ПОЛТАВСЬКИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТОРГІВЛІ
Навчально-науковий інститут денної освіти
Форма навчання денна
Кафедра комп'ютерних наук та інформаційних технологій

Допускається до захисту
Завідувач кафедри
_____ Олена ОЛЬХОВСЬКА
(підпис)
«___» _____ 202_ р.

КВАЛІФІКАЦІЙНА РОБОТА
на тему:
«РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ПРОГРАМИ ОБЛІКУ УСПІШНОСТІ СТУДЕНТІВ»

зі спеціальності 122 Комп'ютерні науки
освітня програма «Комп'ютерні науки»
ступеня магістра
Виконавець роботи Оразов Вепа
_____ «___»_____202_ р.
(підпис)
Науковий керівник к. п. н., доцент, Кошова Оксана Петрівна
_____ «___»_____202_ р.
(підпис)
Рецензент _____

Полтава – 202_

ЗАТВЕРДЖУЮ
Завідувач кафедри _____ Олена ОЛЬХОВСЬКА
«___» вересня 202_р.

ЗАВДАННЯ І КАЛЕНДАРНИЙ ГРАФІК
ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

на тему «Розробка програмного забезпечення програми обліку успішності студентів»
зі спеціальності 122 Комп'ютерні науки»
освітня програма «Комп'ютерні науки»
ступеня магістр
Прізвище, ім'я, по батькові Оразов Вепа_____
Затверджена наказом ректора № ___-Н від «__» _____ 202__ р.
Термін подання студентом роботи «__» _____ 202__ р.
Вихідні дані до кваліфікаційної роботи: публікації з теми та програмне забезпечення аналогічного призначення.
Зміст пояснювальної записки (перелік питань, які потрібно розробити)
ВСТУП
РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ ТА ІНФОРМАЦІЙНИЙ ОГЛЯД
1.1 Перегляд основ систем обліку успішності в навчальному процесі
1.2 Огляд подібних програмних продуктів
1.3 Позитивні аспекти оглянутих програмних продуктів
1.4 Негативні аспекти оглянутих програмних продуктів
РОЗДІЛ 2. ОГЛЯД ТЕХНОЛОГІЙ ТА ЗАСОБІВ РОЗРОБКИ
2.1 Мова програмування Python
2.2 Середовище розробки PyCharm
2.3 Python модуль Tkinter
2.4 Python модуль SQLite3
РОЗДІЛ 3. ПРАКТИЧНА ЧАСТИНА
3.1 Обґрунтування вибору мови програмування
3.2 Обґрунтування вибору програмних засобів
3.3 Опис програмної реалізації
3.4 Тестування створеного продукту
ВИСНОВКИ
СПИСОК ЛІТЕРАТУРИ
ДОДАТОК А. ВИХІДНІ КОДИ

Перелік графічного матеріалу: 1 блок-схема, інші необхідні ілюстрації.
Консультанти розділів кваліфікаційної роботи

Розділ	ПІП, посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Постановка задачі та інформаційний огляд	Кошова О.П.		
Огляд технологій та засобів розробки	Кошова О.П.		
Практична частина	Кошова О.П		

Календарний графік виконання кваліфікаційної роботи

Зміст роботи	Термін виконання	Фактичне виконання
1. Вступ		
2. Вивчення методичних рекомендацій і стандартів та звіт керівнику		
3. Постановка задачі		
4. Інформаційний огляд джерел бібліотек та інтернету		
5. Огляд технологій та засобів розробки		
6. Практична частина		
7. Закінчення оформлення		
8. Доповідь студента на кафедрі		
9. Доопрацювання (за необхідності), рецензування		

Дата видачі завдання «__» _____ 202_р.
Здобувач вищої освіти _____ Вепа ОРАЗОВ
(підпис)
Науковий керівник _____ к.пед.н., доц., Кошова О.П.
(підпис) (науковий ступінь, вчене звання, ініціали та прізвище)

Результати захисту кваліфікаційної роботи

Кваліфікаційна робота оцінена на _____
Протокол засідання ЕК №____ від «____»_____ 202_ р.
Секретар ЕК _____ _____
(підпис) *(ініціал та прізвище)*

Затверджую

Зав. кафедрою _____

к.ф.-м.н. Олена Ольховська

«____» _____ 202__ р.

Погоджено

Науковий керівник _____

к.п.н. Оксана КОШОВА

«____» _____ 202__ р.

План

кваліфікаційної роботи ступеня магістр
зі спеціальності 122 Комп'ютерні науки
освітня програма 122 Комп'ютерні науки
Оразов Вепа
Прізвище, ім'я

на тему «Розробка програмного забезпечення програми обліку успішності студентів»

Здобувач вищої освіти _____ Вепа ОРАЗОВ
«____» _____ 202__ р.

АНОТАЦІЯ

Записка: 55 с., 5 рис., 1 додаток, 1 блок-схема, 13 джерел.

Мета роботи – Алгоритмізація та створення програмного забезпечення з теми "Розробка програмного забезпечення програми обліку успішності студентів".

Об’єкт розробки – є програмна реалізація навчального з теми «Розробка програмного забезпечення програми обліку успішності студентів»

Методи дослідження та інформаційне забезпечення - бази даних sqlite3 середовище розробки PyCharm та мова програмування Python, модуля Tkinter там sqlite3.

Результати дослідження. Розроблено алгоритм роботи програмного забезпечення з теми "Розробка програмного забезпечення програми обліку успішності студентів", побудовано його блок-схему. Здійснена програмна реалізація в середовищі PyCharm за допомогою мови програмування Python, модуля Tkinter та sqlite3 проведення тестування створеного продукту.

Ключові слова*: sqlite3, Tkinter, програмне забезпечення, PyCharm, мова програмування Python, програмна реалізація.*

ЗМІСТ

ВСТУП

Розробка програмного забезпечення для обліку успішності студентів є важливим завданням, яке потребує детального аналізу та інноваційних підходів. В сучасних умовах цифровізації освіти, ефективне управління академічними даними студентів стає критичним фактором успіху як для навчальних закладів, так і для самих студентів. Завдання полягає в створенні інструменту, який би забезпечував зручний та ефективний процес зберігання, обробки та аналізу даних про успішність студентів, що сприятиме підвищенню якості освіти та полегшенню адміністративних процесів.

