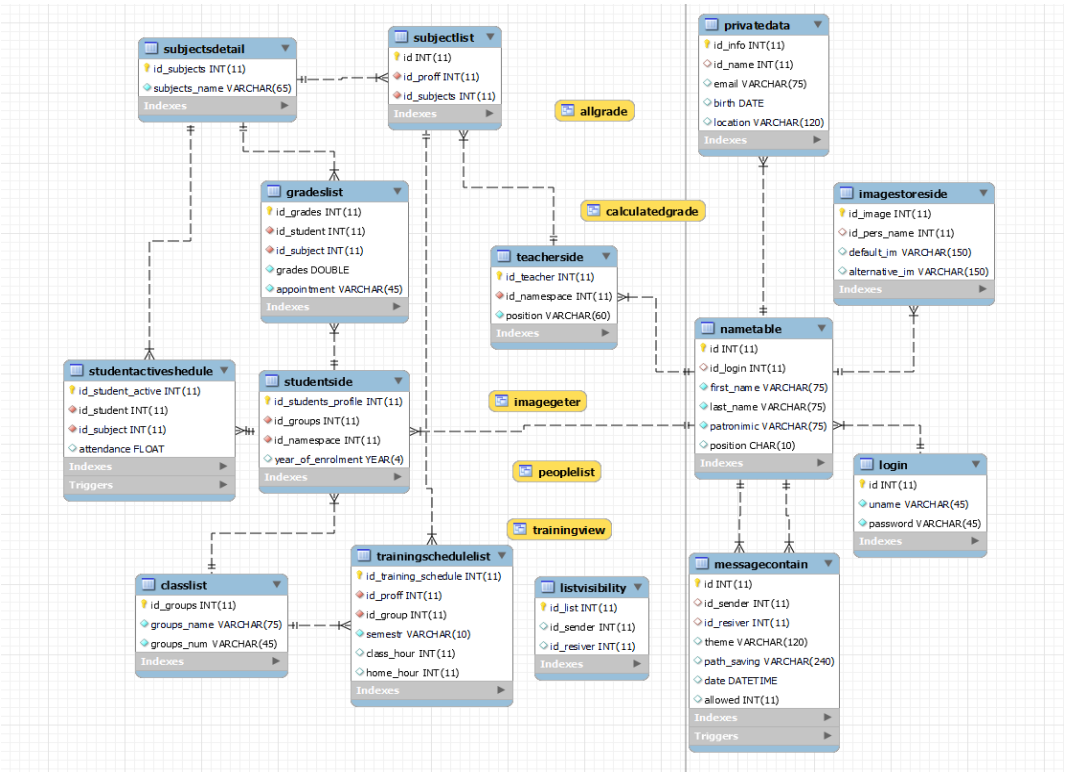


Рисунок 3.5.1 - ER-діаграма сутностей

Діаграма відображає взаємодії в предметній області в вигляді набору наступних таблиць, представлень та процедур:

- Таблиця «Login» відображає структуру збереження інформації про заходи користувачів, через який відбувається розподіл доступності даних системи;

- Таблиця «Namestable» дозволяє зберігати дані про ідентифікацію конкретного користувача через його ім’я визначене через адміністратора;
- Таблиця «PrivateData» розширює вміст інформації про користувачів системи, через контактні дані особи визначеної в системі;
- Таблиця «ImageStorageSide» визначає шлях зберігання посилань на конкретне збереження фотографій пов'язаних з користувачем з використанням ідентифікатора користувача визначеного в базі даних;
- Таблиця «TeacherSide» визначає ідентифікатори викладачів в програмному середовищі, та описує атрибути які їм призначені в системі;
- Таблиця «StudentSide» реалізує структуру зберігання деталей пов'язаних з студентами, та визначає атрибути їх ідентифікують;
- Таблиця «ClassList» створює визначення про навчальні групи в середовищі та їх основні деталі;
- Таблиця «SubjectsDetail» визначає структуру збереження інформації пов'язаної з ідентифікацією навчальних предметів;
- Таблиця «GradesList» визначає спосіб побудови збереження інформації для оцінок студентів-користувачів в мережі системи через визначення ідентифікатора користувача предмету та числового значення отриманого балу;
- Таблиця «StudentActiveShedule» створює механізм який дозволяє визначати і відстежувати рівень відвідуваності уроків студентом зберігаючи її за визначеним ідентифікатором користувача, спредметом та фактично процентним еквівалентом відвідуваності учня;

- Таблиця «SubjectList» дозволяє визначати складені структури для визначення рівня доступності користувача викладача до інформації системи;
- Таблиця «TrainingScheduleList» Створює структуру для збереження та генерації даних про навчальний плани для конкретних груп визначених системою через ідентифікатори таблиць з відповідною інформацією;
- Таблиця «MessageContain» реалізує структуру для визначення доступності до повідомлень користувачів до повідомлень та посилань для їх відтворення в мережі системи;
- Таблиця «ListVisibility» генерує визначення маніпуляції видимості повідомлень для конкретних користувачів, і вирішує термін існування повідомлення в системі від загальної доступності до листа між користувачами;
- Вид «CalculatedGrade» який включає сумарне значення оцінки за предмет для всіх користувачів;
- Вид «AllGrade» відтворює таблицю яка видає всі збережені в системі оцінки студентів по предметах;
- Вид «TrainingView» визначає вигляд навчального плану з відповідним включеннями таблиць для генерації навчальних планів;
- Вид «PeopleList» створює вид таблиці для даних про всіх студентів, зі включенням особистих даних користувачів;
- Вид «ImageGetter» створює вибірку інформації що пов'язана з визначенням шляхів до відтворення фото матеріалів користувачів.

3.6 Тестування програмного продукту

Тестування програми відбуватиметься з використання двох пристроїв які будуть перевіряти коректність передачі даних в спільній локальній мережі між комп'ютерами. Тестується система з допомогою системи Windows 10, з об’ємом оперативної пам'яті 8ГБ на обох пристроях.

Перед початком тестування необхідно запустити службу Windows для коректної роботи сервера системи. Запуск відбувається на пристрої який відіграватиме роль клієнта-сервера

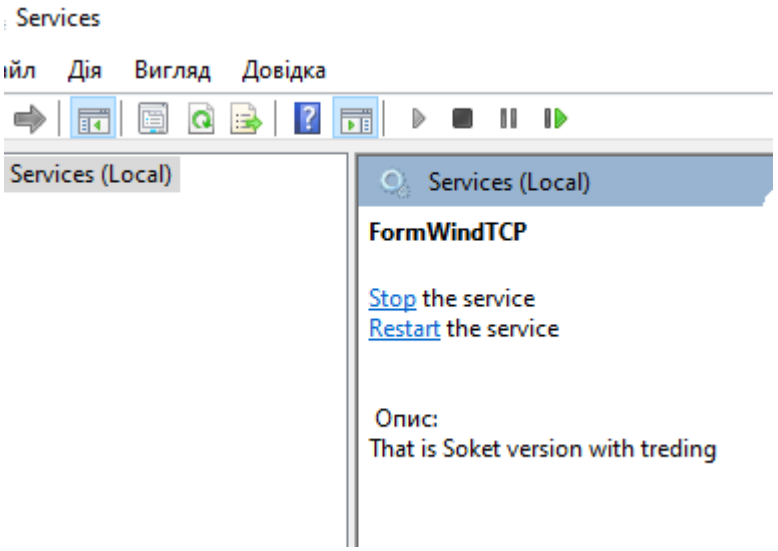


Рисунок 3.6.1 – Запуск служби-сервера

При запуску користувач отримує наступне вікно яке підготовлює програму перед початком роботи для користувача:

Рисунок 3.6.2 – Вікно завантаження роботи системи

Після успішного завантаження користувач отримує вікно авторизації де може входити реєструватися і встановлювати обліковий запис адміністратора.

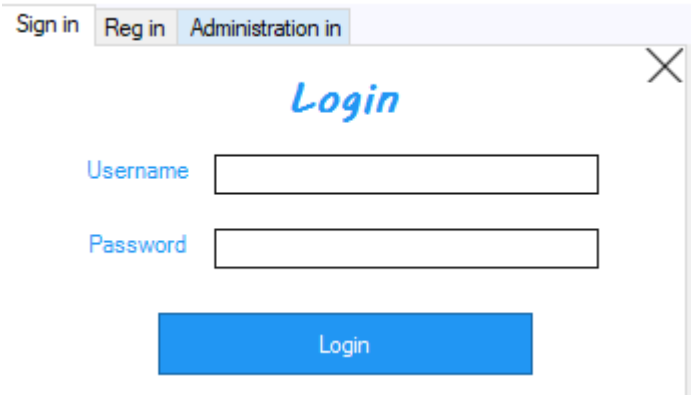


Рисунок 3.6.3 – Вікно авторизації користувача при вході в систему

При помилці введених даних до відповідно допустимих системою отримуємо наступний вигляд системи:

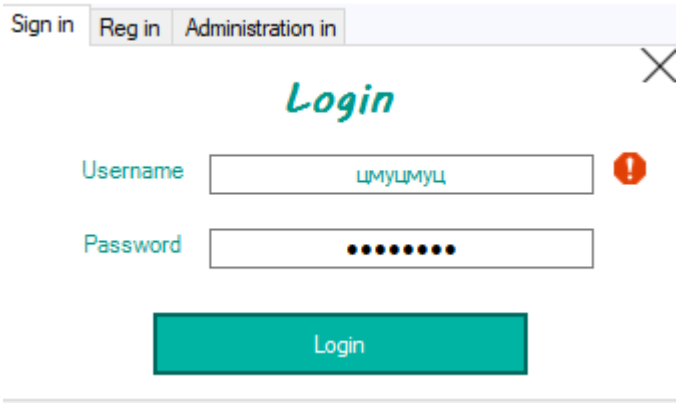


Рисунок 3.6.4 – Вікно авторизації при помилці входу

А також створюється спливаюче вікно що повідомляє про помилку користувачу.

Рисунок 3.6.5 – Спливаюче повідомлення про попередження системи

Для авторизації викликається друга вкладка яку еадає вікна авторизації. Вона виглядає наступним чином:

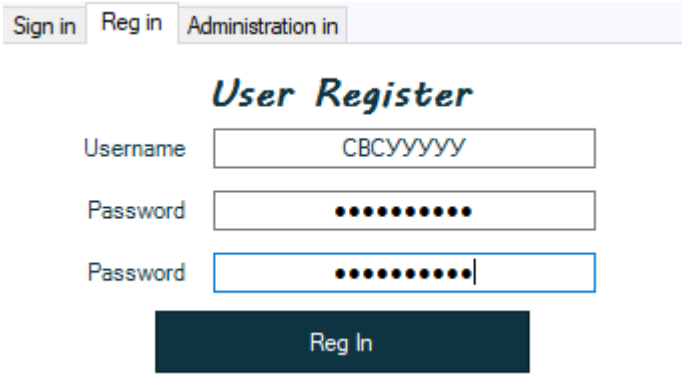


Рисунок 3.6.6 – Вкладка реєстрації

Після введених даних виконується перевірка на відповідність певним нормам і якщо вони пройдені відображається наступне вікно програми:

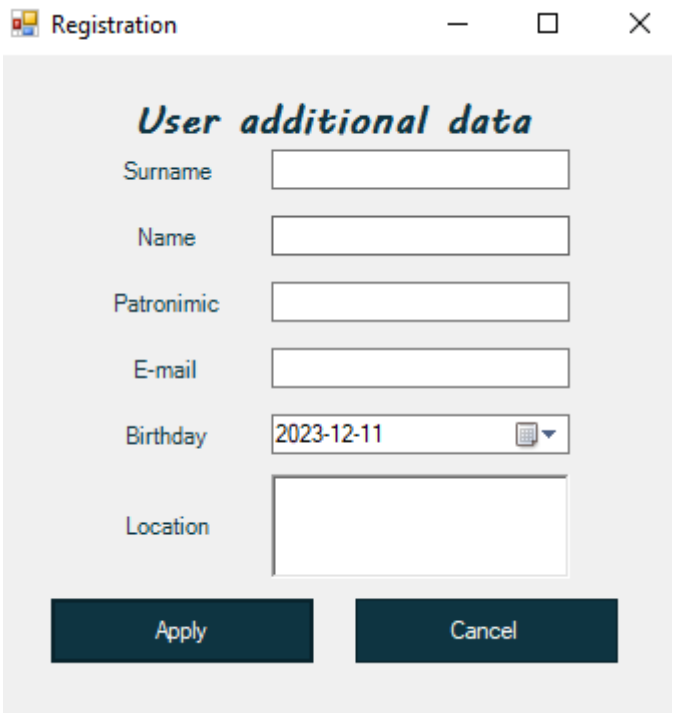


Рисунок 3.6.7 – Вікно реєстрації користувача при вході в систему

Тут вводяться певні дані для точно визначення користувача в системі, щоб уникнути недостовірного присвоєння даних та запобігти плутанини інформації всередині системи.Для успішної реєстрації необхідно ввести всі доступні для вводу поля, і якщо в системі наявні такі самі дані (введені через адміністратора) то реєстрації пройде успішно в іншому випадку випаде наступне вікно:

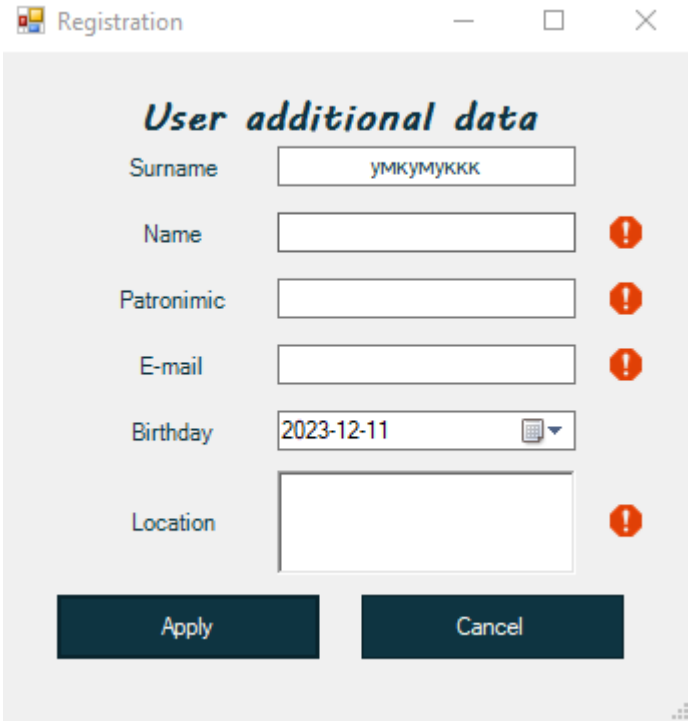


Рисунок 3.6.8 – Вікно реєстрації користувача при неправильному заповненні полів

Якщо не має зареєстрованого користувача в системі і існують такі записи в системі тоді реєстрація буде успішною і вхід виконуватиметься під таким профілем.

Остання вкладка визначена в вфкнф входу в систему це розділ реєстрації адміністратора

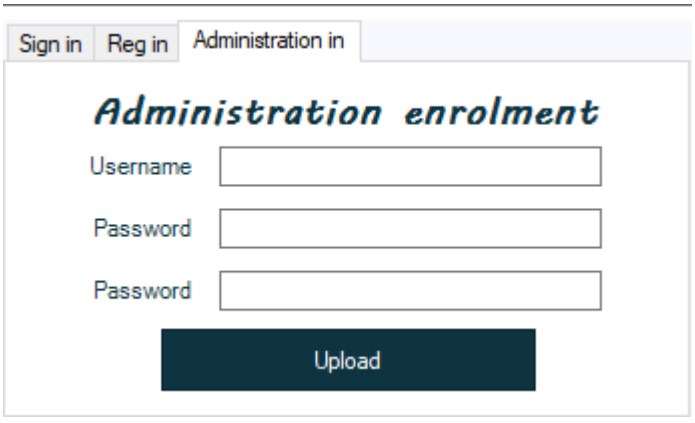


Рисунок 3.6.9 – Вікно створення облікового запису адміністратора.

Працює як поля авторизації проте може виконуватися тільки один раз при першому вході в систему, тому що передбачається тільки один профіль адміністратора в створеній системі.

Після входу користувача переходить в наступне вікно:

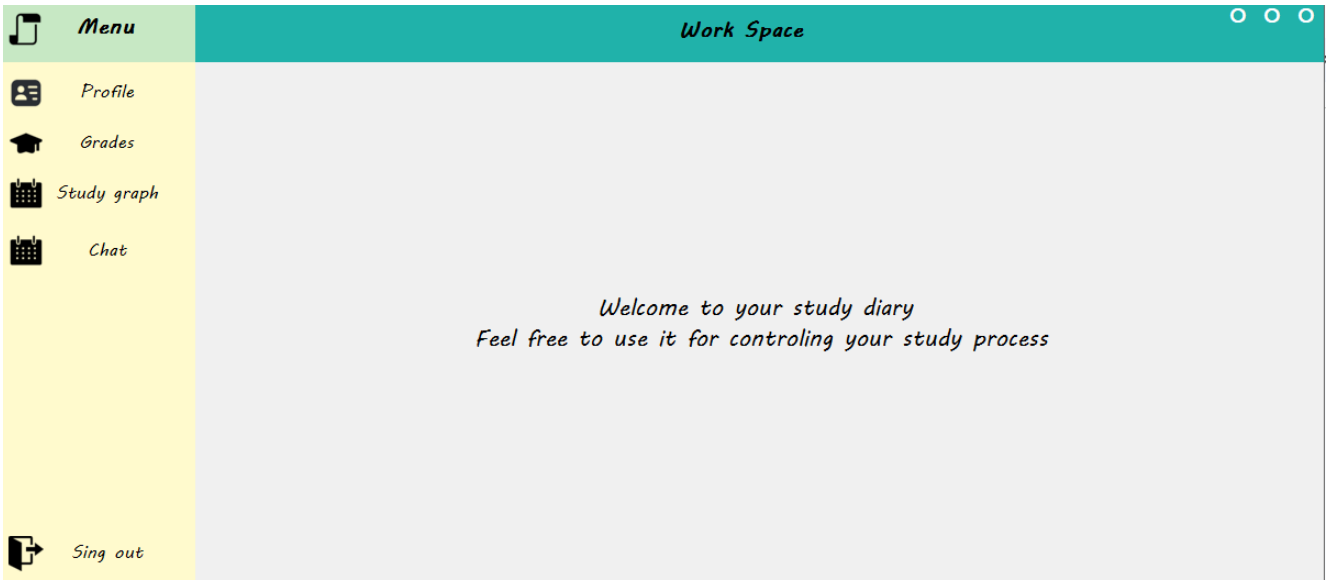


Рисунок 3.6.10 – Вікно після авторизації користувача

В залежності від ролі користувача інструменти адміністратора приховані або відображені на панелі меню.

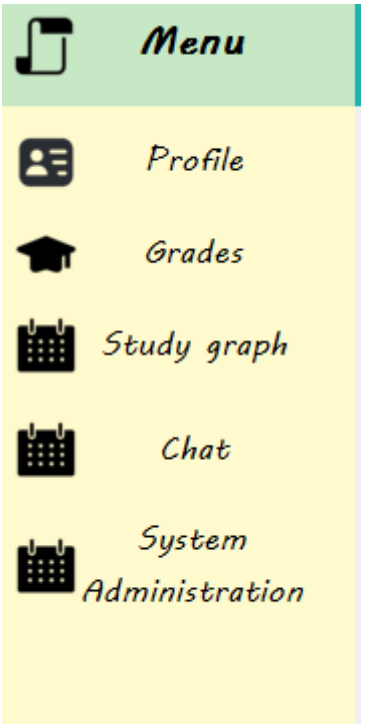


Рисунок 3.6.11 – Вигляд меню зі сторони адміністратора

При виборі вкладки Profile користувачу надається доступ до свого профілю в системі і створюється наступний вигляд:

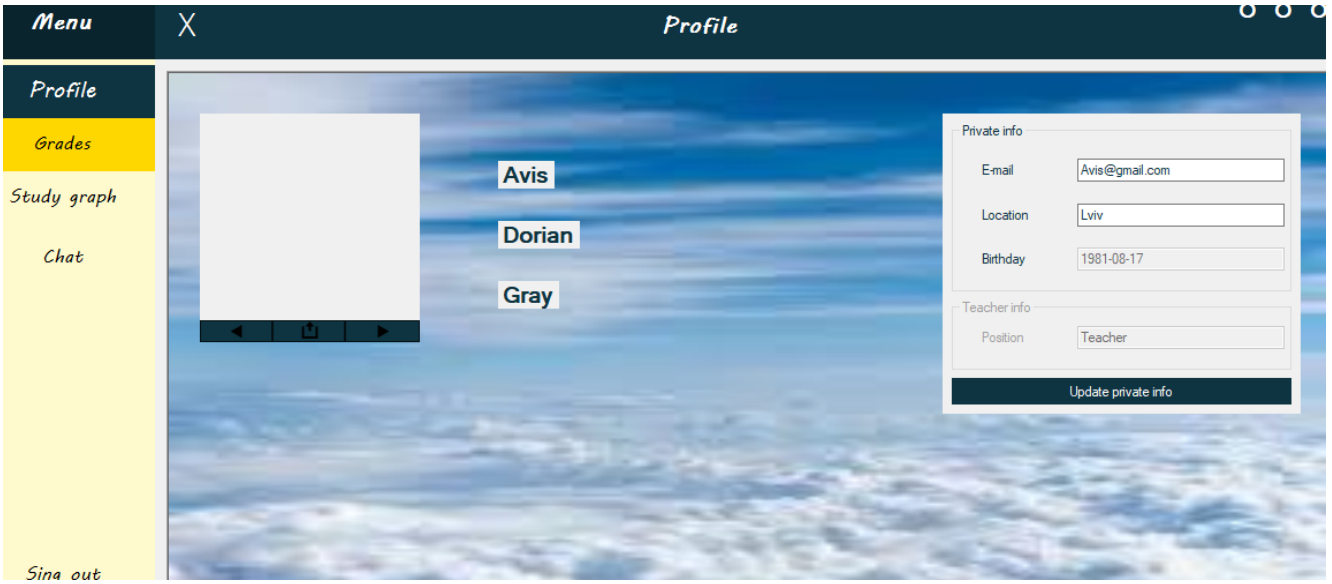


Рисунок 3.6.12 – Вікно профілю користувача

Для всіх акторів воно матиме однаковий вигляд

Подальші вигляди і можливості системи варіюються відповідно до ролей. Для учня як і для адміністратора вигляд вкладки оцінювання буде набувати наступного вигляду, оскільки їм не має можливості змінювати оцінки в системі:

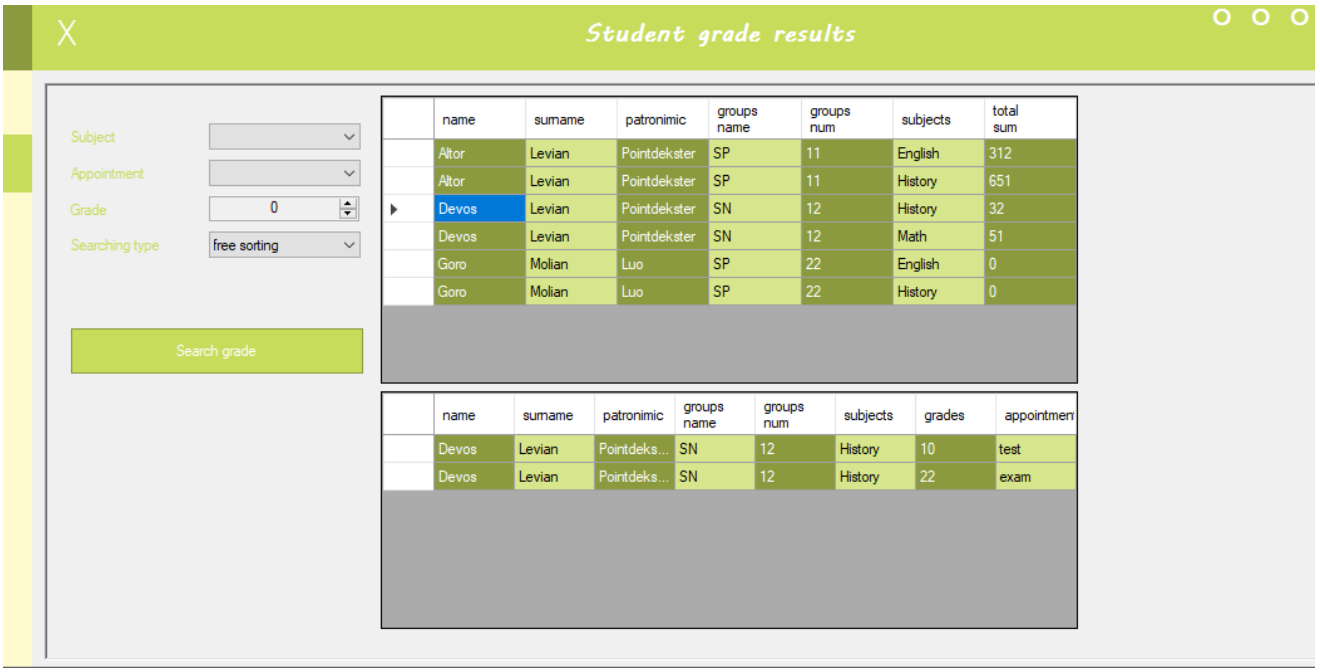


Рисунок 3.6.13 – Інструменти перегляду оцінок студента вигляд студента і адміністратора

Різниця полягає тільки в записах які відображаються. Учень може переглядати тільки власні записи, адміністратор усі.