Необхідність розробки програмного забезпечення для обліку успішності студентів обумовлена зростанням обсягів інформації, що потребує обробки в навчальних закладах, а також вимогами до підвищення прозорості та точності обліку академічних досягнень. Відсутність єдиного стандартизованого рішення ускладнює процеси моніторингу та аналізу даних, що, в свою чергу, впливає на ефективність управління навчальним процесом. Вихідними даними для дослідження є потреби навчальних закладів, існуючі програмні продукти, нормативні документи, що регулюють облік успішності студентів, та сучасні технології розробки програмного забезпечення.

Проведення даного дослідження є необхідним з огляду на численні проблеми, з якими стикаються навчальні заклади у процесі обліку успішності студентів. Це включає в себе труднощі з обробкою великих обсягів даних, помилки у збереженні та передачі інформації, відсутність можливості оперативного доступу до необхідних даних, а також відсутність зручних інструментів для аналізу та візуалізації результатів навчання. Вирішення цих проблем сприятиме підвищенню ефективності управління навчальним процесом, покращенню якості освіти та забезпеченню більшої прозорості в оцінюванні знань студентів.

У роботі буде представлено аналіз існуючих рішень для обліку успішності студентів, їх переваги та недоліки. Буде обґрунтовано вибір методології та інструментів розробки програмного забезпечення, запропоновано архітектуру системи та детально

описано етапи її реалізації. Також буде проведено тестування розробленого програмного продукту та представлено результати його впровадження в навчальний процес.

Актуальність даної роботи зумовлена необхідністю модернізації систем управління навчальними закладами відповідно до сучасних вимог цифрового суспільства. Впровадження ефективного програмного забезпечення для обліку успішності студентів сприятиме підвищенню якості освітнього процесу, забезпеченню своєчасного та точного обліку академічних результатів, а також полегшенню роботи адміністративного персоналу. Порівняння з існуючими рішеннями демонструє, що розробка нового програмного продукту, орієнтованого на сучасні технологічні стандарти, є доцільною та своєчасною.

Таким чином, розробка програмного забезпечення для обліку успішності студентів є важливим кроком у напрямку підвищення ефективності та якості управління навчальним процесом, що обумовлює актуальність та значущість проведеного дослідження для розвитку освітньої галузі.

Мета проєкту – Алгоритмізація та створення програмного забезпечення з теми "Розробка програмного забезпечення програми обліку успішності студентів".

Об’єктом розробки є програмна реалізація обліку успішності студентів, блок-схема.

Предмет розробки — програмне забезпечення обліку успішності студентів.

Перелік використаних методів: при створенні програми було використано мову програмування Python, модулі Tkinter, sqlite3;

Пояснювальна записка складається з трьох розділів. У першому розділі розглянуто постановку задачі та оглянуто подібні програмні продукти та виділено позитивні та негативні аспекти. У другому розділі було представлено огляд технологій та засобів розробки. У третьому розділі розглянуто якими методами створювалось програмне забезпечення та яке програмне середовище було обрано для розробки.

У додатку міститься вихідний код проєкту.

Обсяг пояснювальної записки: 55 стор., в т.ч. основна частина - 27 стор., додатки - 17 стор., 5 рис., джерела – 13 назв.

Розробка програмного забезпечення програми обліку успішності студентів врахування сучасних тенденцій у навчанні та використання технологій дозволяють ефективно оптимізувати навчальний процес, надаючи можливість студентам та викладачам автоматизувати рутинні завдання, покращувати управління даними та підвищувати прозорість і доступність інформації. Такі системи сприяють оперативному відстеженню академічних досягнень, забезпечуючи своєчасний доступ до результатів навчання, аналітичних звітів та рекомендацій щодо подальшого розвитку.

Враховуючи сучасні вимоги до методів навчання, створений програмний продукт може бути інтегрованим в освітній процес, сприяючи підвищенню актуальності інформації обліку студентів.

В рамках роботи створено засіб для обліку оцінювання студентів та впроваджено інтерактивні функції, які спрямовані на практичне використання у реальних ситуаціях. Це додає практичну цінність до навчання.

Ця програма представляє собою унікальний продукт, який був розроблений власноруч. Було запрограмовано графічний інтерфейс, а також вбудовано звязок з таблицею для додавання, виведення та видалення даних.

РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ ТА ІНФОРМАЦІЙНИЙ ОГЛЯД

1.1 Перегляд основ систем обліку успішності в навчальному процесі

В рамках кваліфікаційної роботи поставлено завдання зробити програму обліку успішності студентів, яка буде мати простий дизайн, та зрозумілий функціонал, що передбачає оцінювання, датування, ведення списку студентів.

Система обліку успішності студентів являє собою комплекс програмних засобів для автоматизації моніторингу та аналізу навчальних досягнень студентів. Вона включає електронні журнали, де викладачі вводять оцінки та відвідуваність студентів в електронному вигляді. Це забезпечує зручність використання та доступність даних для всіх учасників освітнього процесу. Системи автоматично аналізують успішність студентів, що дозволяє швидко виявляти проблеми в навчальному процесі та оперативно реагувати на них. Вони забезпечують прозорість та відкритий доступ до інформації про успішність, що підвищує довіру до системи оцінювання. Зібрані дані дозволяють приймати обґрунтовані рішення щодо покращення методик навчання та підвищення мотивації студентів, які можуть самостійно стежити за своїм прогресом [2].

Ці системи використовуються в навчальних закладах різних рівнів для ефективного управління навчальним процесом і забезпечення прозорості оцінювання. Основні функції системи обліку успішності студентів включають збір даних, зберігання даних, аналіз даних, управління розкладом, зворотній зв’язок, персоналізоване навчання, прозорість та доступність, інтеграцію з іншими системами. Збір даних здійснюється через введення оцінок та інших даних про успішність студентів в електронні журнали, які інтегровані з системою. Усі ці дані зберігаються у централізованій базі даних.