Для викладача оцінювання вікно оцінювання буде більш детальним.

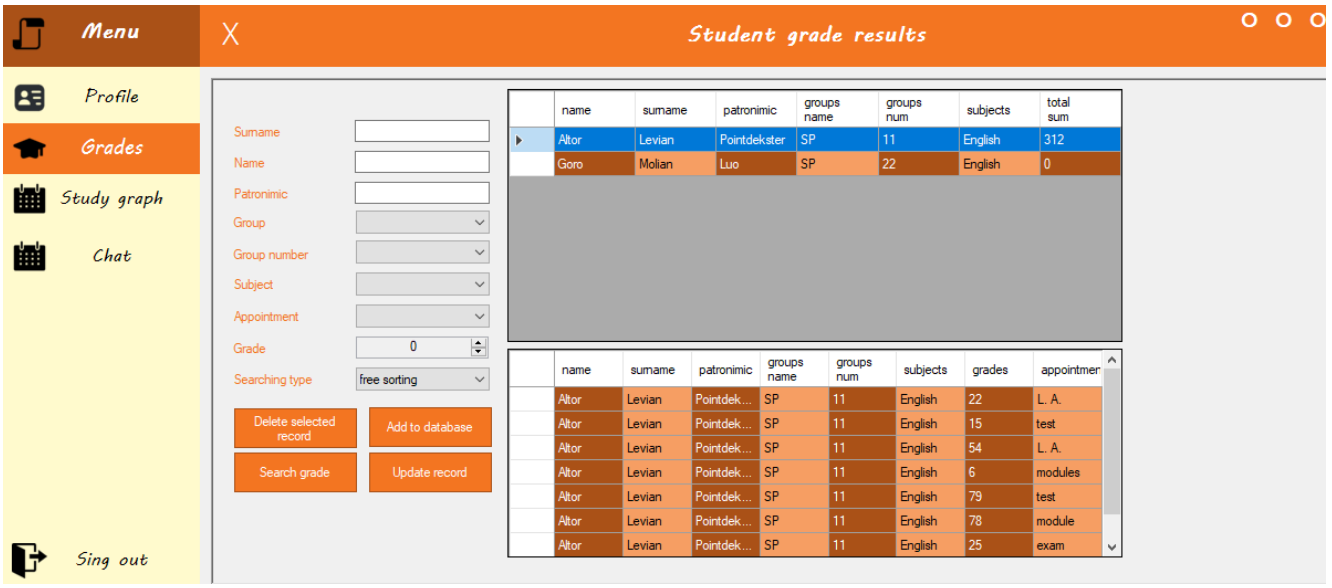


Рисунок 3.6.14 – Інструменти оцінювання для викладача

Вкладка навчального процесу для адміністратора матиме наступний вигляд в системі:

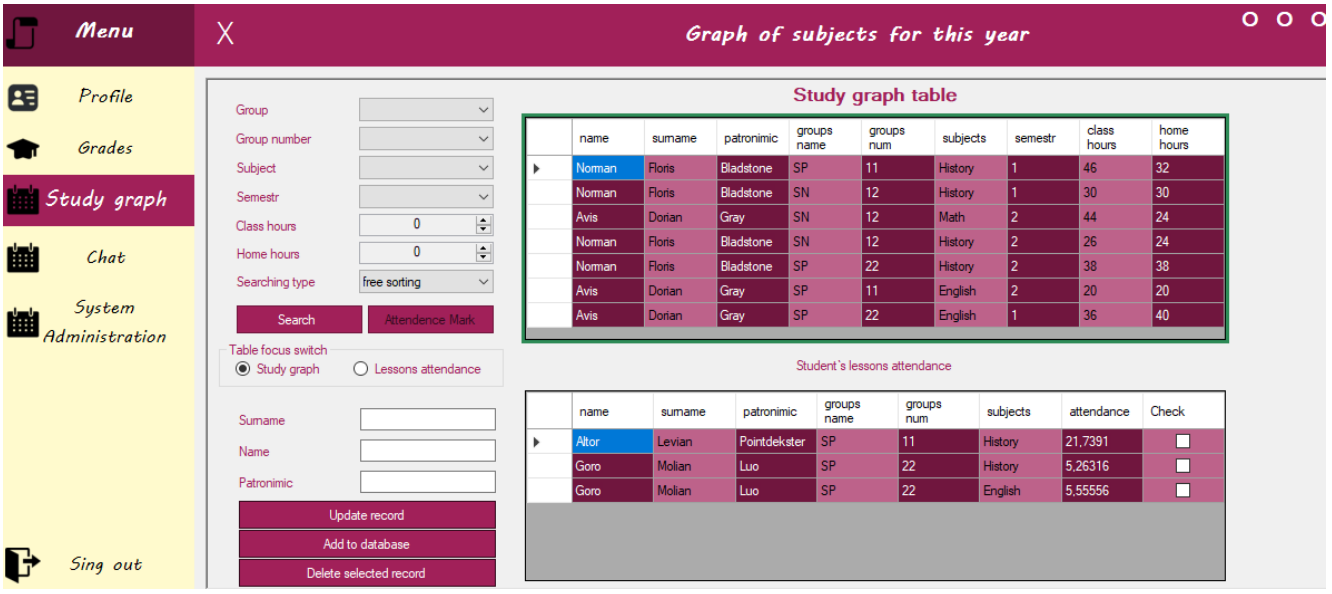


Рисунок 3.6.15 – Інструменти для формування навчального плану для адміністратора

Адміністратор може виконувати будь-які дії щодо навчального плану і переглядати всі дані про них. Проте не має можливості впливати на дані відвідуваності студентів.

Для учня навчальний графік виглядатиме так:

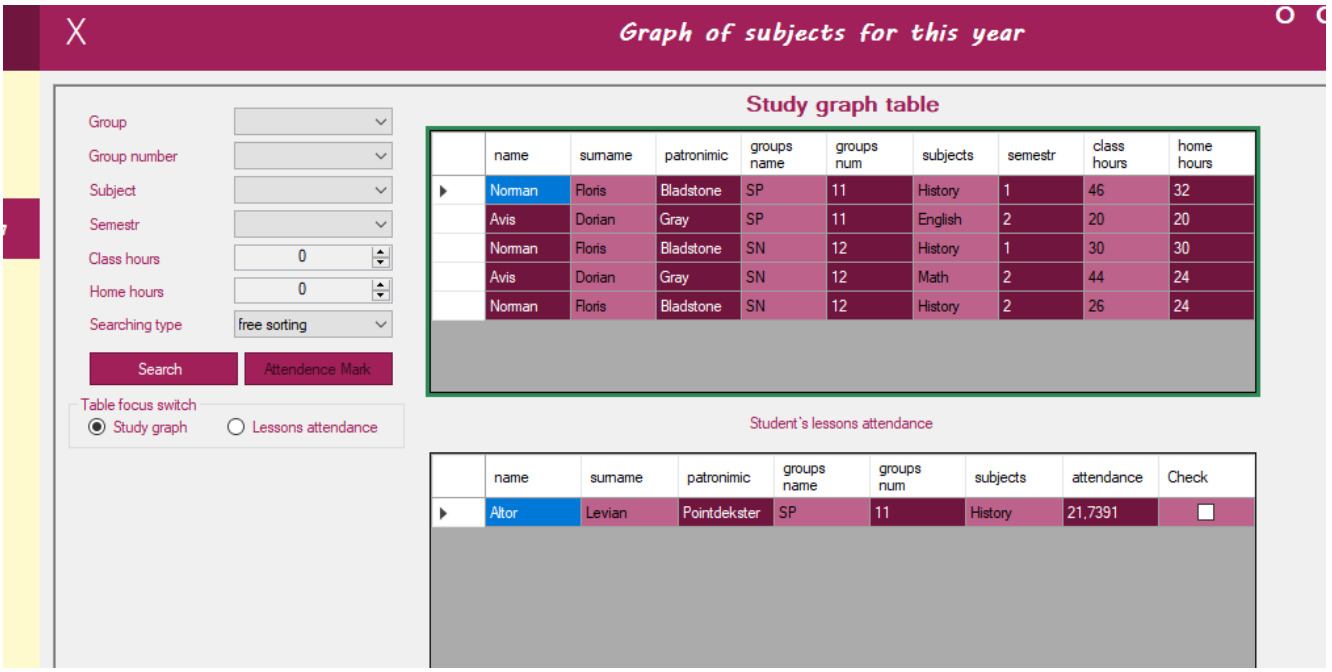


Рисунок 3.6.16 – Інструменти для формування навчального плану вигляд для учня

Тут відображаються дані тільки про викладачів і предмети щодо групи і дозволений лише пошук.

Для вчителя буде доступним виставлення відвідуваності по предметах які призначенні йому для ведення. Це вікно набуватиме наступного вигляду:

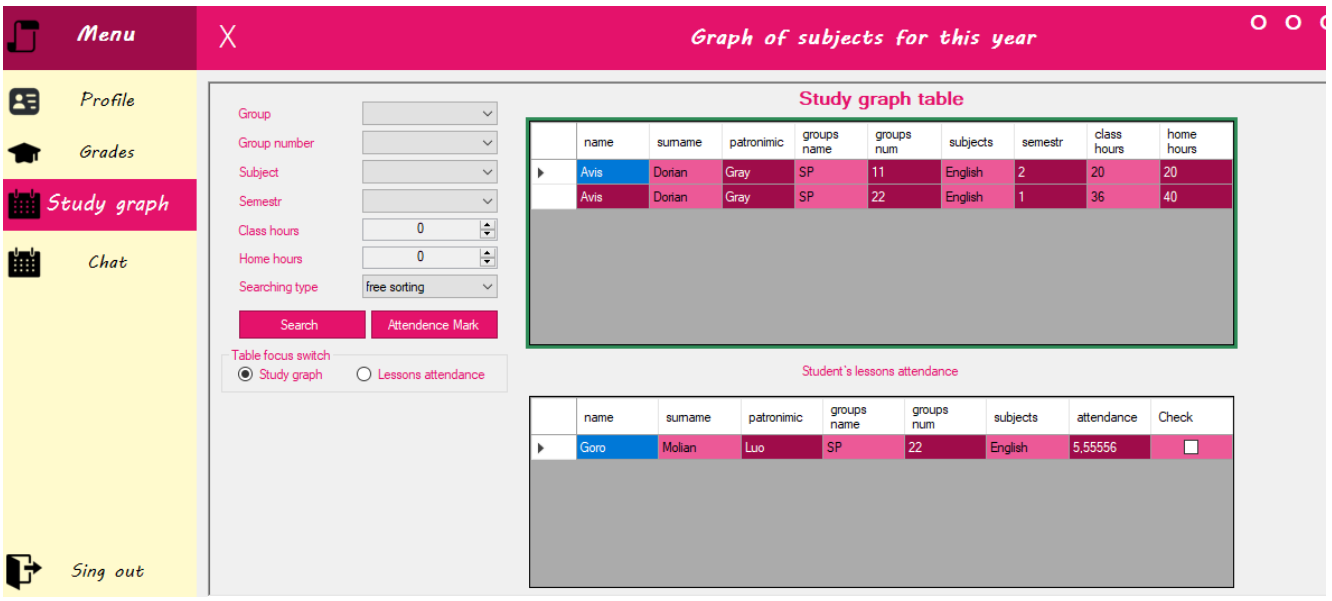


Рисунок 3.6.17 – Інструменти для формування навчального плану вигляд для вчителя

Вкладка адміністраторських інструментів виглядає наступним чином:

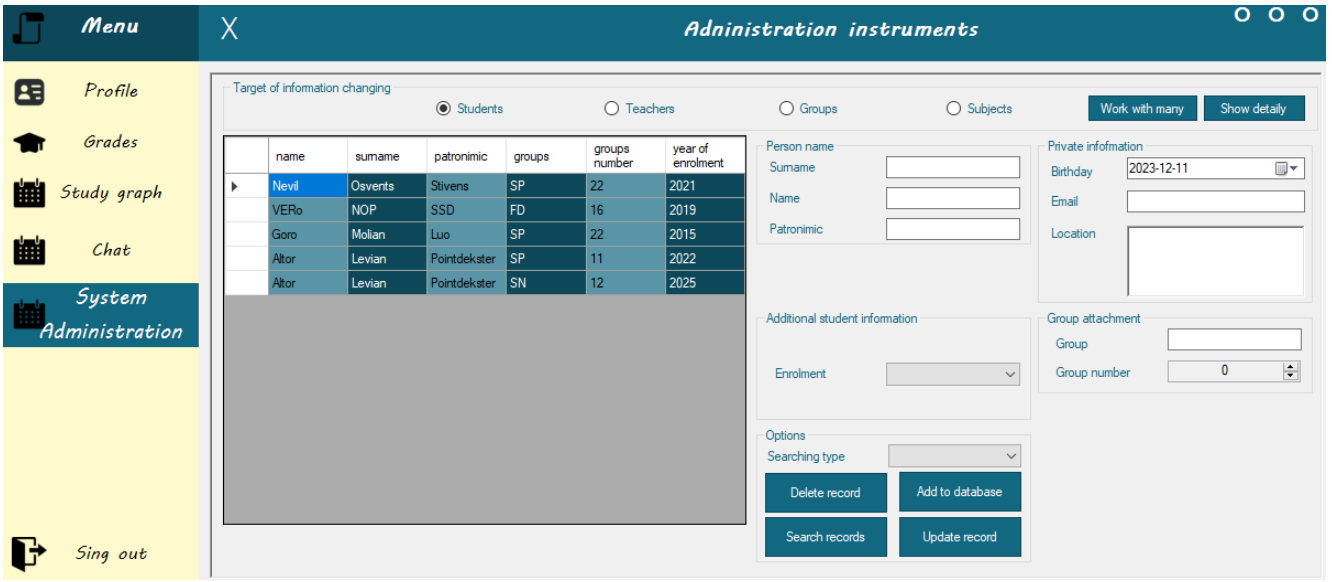


Рисунок 3.6.18 – Інструменти адміністратора системи

З їх використанням адміністратор може міняти дані користувачів, груп, предметів, переглядати та видаляти їх. ТАкож є можливість роботи збагаться елементами одночасно. Крім того є форма для внесення багатьох елементів в систему за один раз.

Також є вкладка чату. Вона відображається для всіх користувачів однаково.

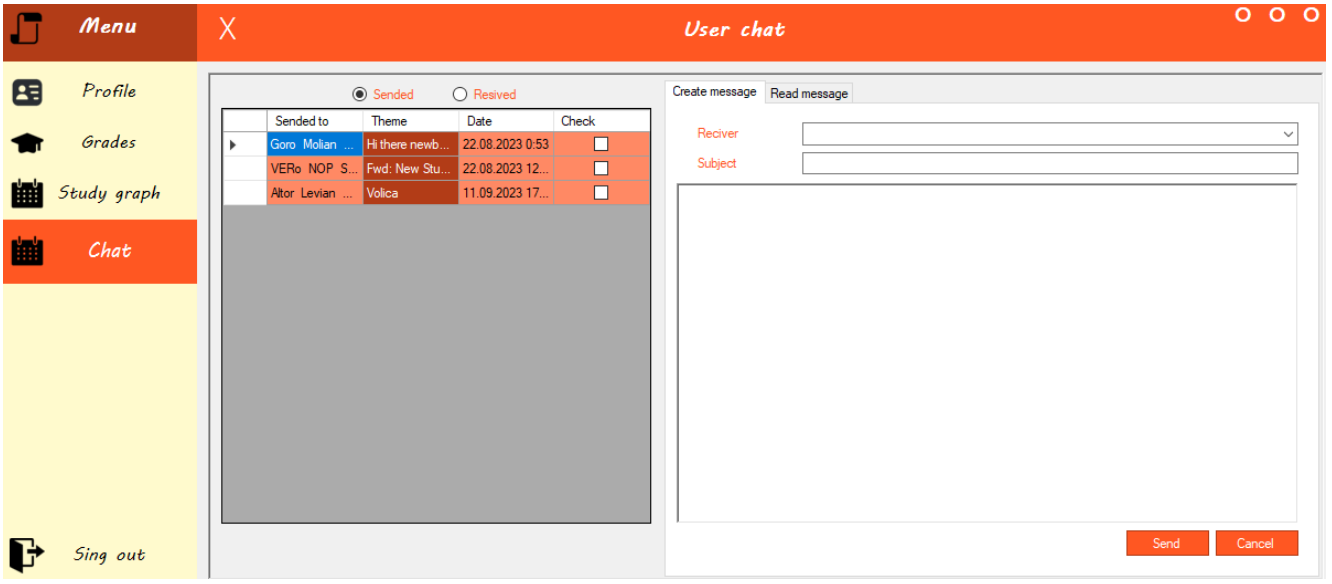


Рисунок 3.6.19 – Інструменти чату системи

Є поділ на відправлені листи і отримані. Відділи для написання листа. та його читання.

Якщо проводити будь-які дії спрямовані на внесення змін, видалення інформації з бази то можуть відображатися вікна сповіщень які повідомлятимуть про

успішність виконання дій. Успішне завершення дій виглядатиме наступним повідомленням:

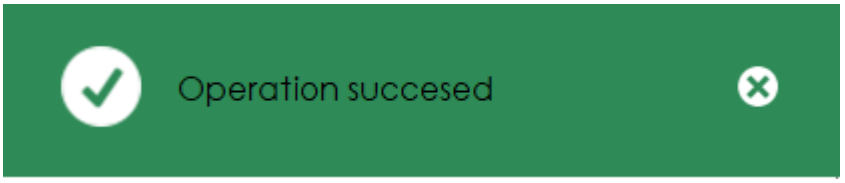


Рисунок 3.6.20 – Спливаюче повідомлення про успішне завершення дії

Або при виникненні помилки виникає наступне сповіщення:

Рисунок 3.6.21 – Спливаюче повідомлення про помилку при виконанні дії

На формі після вибору будь якого розділу присунія кнопки “X” на панелі. Її нажимання призводить до повернення початкового вигляду вікна.

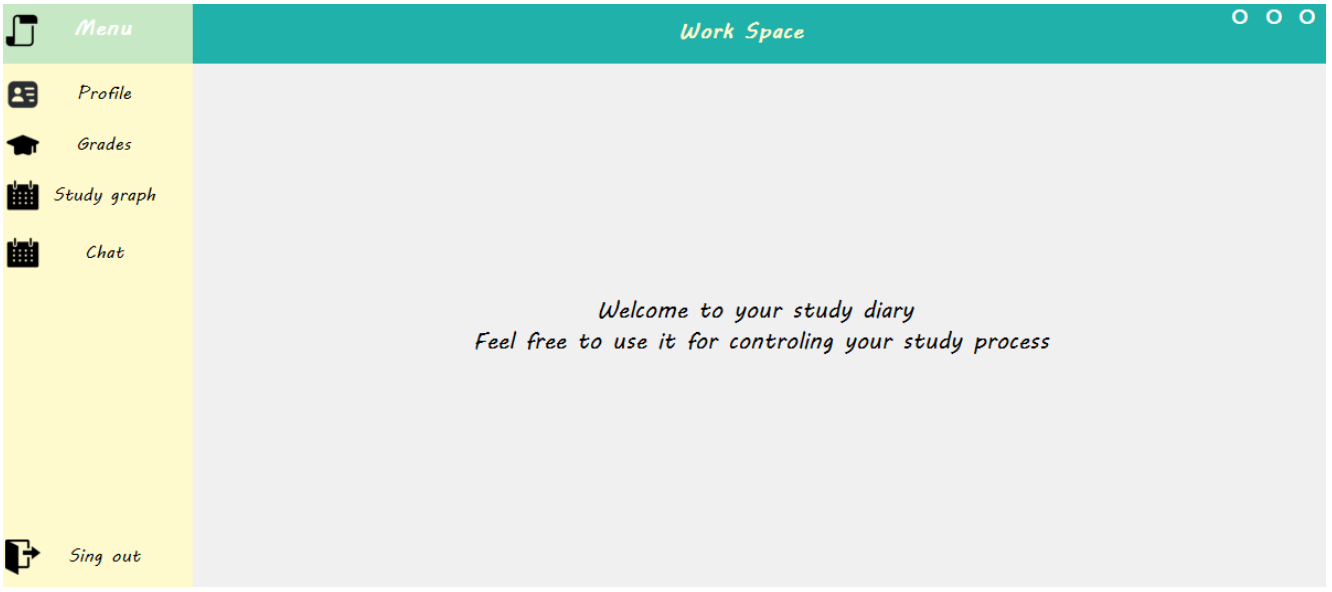


Рисунок 3.6.22 –Закриття активних вкладок програми

Кнопка “Sing out” призводить до ініціалізації закриття робочого вікна програми та повертається до вікна авторизації.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ

4.1 Охорона праці

Розглянемо стандартне робоче комп’ютеризоване місце для працівника, який проводить за комп’ютером більшу частину робочого часу за розробкою інформаційно-електронної системи для контролю знань та обміну інформації. Точніше, робоче місце середньої важкості IIб за ДСН 3.3.6.042-99 з ВДТ [13].

Роботодавець поінформував працівників під розписку про умови праці та наявність на наших робочих місцях небезпечних та шкідливих виробничих факторів, які виникають під час роботи з екранними пристроями та ще не усунуто, а також про можливі наслідки їх впливу на здоров’я працівників відповідно до вимог статті 5 Закону України „Про охорону праці”, тобто робоче місце відповідає ДСанПІН 3.3.2.007-98 [14].

Також він забезпечив навчання і перевірку знань працівників з питань охорони праці та безпечного використання екранних пристроїв до початку роботи з ними, включаючи випадки модифікації та організації роботи обладнання, тобто робоче місце відповідає ДСанПІН 3.3.2.007-98.

Робоче місця працівника з екранними пристроями обране з таким устаткування, яке не створює зайвого шуму та не виділяє надлишкового тепла. Рівні шуму на робочих місцях осіб, які працюють з екранними пристроями, відповідають вимогам санітарних норм виробничого шуму, ультразвуку та інфразвуку ДСН 3.3.6.037-99.

Роботодавець за рахунок тривалості робочої зміни організує внутрішні регламентовані перерви для відпочинку відповідно до Державних санітарних правил і норм роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСанПІН 3.3.2.007-98.

Роботодавець забезпечує за свій рахунок проведення медичних оглядів працівників відповідно до вимог Порядку проведення медичних оглядів працівників певних категорій, затвердженого наказом Міністерства охорони

здоров'я України від 21 травня 2007 року № 246, зареєстрованого в Міністерстві юстиції України 23 липня 2007 року за № 846/14113 [15].