Система надає інструменти для аналізу академічних даних, включаючи статистичні звіти, графіки успішності, виявлення тенденцій та прогалин у знаннях студентів. Деякі системи обліку успішності включають функції управління розкладом

занять, що дозволяє оптимізувати час викладання та забезпечити ефективне використання ресурсів. Системи обліку успішності можуть бути інтегровані з іншими освітніми платформами, такими як системи управління навчанням, бібліотечні системи, платформи онлайн-курсів тощо.

Викладачі можуть надавати студентам зворотній зв'язок щодо їхніх оцінок та успіхів, а також рекомендації для покращення результатів.

Студенти мають змогу переглядати свої оцінки, графіки успішності та отримувати рекомендації в режимі реального часу через особисті кабінети в системі.

Після запуску програми, користувач буде бачити інтерфейс, за допомогою якого він може додати або видалити студента, записати, або видалити оцінку, додати, або видалити предмет, вказати дату оцінювання, побачити список усіх предметів, знайти студента за прізвищем, чи подивитись список всіх студентів.

1.2 Огляд подібних програмних продуктів

Електронний щоденник NZ.UA [3] є сучасною системою для управління навчальним процесом у школах України. Ця платформа надає зручні інструменти для вчителів, студентів та їхніх батьків, дозволяючи ефективно вести облік успішності та організовувати навчальний процес. NZ.UA була створена з метою спрощення та автоматизації процесів управління освітою. Вона дозволяє вчителям вводити оцінки, створювати розклади занять, публікувати домашні завдання та ділитися навчальними матеріалами. Студенти та їхні батьки можуть у будь-який момент переглянути результати навчання, отримати доступ до розкладу та дізнатися про майбутні події. Система дозволяє вчителям швидко та зручно вводити оцінки за різні види робіт, включаючи контрольні роботи, домашні завдання, заліки та екзамени. Вона автоматично обробляє введені дані та надає можливість формування звітів про успішність кожного студента. NZ.UA надає функціонал для створення та публікації розкладів занять. Вчителі можуть додавати події, такі як контрольні роботи, екзамени

та шкільні заходи, що дозволяє студентам та їхнім батькам завжди бути в курсі важливих дат. Вчителі мають можливість публікувати домашні завдання, а студенти можуть переглядати їх та завантажувати виконані роботи безпосередньо через платформу. Це значно спрощує процес комунікації між викладачами та студентами. NZ.UA підтримує обмін повідомленнями між учасниками освітнього процесу. Вчителі можуть надсилати повідомлення студентам та їхнім батькам, що дозволяє оперативно обмінюватися важливою інформацією. Система надає інструменти для аналізу успішності студентів. Вчителі та адміністрація школи можуть створювати звіти про успішність окремих класів або студентів, що допомагає у прийнятті обґрунтованих рішень щодо покращення навчального процесу. NZ.UA має зручний та інтуїтивно зрозумілий інтерфейс, який дозволяє легко орієнтуватися у системі. Платформа доступна через веб-браузер, що робить її доступною з будь-якого пристрою з підключенням до Інтернету. Використання електронного щоденника NZ.UA надає численні переваги для всіх учасників навчального процесу. Вчителі отримують зручний інструмент для ведення обліку успішності та організації навчального процесу, студенти – доступ до необхідної інформації та зворотного зв'язку, а батьки – можливість стежити за успішністю своїх дітей та бути в курсі всіх подій у школі. NZ.UA є потужною та зручною платформою, яка сприяє підвищенню ефективності навчального процесу та покращенню комунікації між вчителями, студентами та їхніми батьками. Завдяки своїм функціональним можливостям та інтуїтивно зрозумілому інтерфейсу, вона стає незамінним інструментом у сучасній освіті.

Moodle [4], або Modular Object-Oriented Dynamic Learning Environment, є системою управління навчанням (Learning Management System, LMS), яка широко використовується у всьому світі. Перша версія Moodle була випущена у серпні 2002 року. Moodle дозволяє створювати, адмініструвати та проводити онлайн-курси. Вчителі можуть вести облік оцінок студентів, зберігати результати тестів та інших форм оцінювання. Система пропонує інтерактивні можливості, такі як форуми, чати та відеоконференції, що дозволяє викладачам і студентам активно взаємодіяти. Moodle

також підтримує інтеграцію з іншими освітніми ресурсами та інструментами, такими як Google Docs і Zoom, і надає інструменти для створення звітів та аналізу успішності студентів. Для використання Moodle необхідно встановити систему на сервері навчального закладу, після чого адміністратори можуть налаштувати курси та додавати користувачів. Студенти та викладачі можуть отримувати доступ до курсів через веб-інтерфейс або мобільний додаток.

Blackboard [5] є комплексною системою для управління навчанням, яка включає інструменти для обліку успішності студентів та організації навчального процесу. Заснована у 1997 році, Blackboard Inc. випустила першу версію свого продукту у 1998 році. Blackboard дозволяє створювати та адмініструвати курси, надавати доступ до навчальних матеріалів і виставляти оцінки. Викладачі можуть коментувати роботи студентів, проводити тести та опитування. Система має потужні аналітичні інструменти для аналізу успішності студентів та звіти про їх активність і прогрес. Blackboard також забезпечує засоби для обміну повідомленнями, організації дискусій і проведення вебінарів, а мобільні додатки дозволяють студентам та викладачам використовувати систему з будь-якого місця. Щоб користуватися Blackboard, навчальний заклад повинен придбати ліцензію, після чого система налаштовується відповідно до потреб закладу. Користувачі отримують доступ через веб-браузер або мобільні додатки, де можуть переглядати курси, завдання та оцінки.

Google Classroom [6], запущений у серпні 2014 року, є безкоштовною веб-службою, розробленою Google для навчальних закладів. Google Classroom інтегрується з іншими сервісами Google, такими як Google Drive і Google Docs, що дозволяє викладачам створювати віртуальні класи, додавати завдання та навчальні матеріали, а також оцінювати завдання. Студенти мають доступ до своїх оцінок і можуть спільно працювати над документами, презентаціями та іншими проектами. Вбудований чат та функція коментування забезпечують зручну комунікацію між учасниками навчального процесу. Для використання Google Classroom необхідно мати обліковий запис Google. Викладачі можуть створювати класи, додавати студентів через

їхні електронні адреси та керувати завданнями і матеріалами. Студенти отримують сповіщення про нові завдання та можуть здавати роботи онлайн.