В приміщенні з робочими місцями середня температура повітря 20 °С, вологість повітря 50%, швидкість повітря 0.2 м/с, тобто воно відповідає п.1.1.1 ДСН 3.3.6.042-99.

Температура внутрішніх поверхонь робочої зони (стіни, підлога, стеля), технологічного обладнання, зовнішніх поверхонь технологічного устаткування, огороджуючих конструкцій не виходить більш ніж на 2 °С за межі оптимальних величин температури повітря для даної категорії робіт, тобто приміщення відповідає п.1.1.2 ДСН 3.3.6.042-99.

В холодний період року в приміщеннях з робочими місцями температура повітря +20 °С, відносна вологість 75%, швидкість руху повітря – 0.2 м/с. В теплий період року в приміщеннях з робочими місцями температура повітря +25 °С, відносна вологість 70%, швидкість руху повітря – 0.3 м/с. Тобто, робоче місце відповідає п.1.2.2 ДСН 3.3.6.042-99.

Температура внутрішніх поверхонь приміщень, а також температура зовнішніх поверхонь технологічного устаткування або захисних обладнань не виходить за межі допустимих величин температури повітря для даної категорії робіт, тобто відповідає п.1.2.5 ДСН 3.3.6.042-99.

Так як приміщення з робочими місцями є зі значними площами засклених поверхонь, в ньому передбачені заходи щодо захисту від перегрівання при попаданні прямих сонячних променів в теплий період року, від радіаційного охолодження - в зимовий. Тобто, робоче місце відповідає п.2.3 ДСН 3.3.6.042-99.

Проводяться попередні та періодичні медичні огляди в процесі роботи відповідно з діючим наказом МОЗ України. Тобто, робоче місце відповідає п.2.13 ДСН 3.3.6.042-99.

Вимірювання параметрів мікроклімату проводяться на робочих місцях і в робочій зоні на початку, в середині та в кінці робочої зміни. А отже, робоче місце відповідає п.3.1 ДСН 3.3.6.042-99.

Вимірювання здійснюються не менше 2-х разів на рік (теплий та холодний періоди року) у порядку поточного санітарного нагляду, а також при прийманні до експлуатації нового технологічного устаткування, внесенні технічних змін в конструкцію діючого устаткування, організації нових робочих місць тощо. При проведенні вимірювання в холодний період року температура зовнішнього повітря не вища за середню розрахункову температуру, в теплий період - не нижча за середню розрахункову температуру, що приймається для опалення та кондиціонування за оптимальними та допустимими параметрами. Тобто, робоче місце відповідає п.3.2 ДСН 3.3.6.042-99.

Вимірювання параметрів мікроклімату на робочих місцях по розробці інформаційно-електронної системи для контролю якості освіти та обміну інформації проводяться на висоті 0,5 - 1,0 м від підлоги - при роботі сидячи, 1,5 м від підлоги - при роботі стоячи. Робоче місце відповідає п.3.3 ДСН 3.3.6.042-99.

Температура та відносна вологість повітря вимірюються приладами, заснованими на психрометричних принципах. Можливе використання тижневих і добових термографів і гігрографів. Робоче місце відповідає п.3.6 ДСН 3.3.6.042-99.

Швидкість руху повітря вимірюється анемометрами ротаційної дії. Малі величини швидкості руху повітря (менше 0,3 м/сек.), особливо при наявності різноспрямованих потоків, вимірюються електро-анемометрами, циліндричними або кульовими кататермометрами. Тобто, робоче місце відповідає п.3.7 ДСН 3.3.6.042-99.

Температура поверхонь огороджуючих конструкцій або обладнань, зовнішніх поверхонь технологічного устаткування вимірюються приладами, що діють за принципом термоелектричного ефекту. Тобто, робоче місце відповідає п.3.8 ДСН 3.3.6.042-99.

Приміщення для роботи з ЕОМ розташовані не у підвальних приміщеннях та не на цокольних поверхах. Тобто, робоче місце відповідає п.2.2 ДСанПіН 3.3.2-007-98.

Природне освітлення здійснюється через світлові прорізи, орієнтовані переважно на північ та північний схід і зебезпечують коефіцієнт природною освітленості (КПО) 3%. Тобто, робоче місце відповідає п.2.5 ДСанПіН 3.3.2-007-98.

Віконні прорізи приміщень для роботи з ВДТ обладнані регульованими пристроями (жалюзі, завіски, зовнішні козирки). Тобто, робоче місце відповідає п.2.9 ДСанПіН 3.3.2-007-98.

Робоче приміщення оснащені аптечками першої медичної допомоги. Тобто, робоче місце відповідає п.2.16 ДСанПіН 3.3.2-007-98.

Зазначення освітлення освітленості на поверхні робочого столу в зоні розміщення документів становить 400 лк. Тобто, робоче місце відповідає п.3.2.3 ДСанПіН 3.3.2-007-98.

Конструкція робочого місця забезпечує підтримання оптимальної робочої пози. Тобто, робоче місце відповідає п.4.2 ДСанПіН 3.3.2-007-98.

В результаті можна сказати, що визначені стандарти охорони праці на робочих місцях розробки інформаційно-електронної системи для контролю якості навчання та обміну інформації відповідає усім розглянутим основним нормативам для робочих місць даного типу.

4.2. Забезпечення безпеки життєдіяльності при роботі з персональним комп’ютером

Комп’ютери використовуються і для офісної роботи, і для дистанційної роботи з дому, і для розваг, і для спілкування та багато іншого. Але крім корисності робота за комп’ютером може мати багато негативних факторів на користувача, дослідження безпеки життєдіяльності при роботі з персональним комп’ютером є дуже важливим. Особливо це стосується розробників, адже практично вся їхня робота пов’язана тим чи іншим чином з використанням ПК.

Вдосконалення цифрових технологій часто вимагають внесення зміни в техніку безпеки використання ПК. Для прикладу, використання LED та LCD дисплеїв комп’ютерів на заміну дисплеїв з електронно-променевою трубкою, допомогло зменшити негативний вплив екранів на очі користувача, тобто збільшити безпечно можливий час роботи за комп'ютером. Тоді як використання лазерних принтерів замість струйних призвело до необхідності виконання вентиляції озону з приміщення, так як лазери можуть генерувати цей шкідливий газ.

До заходів усунення небезпеки ураження електричним струмом зводяться до правильного розміщення устаткування та електричних кабелів. Тоді як деякі інші заходи забезпечення електробезпеки, збігаються з заходами пожежо- та електробезпеки.

Крім того варто взяти до уваги профілактичні заходи для забезпечення пожежної безпеки. До таких заходів можна віднести, використання прихованої електромережі, надійних розеток з пожежобезпечних матеріалів, силові мережі живлення устаткування виконувати кабелями, розрахованими на підключення в 3-5 разів більшого навантаження, включати й виключати живлення обладнання за допомогою штатних вимикачів. Доцільно часто робити чистку внутрішніх частин комп'ютерів, іншого устаткування від пилу, розташовувати комп'ютери на окремих неспалюваних столах. Важливим для безпеки роботи з комп'ютером є запобігання іскріння для цього потрібно рідше встромляти і виймати штепсельні вилки з розеток.

Екран дисплея повинен бути розташованим перпендикулярно до напрямку погляду інакше це призводить до відхилень статури. Відстань від дисплея до очей повинна трохи перевищувати звичну відстань між книгою та очима. Перед екраном монітора, особливо старих типів, повинен бути спеціальний захисний екран. При його відсутності треба сидіти на відстані витягнутої руки від монітора. Ще одним моментом, який стосується зору, є необхідність створення неоднорідного поля зору. Для цього можна розвісити на стінах плакати та картини, виконані у спокійних тонах.

Важливою є форма спинки крісла, яка повинна повторювати форму спини. Висота крісла повинна бути такою, щоб користувач не почував тиску на куприк або стегна. Його потрібно встановити так, щоб не треба було тягнутися до клавіатури. Періодично користувачу необхідно рухатися, вчасно змінювати положення тіла і робити перерви у роботі.

При напруженій роботі за комп'ютером щогодини необхідно робити перерву на 15 хвилин через кожну годину і треба займатися іншою справою. Декілька разів на годину бажано виконувати серію легких вправ для розслаблення. Наслідками регулярної роботи з комп'ютером без застосування захисних засобів можуть бути: захворювання органів зору, хвороби серцево-судинної системи, захворювання шлунково-кишкового тракту, шкірні захворювання.

Режим праці та відпочинку при роботі з комп'ютером залежить від категорії трудової діяльності. Всі ділять на три категорії. Перша – епізодичне зчитування і робота з інформацією не більше 2-х годин за 8-годинну робочу зміну. Друга – зчитування інформації або творча робота не більше 4-х годин за восьми годинну зміну. Третя – зчитування інформації або творча робота тривалістю більше 4-х годин за зміну.

Якщо у приміщенні експлуатується більше одного комп'ютера, то треба врахувати, що на користувача одного комп'ютера можуть впливати випромінювання від інших, в першу чергу бокових, а також і задньої стінки сусіднього дисплея. Тому необхідний захист спеціальними фільтрами і щоб користувач розміщався від бічних і задніх стінок інших дисплеїв на відстані не ближче одного метра.

Щоб запобігти негативним впливам необхідно знати й небезпечні сторони самого комп'ютера і правила безпечної роботи, знати засоби запобігання небезпек. Всесвітня організація охорони здоров'я роботу з персональним комп'ютером віднесла до небезпечних, бо їй притаманний фактор постійно діючого стресу.

Електромагнітні поля комп'ютера негативно впливають на людину і в першу чергу на її центральну нервову систему, викликаючи головний біль, запаморочення, нудоту, депресію, безсоння, відсутність апетиту, виникнення

синдрому стресу. Також не варто нехтувати і на короткі за тривалістю впливи слабких полів: змінюється гормональний стан організму, порушуються біоструми мозку. Це призводить до погіршення зору, ускладненню серцево-судинних захворювань, зниженню імунітету, виникають негативні впливи на плин вагітності.

Характерною рисою професії оператора ПК є статичний режим роботи: великий обсяг праці треба виконувати в сидячому положенні. При цьому більшість груп м'язів постійно напружені, що призводить до швидкої стомлюваності, сприяє розвитку патологічних вигинів хребта: грудному гіперкифозу, сплощенню шийного лордозу і формуванню сколіозів. Неправильне розташування дисплеїв по висоті – занадто низьке або високе, під неправильним кутом – є головною причиною появи сутулості. Занадто високе розташування дисплея призводить до тривалої напруги шийного відділу хребта, що, зрештою, може призвести до розвитку остеохондрозу.

Також зазвичай виділяють спеціальні "комп'ютерні" хвороби. Нерухома напружена поза оператора призводить до втоми і виникнення болю в хребті, шиї, плечових суглобах. Інтенсивна робота з клавіатурою викликає болючі відчуття в ліктьових суглобах, передпліччях, зап'ястях і пальцях рук. Постійні користувачі ПК найчастіше піддаються психічним стресам, хворобам серцево-судинної системи і верхніх дихальних шляхів. Значному навантаженню піддається зоровий апарат. При тривалій та інтенсивній роботі за комп'ютером з'являється синдром комп'ютерного стресу, який проявляється головним болем, запаленням очей, алергією, дратівливістю, млявістю і депресією, погіршенням зосередженості і працездатності. Їх причинами є: неправильна робота очей і поза тіла, носіння невідповідних окулярів або контактних лінз, неправильна організація робочого місця, розподілення фізичних, розумових, візуальних навантажень, низький рівень візуальної підготовленості для роботи з комп'ютером. Особливо це характерно для дітей, молодших школярів.

Існує і медико-соціальна проблема – комп'ютер і здоров'я дітей. Тепер в світі існує потужна індустрія виробництва комп'ютерних ігор. Діти з великим

задоволенням віддають цим гарним і захоплюючим, а часто і агресивним іграшкам свій вільний час. Діти в значно меншому ступені, чим дорослі, спроможні контролювати себе. їхня психіка дуже нестійка, тому надмірне захоплення комп'ютерними іграми може стати причиною дуже важких наслідків – розвивається підвищена збудженість, у школярів знижується успішність, діти стають примхливими, некерованими, перестають будь-чим цікавитися крім комп'ютера.