Canvas [7], створена у 2011 році, є інноваційною системою управління навчанням, призначеною для покращення навчального процесу в освітніх установах. Canvas дозволяє створювати, адмініструвати та управляти онлайн-курсами, виставляти оцінки та надавати зворотний зв'язок. Система пропонує додатки для мобільних пристроїв, що дозволяють використовувати її на ходу, а також потужні аналітичні інструменти для відстеження прогресу та успішності студентів. Canvas підтримує інтеграцію з численними іншими освітніми інструментами та платформами, що робить її дуже гнучкою і універсальною для різних освітніх потреб. Для використання Canvas навчальний заклад повинен придбати ліцензію, після чого система налаштовується відповідно до потреб закладу. Користувачі можуть отримувати доступ до курсів через веб-браузер або мобільний додаток, де можуть переглядати матеріали, виконувати завдання та отримувати оцінки.

1.3. Позитивні аспекти оглянутих програмних продуктів

Розглянуті програми мають такі позитивні аспекти:

1. Електронний щоденник NZ.UA:

Система доступна через веб-браузер з будь-якого пристрою, що дозволяє студентам та батькам переглядати інформацію у будь-який час. Вчителі можуть швидко вводити оцінки, створювати розклади, публікувати домашні завдання та ділитися навчальними матеріалами. Платформа підтримує обмін повідомленнями між учасниками освітнього процесу, що покращує комунікацію між вчителями, студентами та батьками. Використовуються сучасні технології шифрування для забезпечення конфіденційності даних. Інструменти для аналізу успішності студентів та створення звітів допомагають у прийнятті обґрунтованих рішень щодо покращення навчального процесу. Сервіс повністю безкоштовний.

2. Moodle:

Підтримує різні модулі та плагіни, що дозволяє адаптувати систему під конкретні потреби навчального закладу. Забезпечує інтерактивні інструменти для онлайн-курсів, включаючи форуми, відеоконференції, тести та завдання. Можливість інтеграції з різними освітніми платформами та інструментами для обліку успішності студентів. Велика кількість користувачів та активна спільнота, яка надає підтримку та ділиться досвідом.

3. Blackboard:

Включає всі необхідні інструменти для управління навчальним процесом, комунікації та обліку успішності студентів. Високий рівень захисту даних та дотримання стандартів конфіденційності. Підтримка мобільних додатків для зручного доступу до матеріалів та комунікації з будь-якого місця. Розвинуті інструменти аналітики та звітності, що допомагають відстежувати успішність студентів та ефективність навчального процесу.

4. Google Classroom:

Платформа є безкоштовною для навчальних закладів. Тісна інтеграція з іншими сервісами Google, такими як Google Drive і Google Docs.Інтуїтивно зрозумілий інтерфейс. Вбудований чат та функція коментування забезпечують зручну комунікацію.

5. Canvas:

Canvas пропонує сучасні інструменти для управління навчанням. Додатки для мобільних пристроїв дозволяють використовувати платформу на ходу. Потужні інструменти для аналізу успішності студентів. Підтримує інтеграцію з численними освітніми інструментами.

1.4. Негативні аспекти оглянутих програмних продуктів

Розглянуті програми мають такі негативні аспекти:

1. Електронний щоденник NZ.UA:

Деякі користувачі можуть потребувати часу для освоєння платформи. Для доступу до інформації необхідний стабільний інтернет-зв'язок, що може бути проблематичним у деяких регіонах. Платформа призначена виключно для шкіл, що обмежує її використання у інших освітніх закладах, таких як університети чи коледжі.

2. Moodle:

Потребує певного рівня технічних знань для встановлення та налаштування, що може бути складним для деяких навчальних закладів. Деякі користувачі вважають інтерфейс менш інтуїтивно зрозумілим, що може вимагати додаткового навчання. Необхідність власного сервера для розміщення системи.

3. Blackboard:

Необхідність придбання ліцензії, що може бути дорогим для деяких закладів. Викладачам та студентам може знадобитися час на освоєння всіх функцій системи. Потреба у технічній підтримці для налаштування та обслуговування системи.

4. Google Classroom:

Необхідність мати обліковий запис Google для користування системою. Можливості системи можуть бути недостатніми для складних або специфічних потреб деяких навчальних закладів. Використання сервісів Google може викликати занепокоєння щодо конфіденційності даних.

5. Canvas:

Необхідність придбання ліцензії для використання системи. Потреба у часі для освоєння всіх функцій та можливостей платформи. Потреба у технічній підтримці для налаштування та обслуговування системи.

РОЗДІЛ 2. ОГЛЯД ТЕХНОЛОГІЙ ТА ЗАСОБІВ РОЗРОБКИ

2.1 Мова програмування Python

Python – це високорівнева мова програмування загального призначення, яка була створена, щоб бути простою у використанні та читабельною. Вона підтримує кілька програмних парадигм, включаючи об'єктно-орієнтоване, процедурне та функціональне програмування (Рисунок 2.1) [8].

Рисунок 2.1 - Логотип мови програмування Python

Сумісність з девайсами

Python був розроблений Гвідо ван Россумом, голландським програмістом, у кінці 1980-х років. Перша версія Python була випущена у 1991 році. Ван Россум почав роботу над Python як проектом для заповнення часу під час різдвяних канікул. Він прагнув створити мову, яка була б більш читабельною і простішою у використанні, ніж існуючі мови програмування [9].

Python має кілька видатних особливостей, які роблять його привабливим для програмістів. Однією з ключових особливостей є простота і читабельність. Синтаксис Python легкий для розуміння та використання, що робить його ідеальним для новачків. Наприклад, блоки коду визначаються відступами, а не фігурними дужками, що значно покращує читабельність [10].

Python також пропонує високий рівень абстракції, абстрагуючи складності, пов'язані з керуванням пам'яттю та низькорівневими операціями. Це дозволяє програмістам зосередитись на розв'язанні бізнес-логіки, замість витрачання часу на технічні деталі. Багатопарадигмовість є ще однією сильною стороною Python: він підтримує кілька програмних парадигм, включаючи об'єктно-орієнтоване, процедурне та функціональне програмування.