При роботі з не комфортною клавіатурою можуть розвиватися хвороби такі як тунельний синдром зап’ястного каналу – запалення медіального нерва руки через набряк сухожиль, синовіальної оболонки. Для цього фірма Microsoft розробила ергономічну клавіатуру, що своєрідною формою, конструкцією знижує навантаження на руки. Інший пристрій, який привертає до себе увагу фахівців в області ергономіки – маніпулятор типу "миша". На жаль, навіть найергономічніша клавіатура не може цілком вирішити проблему, оскільки причини захворювання "травми повторюваних навантажень" дотепер цілком не виявлені.

Всесвітня мережа Інтернет вже охопила великі обсяги виробництва, банківську справу, наукові дослідження, освіти – від середньої до вищої. За допомогою глобального інтернету спілкується населення Землі. Виникає єдиний організм з спільною економікою, культурою, політикою, наукою, спільним інформаційним полем. Надалі ці інтегруючі процеси будуть наростати, але це може мати і негативні наслідки, зокрема з огляду загальної, індивідуальної безпеки. Отже, щоб цьому запобігти потрібна висока комп'ютерно-інформаційна грамотність, і передусім спеціалістів у будь-якій сфері діяльності – промисловій, науковій, навчальній.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи магістра проведено заходи щодо дослідження та аналізу предметної області, було сформульовано мету виконання роботи та визначено завдання та вимоги щодо розробки інформаційної системи контролю знань та взаємодії між користувачами системи.

Безсумнівно використання програмних систем для контролю над навчання є хорошим рішенням для мінімізації турбот працівників освіти пов'язаних з веденням обліку академічної успішності, та оптимізації зв'язку з студентами, що в свою чергу дозволяє утримувати рівень контролю над якістю освіти та визначати та оцінювати знання студентів.

Досліджуючи особливості роботи предметної області було визначено інструменти реалізації програми, які необхідно використати для створення системи та реалізації визначених вимог. Таким чином увага була зосереджена на підборі інструментів для оптимізації швидкодії програми на дії користувача, та проектування гнучкої архітектури. Враховуючи те що програма буде інформаційною системою необхідно було залучити інструменти управління бази даних. Оскільки планувалося розробити програму для Windows та орієнтуючись на простоту і легкість інтегрування було обрано систему управління бази даних на основі інструментів MySQL, а саме HeidiSQL. Для реалізації програмних рішень було вирішено вибрати мову програмування C# з використанням її розширень .Net та інструментів з пакету системних зв'язків та програмних рішень Entity Framework. В якості середовища обрано Windows орієнтовану систему розробки Visual Studio з використанням різних інструментів графічної розробки програмних систем які може надати система, а також з використанням інструментів відлагодження та відстеження критичних точок програми.

Створюючи архітектури майбутньої системи вирішено виконувати розробку за використанням принципів еволюційної моделі розробки програмної системи, яка дозволяє на будь якому етапі розробки повернутися до попередніх кроків

розробки та змінити їх відповідно до потреб. Структура програми була розділена на компоненти, які виконують роль модулів, які визначають виконання певних дій чи генерацію візуальних компонентів для найкращого досвіду роботи з програмним продуктом. Кожен з визначених компонентів має власну ціль розробки та функціональну відповідальність, це дозволяє розрізнити властивості та очікувані задачі які виконуватиме той чи інший клас та унеможливити нагромадження у класах надмірного функціоналу. Також для розробленої моделі передбачено можливості подального розширення та внесення змін в поточні компоненти структури, це не впливатиме на коректність відображення реалізованих графічних та системних елементів реалізованих. Крім реалізованої програмної структури було визначено сутності елементів бази даних та зв'язків між ними побудовані на основі ER-діаграми та модифіковані з використанням інструментів адміністрування бази даних.

В результаті виконання завдань поставлених для реалізації програмної системи було створено програму, яка найкраще відповідає визначеним в процесі дослідження вимогам з реалізованим програм інтерфейсом, для швидкої адаптації для роботи з програмою. В процесі тестування програми не було визначено критичних відхилень від очікуваної роботи програми, всі дії працюють в межах визначеними програмними обмеженнями з максимально економним використанням системних ресурсів середовища тестування програмного продукту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Методичні вказівки до виконання атестаційної роботи магістра за спеціальністю 121 – Інженерія програмного забезпечення (Освітньо-професійна програма – «Програмне забезпечення систем», Освітньо-наукова програма – «Інженерія програмного забезпечення») для студентів усіх форм навчання / Упор.: М.Р. Петрик, Д.М. Михалик, О.Ю. Петрик, Г.Б. Цуприк - Тернопіль: ТНТУ, 2020-51с.

2. Офіційний сайт програми "iStudiez pro" [Електронний ресурс] – Режим доступу до ресурсу: https://istudentpro.com/.

3. Ресурс програмного забезпечення відкритого доступу "Timetable" [Електронний ресурс] – Режим доступу до ресурсу: https://soft.mydiv.net/android/download-Timetable.html.

4. Рекомендації щодо реалізації програмних рішень на мові C# [Електронний ресурс] – Режим доступу до ресурсу: https://programm.top/uk/c-sharp/.

5. Офіційний сайт Visual Studio 2019 [Електронний ресурс] – Режим доступу до ресурсу: https://visualstudio.microsoft.com/.

6. Entity Framework інструкції щодо застосування [Електронний ресурс] – Режим доступу до ресурсу: https://metanit.com/sharp/entityframework/.

7. Сайт-інструкція з використання системи HeidiSQL [Електронний ресурс] – Режим доступу до ресурсу: https://www.heidisql.com/.

8. Яровенко А. Г. Лабораторний практикум з проектування алгоритмів та програм. Навчально-методичний посібник / А. Г. Яровенко. – Вінниця, 2014. – 105 с.

9. Розбір особливостей проектування та моделювання БД [Електронний ресурс] // TextReferat. – 2018. – Режим доступу до ресурсу: http://ua.textreferat.com/referat-23369-5.html.

10. Сайт-інструкція з використання та створення Windows служб [Електронний ресурс] – Режим доступу до

ресурсу:https://learn.microsoft.com/dotnet/framework/windows-services/introduction-to-windows-service-applications

11. Настроювання безпровідної мережі в Windows [Електронний ресурс] – Режим доступу до ресурсу: https://support.microsoft.com/uk-ua/windows/настроювання-безпровідної-мережі-в-windows-97914e31-3aa4-406d-cef6-f1629e2c3721.

12. Сайт-посібник по методологіям розробок програмного забезпечення з використанням інструментів C# [Електронний ресурс] – Режим доступу до ресурсу: https://csharp-tutorials1.blogspot.com/.

13. Санітарні норми мікроклімату виробничих приміщень ДСН 3.3.6.042-99. [Електронний ресурс]. – Режим доступу: https://zakon.rada.gov.ua/rada/show/va042282-99 .

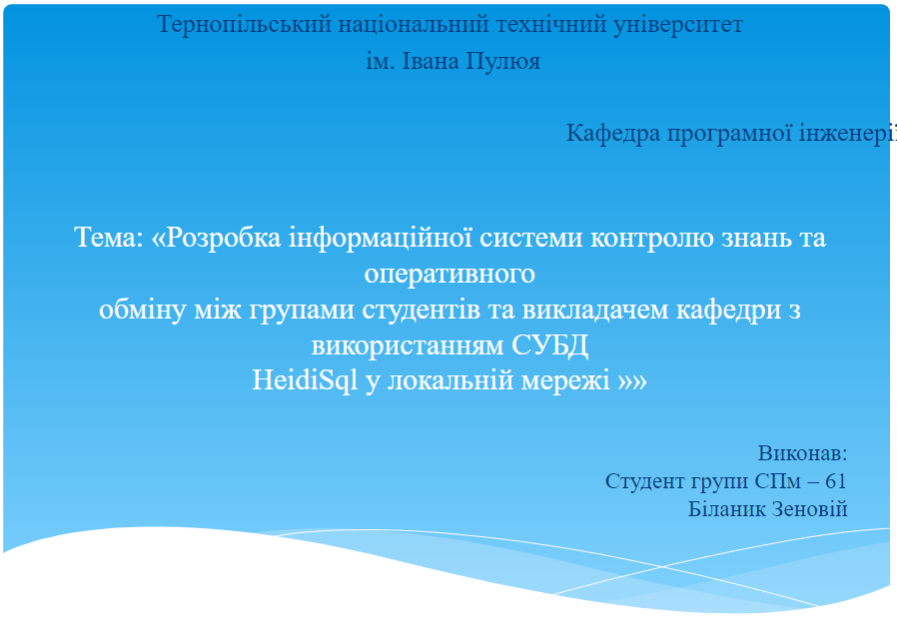
14. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСанПІН 3.3.2.007-98. [Електронний ресурс]. – Режим доступу: https://zakon.rada.gov.ua/rada/show/v0007282-98 .

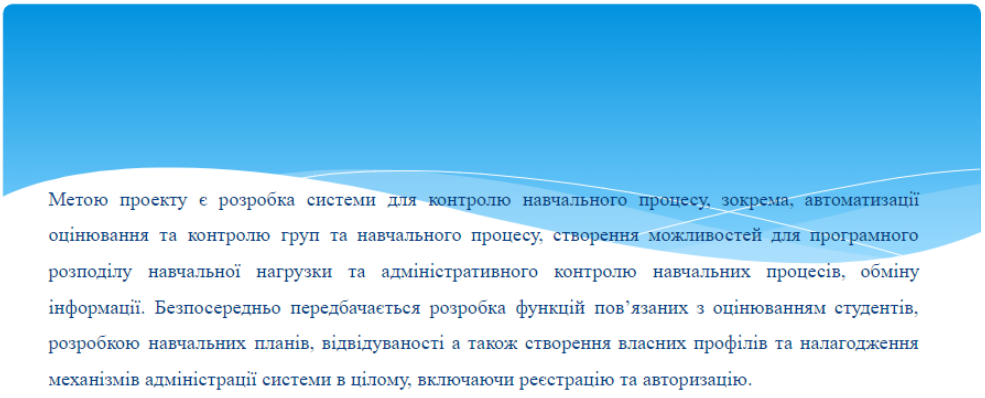
15. Наказ Міністерство охорони здоров'я України «Про затвердження Порядку проведення медичних оглядів працівників певних категорій». [Електронний ресурс]. – Режим доступу: https://zakon.rada.gov.ua/laws/show/z0846-07 .

ДОДАТКИ

Додаток А Технічне завдання

Додаток Б Слайди презентації





Вибрана модель розробки

Для реалізації тестової демонстраційної програми буде використана методологія еволюційної моделі проектування програмного забезпечення. Вона призначена для послідовного створення програмних елементів через реалізацію блоків конструкцій. Перевагою еволюційної моделі над її аналогом інкрементною моделлю є в тім, шо вона дозволяє в процесі реалізації програмних блоків додати та покращувати вимоги до системи, тим самим дозволяючи створювати вільну специфікацію для початкових етапів розробки програми.

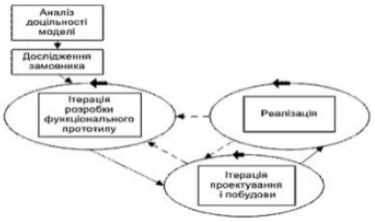


Рисунок 1 – Еволюційна модель розробки

Варіанти використання

Програма повинна виконувати наступні функції:

* авторизація користувачів;
* перегляд списку користувачів;
* редагування даних користувачів;
* додавання, редагування та видалення значень оцінок;
* додавання, редагування та видалення навчальних планів;
* контроль відвідуваності;
* регулювання рівня доступу інформації;
* генерація інформації з введених користувачем даних;
* обмін інформації між користувачами.

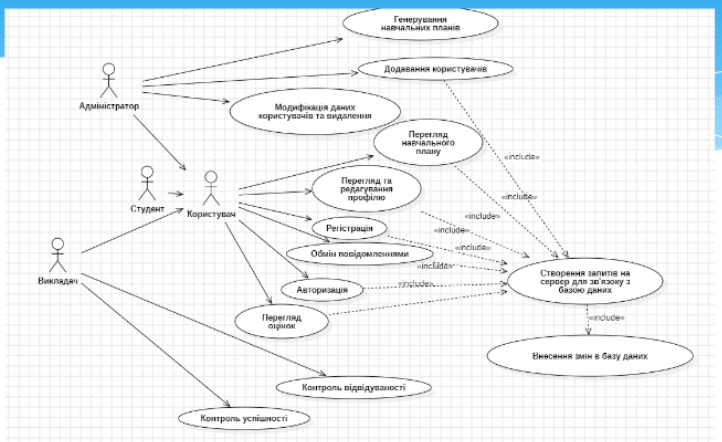


Рисунок 2 – Діаграма варіантів використання

Для реалізації цієї програми вирішено проектувати на платформі Windows з використанням мови програмування C# з використанням інструментів Visual Studio та контролю за базою даних MySQL Server.

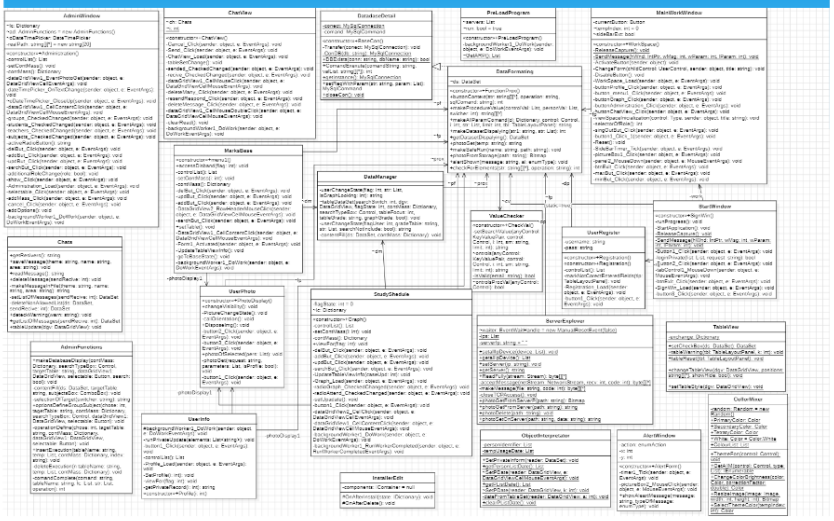
Також для нього у вільному доступі є багато програмних рішень які розширюють можливості і у значній мірі полегшують розробку програмного забезпечення.

Також використовуючи форми які оптимізують код інтерфейсу і збільшується його читабельність.

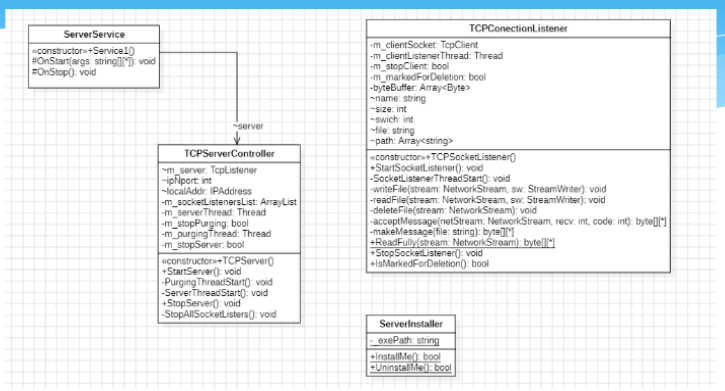
Деякі класи являють собою набір окремого функціоналу, що в комплексі зменшує навантаження на окремі елементи проекту.

Оскільки, проект надає можливість користувачеві оперувати даними (додавання, видалення, перегляд записів) було вирішено підключити базу даних з допомогою сервісу HeidiSql(MySQL), як сховище інформації .Також однією з причин вибору цієї бази було те що програма розроблена для централізованого користування (збереження даних відбуватиметься на обмеженому наборі пристроїв зі встановленою програмою) . Звідси, для перевірки впливу програми на сховище даних використовується відповіднbq програмний додаток

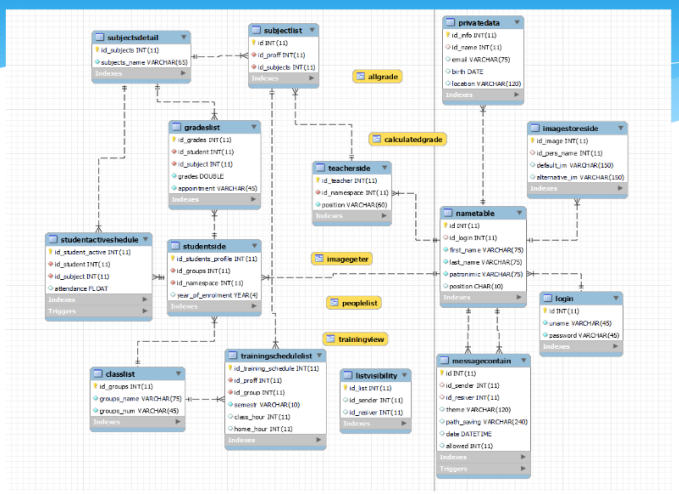
Діаграма класів клієнтської частини програми



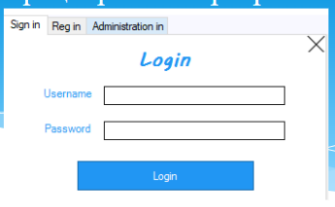
Діаграма класів серверної частини програми

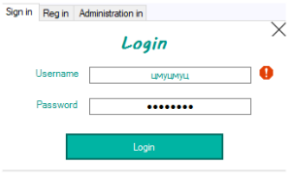


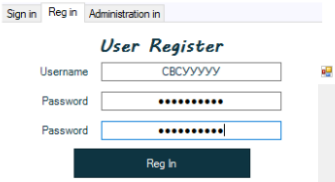
ER – діаграма сутностей

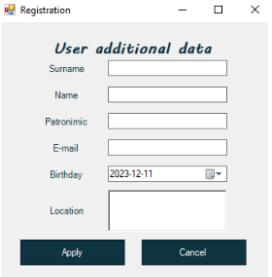


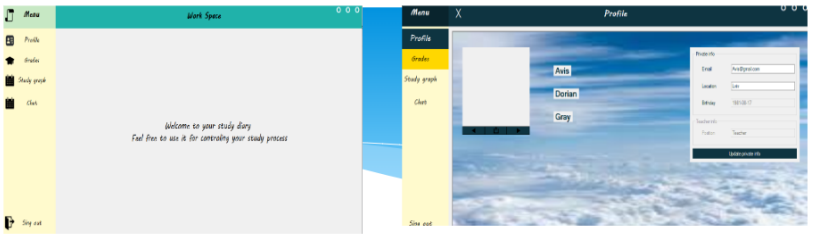
Демонстрація роботи програми

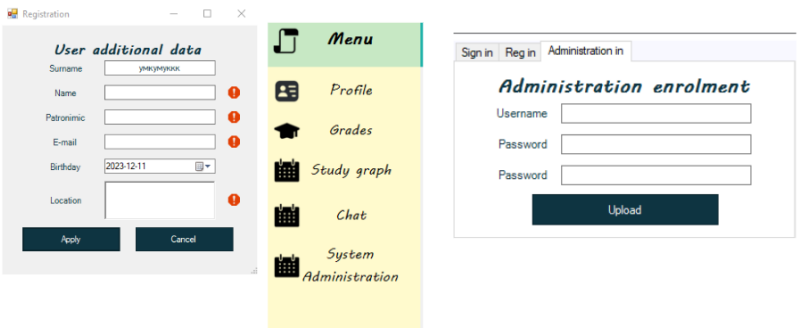


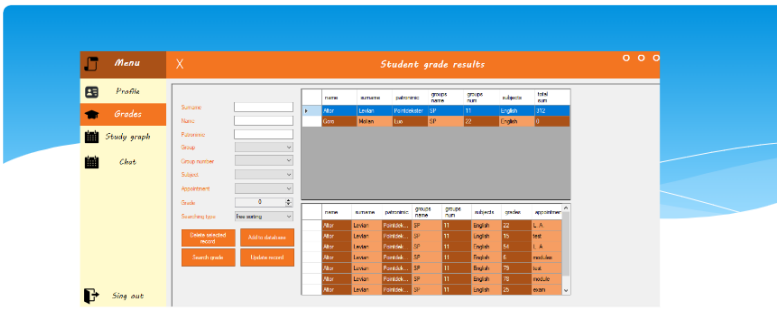


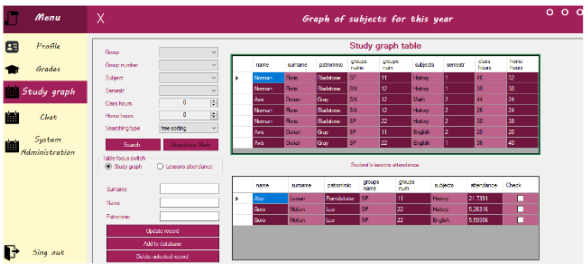


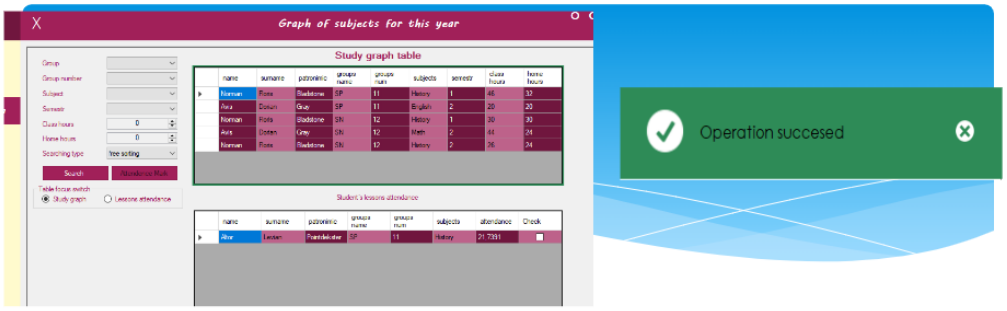



Menu
Work Space
Profile
Grades
Study graph
Chat
Sing out
Welcome to your study diary
Feel free to use it for controling your study process
Menu
Profile
Grades
Study graph
Chat
Sing out
Profile
Axis
Dorian
Gray

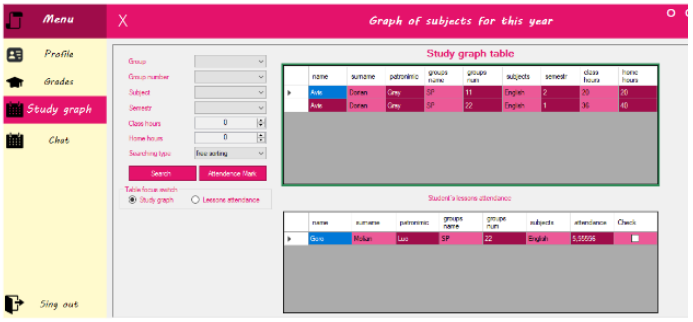

Registration
User additional data
Surname
Name
Patronimic
E-mail
Birthday
2023-12-11
Location
Apply
Cancel
Menu
Profile
Grades
Study graph
Chat
System Administration
Sign in
Reg in
Administration in
Administration enrolment
Username
Password
Password
Upload