Python постачається з великою стандартною бібліотекою, яка забезпечує широкий спектр функціональних можливостей. Вона включає інструменти для роботи з файлами, веб-програмування, наукові обчислення, машинне навчання тощо. Це робить Python надзвичайно універсальним інструментом для вирішення різноманітних задач .

Інтерактивність і динамічність Python дозволяють програмістам швидко тестувати і змінювати код. Python є динамічно типізованою мовою, що означає визначення типу змінної під час виконання програми, а не під час компіляції, що спрощує і прискорює процес розробки. Завдяки своїй переносимості, програми на Python можуть бути запущені на різних операційних системах без модифікацій, що робить його кросплатформеним інструментом.

Python був створений для того, щоб зробити програмування доступнішим і простішим, забезпечуючи при цьому достатню потужність для вирішення складних завдань. Основні сфери застосування Python включають кілька ключових областей.

Веб-розробка є однією з найбільш популярних сфер застосування Python. Завдяки фреймворкам, таким як Django та Flask, програмісти можуть створювати потужні та масштабовані веб-додатки. Django надає комплексне рішення для створення веб-додатків, включаючи автоматичне керування базами даних, маршрутизацію URL та підтримку форм. Flask, у свою чергу, є більш легким фреймворком, який дозволяє розробникам гнучко налаштовувати додатки під конкретні потреби.

Наукові обчислення та аналіз даних також є важливою сферою застосування Python. Завдяки бібліотекам, таким як NumPy, SciPy та pandas, Python забезпечує потужні інструменти для обробки та аналізу даних. NumPy забезпечує ефективну роботу з багатовимірними масивами і матрицями, SciPy надає чисельні методи для наукових обчислень, а pandas пропонує засоби для маніпуляції з даними та їх аналізу.

Машинне навчання і штучний інтелект є одними з найбільш швидкозростаючих областей застосування Python. Завдяки бібліотекам, таким як TensorFlow, Keras і Scikit-learn, Python став однією з основних мов програмування в цій галузі. TensorFlow

і Keras забезпечують потужні інструменти для побудови та тренування нейронних мереж, а Scikit-learn пропонує широкий спектр алгоритмів машинного навчання для класифікації, регресії та кластеризації.

Python широко використовується для автоматизації завдань та написання сценаріїв, що дозволяє спростити рутинні процеси. Завдяки своїй простоті та широкому набору бібліотек, Python є ідеальним інструментом для автоматизації різноманітних завдань, від обробки текстів до автоматизації веб-інтерфейсів.

Розробка ігор також є однією з можливостей Python. Існують бібліотеки, такі як Pygame, які дозволяють розробляти комп'ютерні ігри. Pygame надає інструменти для роботи з графікою, звуком і подіями, що дозволяє створювати інтерактивні ігрові додатки.

Python використовується для розробки програмного забезпечення для інших розумних девайсів завдяки своїй простоті і ефективності. Це дозволяє розробникам швидко створювати прототипи та впроваджувати інноваційні рішення для інтернету речей та інших сучасних технологій.

Таким чином, Python є універсальною мовою програмування, яка завдяки своїй простоті, потужності та широкому спектру застосувань здобула велику популярність серед розробників по всьому світу.

Python є однією з найпопулярніших мов програмування і використовується у багатьох відомих програмах і проектах. Ось кілька прикладів відомих програм та проектів, написаних на Python:

1. YouTube: платформа YouTube використовує Python для різних сервісів і обробки даних.
2. Google: Багато внутрішніх систем Google використовують Python, і ця мова активно використовується для розробки прототипів та скриптів у компанії.
3. Instagram: Популярна соціальна мережа Instagram використовує Django, веб-фреймворк на Python, для свого бекенду.

4. Spotify: Стрімінговий музичний сервіс Spotify використовує Python для аналізу даних і роботи з бекендом.
5. Dropbox: Хмарне сховище Dropbox використовує Python для більшості своїх серверних компонентів. Один із засновників Dropbox, Гвідо ван Россум, є творцем Python.
6. Reddit: Спочатку Reddit був написаний на Lisp, але згодом перейшов на Python, що дозволило збільшити продуктивність і швидкість розробки.
7. Quora: Веб-сайт питань і відповідей Quora також використовує Python для свого бекенду.
8. NASA: NASA використовує Python для наукових обчислень, аналізу даних і побудови моделей.
9. Pinterest: Соціальна мережа Pinterest використовує Python для обробки великої кількості даних і роботи з бекендом.
10. BitTorrent: Протокол обміну файлами BitTorrent був розроблений за допомогою Python.
11. OpenStack: Хмарна платформа з відкритим вихідним кодом OpenStack використовує Python для своїх основних компонентів.
12. Blender: Відкрите програмне забезпечення для 3D моделювання, анімації та рендерингу Blender має вбудовану підтримку Python для скриптів і автоматизації завдань.
13. Autodesk Maya: Потужний інструмент для 3D моделювання та анімації, Autodesk Maya, використовує Python для написання скриптів і автоматизації процесів.
14. Battlefield 2: Відеогра Battlefield 2 використовує Python для керування логікою гри і сценаріїв.
15. Ubuntu Software Center: Центр програмного забезпечення в операційній системі Ubuntu написаний на Python.

Сумісність Python із різними девайсами дуже широка завдяки його кросплатформеності та підтримці великої кількості бібліотек. Продукти розроблені на

Python можуть працювати на пристроях з операційними системами Windows, macOS та Linux, мобільних девайсах, мікроконтролерах, одноплатних комп'ютерах та у системах робототехніки.

2.2 Середовище розробки PyCharm

PyCharm [11] — це інтегроване середовище розробки (IDE) для мови програмування Python, розроблене компанією JetBrains. PyCharm відомий своєю потужністю та широкими можливостями, що роблять розробку на Python зручнішою і продуктивнішою (Рисунок 2.2).

Рисунок 2.2 - Логотип середовища розробки PyCharm

PyCharm був випущений у лютому 2010 року компанією JetBrains, яка спеціалізується на створенні інструментів для розробників. З моменту свого випуску PyCharm став одним із найпопулярніших IDE для Python завдяки своїм численним функціям і підтримці сучасних стандартів розробки.

PyCharm є потужним інструментом для розробки на Python, який забезпечує широкий спектр можливостей для підвищення продуктивності та ефективності роботи. Однією з ключових особливостей PyCharm є редактор коду. Він підтримує синтаксичне підсвічування, що допомагає краще орієнтуватися в коді. Інтелектуальне автодоповнення коду дозволяє швидко писати код, пропонуючи варіанти продовження фрази. Можливість навігації по коду забезпечує швидкий перехід до визначень класів,

методів і змінних, а функція "Go to definition" дозволяє зручно переглядати структуру коду.

Налагодження є ще однією важливою особливістю PyCharm. Він має потужний візуальний налагоджувач, який дозволяє встановлювати точки зупину, виконувати кроки коду і переглядати змінні в реальному часі. PyCharm також інтегрується з різними фреймворками для автоматизованого тестування, такими як pytest, unittest і nose, що значно спрощує процес виявлення і виправлення помилок.

PyCharm підтримує інтеграцію з системами контролю версій, такими як Git, SVN, Mercurial і Perforce, що дозволяє легко керувати версіями коду і співпрацювати з іншими розробниками. Інструменти для аналізу коду, такі як статичний аналіз коду і рефакторинг, допомагають перевіряти якість коду, виявляти потенційні помилки і дотримуватися стилю кодування.

Підтримка веб-розробки включає фреймворки, такі як Django, Flask і Pyramid. PyCharm також підтримує розробку на HTML, CSS та JavaScript, що важливо для повного циклу розробки веб-додатків. Інтеграція з базами даних дозволяє підключатися до різних баз даних (MySQL, PostgreSQL, Oracle, SQLite тощо) і виконувати SQL-запити безпосередньо з IDE.

Для наукових обчислень і аналізу даних PyCharm підтримує бібліотеки і фреймворки, такі як NumPy, SciPy, pandas і matplotlib. Це робить його популярним серед дослідників і аналітиків даних. Конфігурація середовища включає підтримку віртуальних середовищ Python, таких як virtualenv та conda, а також інтеграцію з Docker і Vagrant для створення контейнеризованих середовищ розробки.

PyCharm доступний у двох основних редакціях. PyCharm Community Edition є безкоштовною версією з відкритим вихідним кодом, яка надає основні функції для розробки на Python. PyCharm Professional Edition є платною версією, яка включає додаткові функції для веб-розробки, роботу з базами даних, наукові інструменти та інтеграцію з іншими мовами програмування.

2.3 Python модуль Tkinter

Tkinter [12] (від англійського "Tk interface") — це стандартна бібліотека Python для створення графічного інтерфейсу користувача (GUI). Tkinter є оболонкою над бібліотекою Tk GUI, яка спочатку була створена для мови програмування Tcl. Завдяки Tkinter, розробники можуть створювати вікна, кнопки, меню та інші елементи GUI за допомогою Python (Рисунок 2.3).

Рисунок 2.3 - Логотип модуля Tkinter

Спочатку бібліотека Tk була розроблена Джоном Оустером у 1988 році для мови програмування Tcl (Tool Command Language).

Tkinter: Перші версії Python отримали інтеграцію з Tk через модуль Tkinter ще на початку 1990-х років. Відтоді Tkinter став стандартною бібліотекою для створення GUI у Python і постачається разом з ним.

Tkinter є простим і інтуїтивно зрозумілим інструментом для створення GUI-додатків на Python. Він надає зрозумілий синтаксис, що робить його доступним навіть для новачків. Кросплатформенність Tkinter дозволяє розробникам створювати додатки, які працюють на різних операційних системах, таких як Windows, macOS та Linux, без необхідності змінювати код. Однією з ключових переваг Tkinter є широкий набір віджетів, що включає кнопки, мітки, текстові поля, меню, радіокнопки, прапорці, списки, шкали та інші елементи GUI. Це дозволяє створювати інтуїтивно зрозумілі та функціональні інтерфейси користувача.

Tkinter також відзначається гнучкістю, дозволяючи легко налаштовувати вигляд і поведінку віджетів через параметри та методи для управління їх властивостями.

Модель обробки подій, яку підтримує Tkinter, забезпечує зручну обробку дій користувачів, таких як натискання кнопок або введення тексту, за допомогою функцій зворотних викликів. Крім того, Tkinter легко розширюється завдяки можливості використовувати додаткові бібліотеки та модулі, що дозволяє розробляти більш складні додатки з багатим функціоналом.

Основним компонентом Tkinter є основне вікно, яке створюється при запуску програми. У цьому вікні розміщуються різноманітні віджети, такі як текстові та графічні мітки (Label), кнопки (Button), поля для введення тексту (Entry), багаторядкові текстові поля (Text), контейнери для організації інших віджетів (Frame), полотно для малювання графіки (Canvas), меню (Menu), списки елементів (Listbox), радіокнопки (Radiobutton), прапорці (Checkbutton) і шкали для вибору числових значень (Scale).

Для розташування віджетів у вікні Tkinter пропонує кілька методів, таких як pack(), grid() та place(). Ці методи дозволяють гнучко розміщувати елементи інтерфейсу відповідно до вимог дизайну додатка. Обробка подій у Tkinter реалізується за допомогою методу bind(), який дозволяє прив'язувати події до віджетів і визначати функції зворотних викликів, що виконуються у відповідь на дії користувачів.

Таким чином, Tkinter є потужним інструментом для створення графічних інтерфейсів користувача на Python, що забезпечує простоту використання, гнучкість, кросплатформенність і можливість розширення. Він підходить як для початківців, так і для досвідчених розробників, дозволяючи створювати як прості, так і складні додатки з багатим функціоналом.

2.4 Python модуль sqlite3

Модуль sqlite3 [13] є частиною стандартної бібліотеки Python і надає інтерфейс для роботи з базами даних SQLite з використанням мови програмування Python. SQLite є вбудованою та самостійною реляційною базою даних, яка не вимагає окремого сервера. Модуль sqlite3 дозволяє розробникам легко створювати, зчитувати,

оновлювати та видаляти дані в базі даних SQLite без необхідності встановлення додаткових програм або серверів баз даних.