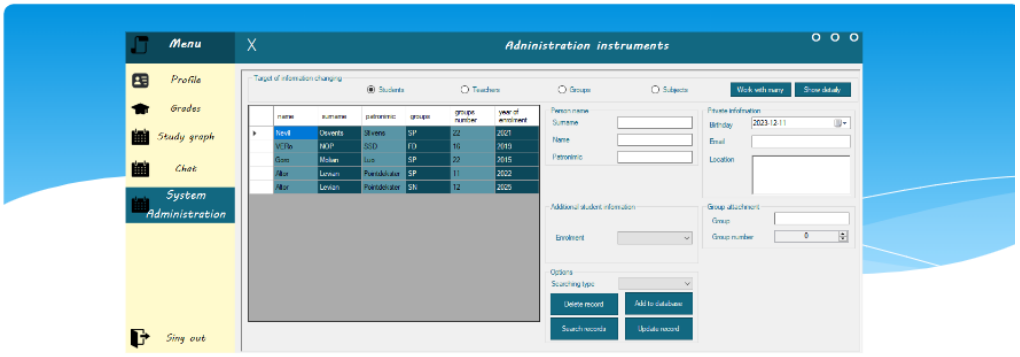

Menu
Student grade results
Profile
Grades
Study graph
Chat
Sing out

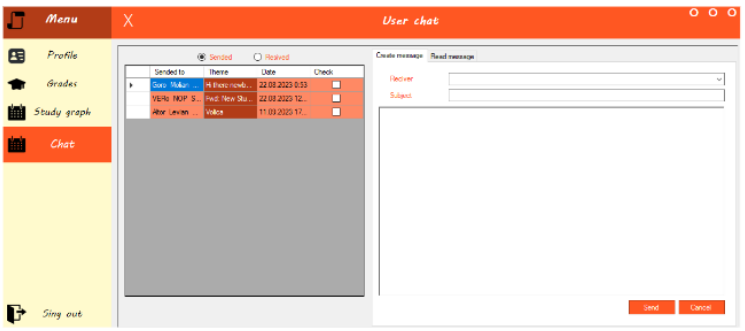

Menu
Graph of subjects for this year
Profile
Grades
Study graph
Chat
System Administration
Sing out
Study graph table


Graph of subjects for this year
Study graph table
Operation succesed


You have not entered all data for


Menu
Graph of subjects for this year
Profile
Grades
Study graph
Chat
Sing out
Study graph table





Висновки

В результаті виконання кваліфікаційної роботи магістра проведено заходи щодо дослідження та аналізу предметної області, було сформульовано мету виконання роботи та визначено завдання та вимоги щодо розробки інформаційної системи контролю знань та взаємодії між користувачами системи.

Досліджуючи особливості роботи предметної області було визначено інструменти реалізації програми, які необхідно використати для створення системи та реалізації визначених вимог. Таким чином увага була зосереджена на підборі інструментів для оптимізації швидкодії програми на дії користувача, та проектування гнучкої архітектури. Враховуючи те що програма буде інформаційною системою необхідно було залучити інструменти управління бази даних. Оскільки планувалося розробити програму для Windows та орієнтуючись на простоту і легкість інтегрування було обрано систему управління бази даних на основі інструментів MySQL, а саме HeidiSQL. Для реалізації програмних рішень було вирішено вибрати мову програмування C# з використанням її розширень .Net та інструментів з пакету системних зв'язків та програмних рішень Entity Framework. В якості середовища обрано Windows орієнтовану систему розробки Visual Studio з використанням інструментів графічної розробки програмних систем, а також з використанням інструментів відлагодження та відстеження критичних точок програми.

Створюючи архітектури майбутньої системи вирішено виконувати розробку за використанням принципів еволюційної моделі розробки програмної системи, яка дозволяє на будь якому етапі розробки повернутися до попередніх кроків розробки та змінити їх відповідно до потреб. Структура програми була розділена на компоненти, які виконують роль модулів, які визначають виконання певних дій чи генерацію візуальних компонентів для найкращого досвіду роботи з програмним продуктом. Кожен з визначених компонентів має власну ціль розробки та функціональну відповідальність, це дозволяє розрізнити властивості та очікувані задачі які виконуватиме той чи інший клас та унеможливити нагромадження у класах надмірного функціоналу. Також для розробленої моделі передбачено можливості подального розширення та внесення змін в поточні компоненти структури, це не впливатиме на коректність відображення реалізованих графічних та системних елементів реалізованих. Крім реалізованої програмної структури було визначено сутності елементів бази даних та зв'язків між ними побудовані на основі ER-діаграми та модифіковані з використанням інструментів адміністрування бази даних.

В результаті виконання завдань поставлених для реалізації програмної системи було створено програму, яка найкраще відповідає визначеним в процесі дослідження вимогам з реалізованим програм інтерфейсом, для швидкої адаптації для роботи з програмою. В процесі тестування програми не було визначено критичних відхилень від очікуваної роботи програми, всі дії працюють в межах визначеними програмними обмеженнями з максимально економним використанням системних ресурсів середовища тестування програмного продукту.

Додаток В Публікація у науковому виданні

Тернопільський національний технічний університет імені Івана Пулюя, Україна
ФІЛОСОФСЬКЕ ОСМИСЛЕННЯ ФЕНОМЕНУ ТЕХНІКИ І ЙОГО ЕВОЛЮЦІЙНИЙ РОЗВИТОК
S. Lytvynenko
PHILOSOPHICAL UNDERSTANDING OF THE PHENOMENON OF TECHNOLOGY AND ITS EVOLUTIONARY DEVELOPMENT
Л. Мосій, А. Завгородній, А. Довгань, докт. філос. наук, проф. 29
Тернопільський національний технічний університет імені Івана Пулюя, Україна
Р. РОРТІ ПРО ПРИРОДУ ФІЛОСОФСЬКОГО ЗНАННЯ
L. Mosiy, A. Zavhorodnii, A. Dovhan, Dr., Prof.
R. RORTI ABOUT THE NATURE OF PHILOSOPHICAL KNOWLEDGE
A. Осадчук 30
Тернопільський національний технічний університет імені Івана Пулюя, Україна
АНТРОПОЛОГІЯ ТЕХНІКИ : ЯК НЕЖИВЕ ЗМІНЮЄ ЖИВЕ
A. Osadchuk
ANTHROPOLOGY OF TECHNOLOGY
О. Петрик, Я. Малюта 32
Тернопільський національний технічний університет імені Івана Пулюя, Україна
ОСНОВИ ТЕХНІЧНОЇ ТВОРЧОСТІ
O. Petryk, Ya. Malyuta
BASICS OF TECHNICAL CREATIVITY

СЕКЦІЯ 2. СОЦІАЛЬНІ АСПЕКТИ ТЕХНІКИ ТА ТЕХНОЛОГІЇ 33

H. Jadav, H. Shchyhelska, Ph.D., Assoc. Prof. 33
Ternopil Ivan Puluj National Technical University, Ukraine
EVOLUTION OF ARTIFICIAL INTELLIGENCE PAINTING
Х. Джадав, Г. Щигельська, канд. істор. наук, доц.
ЕВОЛЮЦІЯ У СФЕРІ ЖИВОПИСУ, СТВОРЕНОГО ШТУЧНИМ ІНТЕЛЕКТОМ
З. Біланик, Н. Шостаківська, канд. пед. наук, доц. 35
Тернопільський національний технічний університет імені Івана Пулюя, Україна
ЦИФРОВІЗАЦІЯ СУСПІЛЬНИХ ПРОЦЕСІВ
Z. Bilanyk, N. Shostakivska, PhD in pedagogy, Assoc. Prof
DIGITALIZATION OF SOCIAL PROCESSES
С. Гесюк 36
Тернопільський національний технічний університет імені Івана Пулюя, Україна
ШТУЧНИЙ ІНТЕЛЕКТ ТА ЕМОЦІЇ
S. Hesiuk
ARTIFICIAL INTELLIGENCE AND EMOTIONS
Городиська Н. 39
Галицький фаховий коледж імені В'ячеслава Чорновола, Україна
СОЦІАЛЬНИЙ ПРОГРЕС В УМОВАХ ДІДЖИТАЛІЗАЦІЇ
Horodyska N.
SOCIAL PROGRESS IN THE CONDITIONS OF DIGITALIZATION
Г. Груць, канд. пед. наук, доц. 41
Тернопільський національний педагогічний університет імені Володимира Гнатюка, Україна
МОРАЛЬНИЙ АСПЕКТ НАУКОВО-ТЕХНІЧНОГО ПРОГРЕСУ
H. Hruts, Ph.D., Assoc. Prof
THE MORAL ASPECT OF SCIENTIFIC AND TECHNOLOGICAL PROGRESS
О. Дігай, Ю. Гумен канд. істор. наук, доц. 42
Тернопільський національний технічний університет імені Івана Пулюя, Україна
ПЕРЕВАГИ АДАПТАЦІЇ РЕЛІГІЙНОГО СВІТОГЛЯДУ В УМОВАХ ВОЄННОГО СТАНУ
O. Dihai, Yu. Humen, PhD, Assoc. Prof.
ADVANTAGES OF THE ADAPTATION OF A RELIGIOUS OUTLOOK IN THE CONDITIONS OF THE STATE OF MARTIAL
Т.Жук 44
Тернопільський національний технічний університет імені Івана Пулюя, Україна
ПРОБЛЕМА ЛЕГАЛІЗАЦІЇ ЗБРОЇ В УКРАЇНІ
T.Zhuk
THE PROBLEM OF WEAPONS LEGALIZATION IN UKRAINE
Н. Захарова А., В. Моллекер 45
Міжнародний науково-технічний університет імені академіка Юрія Бугая, Україна

З. Біланик, Н. Шостаківська, канд. пед. наук, доц.
Тернопільський національний технічний університет імені Івана Пулюя, Україна

ЦИФРОВІЗАЦІЯ СУСПІЛЬНИХ ПРОЦЕСІВ

Z. Bilanyk, N. Shostakivska, PhD in pedagogy, Assoc. Prof.

DIGITALIZATION OF SOCIAL PROCESSES

З плином часу людство проходить етапи модернізації та вдосконалення робочих процесів, метою яких є спрощення та підвищення ефективності виконання завдань виробництва. В результаті виникають нові технології та методології, що дозволяють оптимізувати витрати та покращити якість отриманого результату. Оскільки, технічний розвиток перш за все спрямований на покращення умов роботи та життєдіяльності

суспільства, він спричиняє культурний та інтелектуальний ріст кожного індивіда, що є учасником та об'єктом соціальних процесів, спричинених прогресом.

З приходом нових знань та технологій, суспільство удосконалюється, створює нові інструменти та методології для подолання складних ситуацій. Так і зараз в час глобальної цифровізації людство активно створює та удосконалює механізми реагування та підтримки соціальних процесів. З використанням комп'ютерних систем проводиться створення нових технологій збереження та маніпулювання даними, спрощуються адміністративні та комунальні процедури, стає доступнішою інформація, з'являються перспективи освітнього росту та розвитку суспільства. Дедалі більше набуває популярності цифровізація різних послуг: від замовлення кур'єрських послуг до реєстрації власної компанії.

Впровадження комп'ютерних систем в різні сфери людської діяльності неминуче призведе до оптимізації та реструктуризації, спростить складні процедури документообігу, покращить мобільність і продуктивність. Крім того цифровізація призведе до запобігання помилок в ході роботи, уникнення затримок суспільних процесів, покращення оперативності в надзвичайних ситуаціях.

Застосування цифровізованих технологій створить можливості проводить роботи в місцях, де людина не здатна перебувати, знаходити та аналізувати інформацію швидше, відстежувати стан об'єктів та процесів, та інше.

Надзвичайний потенціал вони мають в розвитку освіти та науки. Користуючись механізмами цифровізації в процесі навчання, викладачі зможуть створити власні методології навчання з можливістю зміни підходів, та інструментів, розширити освітню програму, що дозволить збільшити цікавість учнів до уроку. Студенти отримають можливість підвищити швидкість засвоєння знань та навичок, поглибити розуміння теми, вільний доступ до джерел інформації.

Цифровізація розкриває широкі перспективи для покращення суспільних процесів. Залучення її до вирішення побутових, промислових та соціальних проблем призведе до збільшення інструментів та методів, що дозволить розробити механізми реагування на більшість існуючих потреб суспільства.

Література

1. Маркевич К. Цифровізація: переваги та шляхи подолання викликів [Електронний ресурс]. Режим доступу до ресурсу: https://razumkov.org.ua/statti/tsyfrovizatsiia-perevagy-ta-shliakhy-podolannia-vyklykiv
2. Scott J. The Digitalization of Society [Електронний ресурс]. Режим доступу до ресурсу: https://library.acropolis.org/the-digitalization-of-society/