Модуль sqlite3 був вперше включений до стандартної бібліотеки Python у версії 2.5 і надає Python доступ до функцій SQLite, що дозволяє програмістам створювати локальні бази даних для зберігання та управління даними. Однією з головних переваг модуля sqlite3 є його здатність легко підключатися до баз даних SQLite та виконувати різні операції з даними без необхідності встановлення окремих серверів баз даних. Це робить sqlite3 зручним і простим інструментом для роботи з базами даних у багатьох застосуваннях, де потрібне локальне зберігання даних.

SQLite є вбудованою базою даних, що означає, що вона не потребує окремого сервера баз даних. Всі дані зберігаються в одному файлі бази даних, що значно спрощує управління та розгортання додатків. Це особливо корисно для розробників, які працюють над невеликими або середніми проектами, де немає потреби у великій інфраструктурі для підтримки бази даних.

Модуль sqlite3 підтримує стандартні SQL-операції, такі як SELECT, INSERT, UPDATE, DELETE, CREATE, DROP та інші, що дозволяє розробникам використовувати знайомі інструменти та техніки для роботи з базами даних. Це робить перехід до використання sqlite3 безболісним для тих, хто вже має досвід роботи з SQL. Крім того, SQLite підтримує трансакційний механізм, який дозволяє виконувати групу операцій як одну атомарну операцію. Це забезпечує цілісність даних і дозволяє уникнути проблем, пов'язаних з частковим виконанням запитів.

Модуль sqlite3 дозволяє використовувати підготовлені запити, що дозволяє ефективно виконувати запити до бази даних з параметрами. Це не тільки підвищує продуктивність додатків, але й покращує безпеку, знижуючи ризик SQL-ін'єкцій. Підготовлені запити також полегшують повторне використання коду та його підтримку.

SQLite підтримує різноманітні вбудовані типи даних, такі як INTEGER, REAL, TEXT, BLOB, які можна використовувати для зберігання різних типів даних. Це забезпечує гнучкість у роботі з різноманітними даними і дозволяє розробникам

ефективно використовувати можливості бази даних для зберігання та обробки інформації.

Таким чином, модуль sqlite3 є потужним та зручним інструментом для роботи з локальними базами даних у Python. Він забезпечує легкий доступ до всіх основних функцій SQLite, дозволяючи розробникам ефективно зберігати, управляти та обробляти дані без необхідності в складних налаштуваннях та підтримці окремих серверів баз даних.

РОЗДІЛ 3. ПРАКТИЧНА ЧАСТИНА

3.1 Обґрунтування вибору мови програмування

Обираючи мову програмування, кожна людина враховує свої власні потреби, цілі та переваги, які надає та має конкретна мова програмування. Ось кілька причин, чому було обрано Python:

1. Простота вивчення: Python вважається однією з найбільш доступних мов програмування для початківців. Його зрозумілість та чистий синтаксис дозволяють швидко освоювати основи програмування.
2. Широкі можливості: Python використовується у різних галузях, від веб-розробки та наукових обчислень до штучного інтелекту та аналізу даних. Він має велику кількість бібліотек та фреймворків для вирішення різноманітних завдань.
3. Спільнота та підтримка: Python має велику та активну спільноту розробників, яка постійно доповнює його новими бібліотеками, інструментами та ресурсами для навчання. Це означає, що ви зможете знайти відповіді на багато питань та допомогу в інтернеті.
4. Переносність: Python підтримується на багатьох операційних системах, таких як Windows, macOS, Linux, що робить його ідеальним вибором для розробників, які працюють у різних середовищах.
5. Великі компанії використовують Python: Багато відомих компаній, таких як Google, Facebook, Netflix, Spotify та інші, використовують Python у своїх продуктах та сервісах. Він довів свою ефективність у великих проєктах та високонавантажених системах.

3.2 Обґрунтування вибору програмних засобів

Обираючи середовище розробки, кожна людина враховує свої потреби, зручність та можливості, які надає конкретне середовище. Ось кілька причин, чому було вирішено обрати PyCharm:

1. Потужність та функціонал: PyCharm є потужним інтегрованим середовищем розробки (IDE), яке надає розширені можливості для роботи з Python. Воно включає в себе широкий набір інструментів, таких як автодоповнення коду, візуальний налагоджувач, систему контролю версій та багато іншого, що полегшує процес розробки програм.
2. Підтримка веб-розробки: PyCharm має вбудовану підтримку для веб-розробки, що включає в себе підтримку популярних веб-фреймворків, таких як Django та Flask, а також редактор HTML, CSS і JavaScript.
3. Велика екосистема: PyCharm має активну спільноту користувачів і багато розширень та плагінів, які дозволяють розширити його функціонал для задоволення ваших потреб.
4. Підтримка іншомовних проектів: PyCharm має підтримку різних мов програмування та кодування, що дозволяє працювати з проектами на різних мовах, а не лише на Python.
5. Інтеграція з іншими інструментами: PyCharm інтегрується з різними інструментами розробки, такими як системи контролю версій (Git, Mercurial), засоби для роботи з базами даних і багато іншого, що полегшує процес розробки та спільну роботу в команді.

3.3 Опис програмної реалізації

Для створення програми обліку успішності студентів було використано Python, SQLite3 для бази даних та бібліотеку Tkinter для графічного інтерфейсу користувача. Усі розробки проводились у середовищі PyCharm.

Для розробки було побудовано блок-схему роботи програми (Рисунок 3.1).

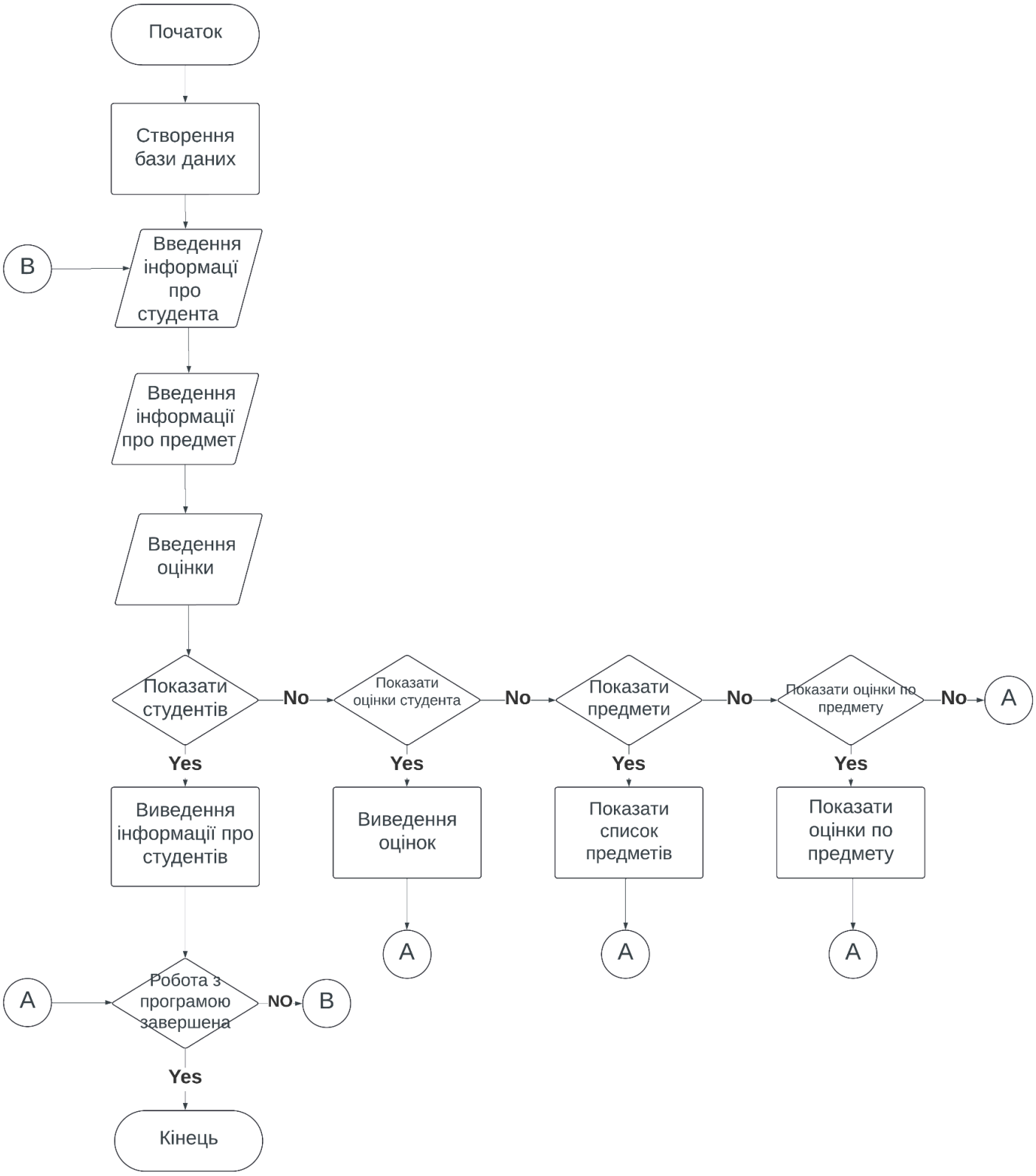

Рисунок 3.1 Блок-схема роботи програми

На етапі розробки перш за все було створено новий проєкт в середовищі PyCharm і підключено модулі Tkinter та Sqlite3. Код реалізації:

```
import sqlite3
import tkinter as tk
from tkinter import ttk
```

Було розроблено структуру бази даних і інтегровано в код програми:

```
def create_db():
    conn = sqlite3.connect('students.db')
    c = conn.cursor()
    c.execute('''
        CREATE TABLE IF NOT EXISTS students (
            student_id INTEGER PRIMARY KEY AUTOINCREMENT,
            first_name TEXT,
            last_name TEXT,
            email TEXT
        )
    ''')
    c.execute('''
        CREATE TABLE IF NOT EXISTS subjects (
            subject_id INTEGER PRIMARY KEY AUTOINCREMENT,
            subject_name TEXT
        )
    ''')
    c.execute('''
        CREATE TABLE IF NOT EXISTS grades (
            grade_id INTEGER PRIMARY KEY,
            student_id INTEGER,
```

```
        subject_id INTEGER,
        grade INTEGER,
        date TEXT,
        FOREIGN KEY(student_id) REFERENCES students(student_id),
        FOREIGN KEY(subject_id) REFERENCES subjects(subject_id)
    )
''')
conn.commit()
conn.close()
```

Для редагування таблиці було написано функції для додавання студента, оцінки, предмета, а також для зміни оцінки, та видалення студента, предмета. Ось деякі з них:

```
def add_student(first_name, last_name, email):
    conn = sqlite3.connect('students.db')
    c = conn.cursor()
    c.execute('''
        INSERT INTO students (first_name, last_name, email) VALUES (?, ?, ?)
    ''', (first_name, last_name, email))
    conn.commit()
    conn.close()
    def add_subject(subject_name):
    conn = sqlite3.connect('students.db')
    c = conn.cursor()
    c.execute('''
        INSERT INTO subjects (subject_name) VALUES (?)
    ''', (subject_name,))
    conn.commit()
    conn.close()
```

```
# Функція для додавання оцінки
def add_grade(student_id, subject_id, grade, date):
    conn = sqlite3.connect('students.db')
    c = conn.cursor()
    c.execute('''
        INSERT INTO grades (student_id, subject_id, grade, date) VALUES (?, ?, ?, ?)
    ''', (student_id, subject_id, grade, date))
    conn.commit()
    conn.close()
# Функція для оновлення оцінки
def update_grade(grade_id, student_id, subject_id, grade, date):
    conn = sqlite3.connect('students.db')
    c = conn.cursor()
    c.execute('''
        UPDATE grades
        SET student_id = ?, subject_id = ?, grade = ?, date = ?
        WHERE grade_id = ?
    ''', (student_id, subject_id, grade, date, grade_id))
    conn.commit()
    conn.close()
# Функція для видалення оцінки
def delete_grade(grade_id):
    conn = sqlite3.connect('students.db')
    c = conn.cursor()
    c.execute('''
        DELETE FROM grades WHERE grade_id = ?
    ''', (grade_id,))
    conn.commit()
```

conn.close()

Було створено головне вікно програми за допомогою Tkinter. Воно включає блоки форми для введення нових студентів та перегляду списку існуючих студентів, управління студентами, предметами і оцінками (Рисунок 3.1).

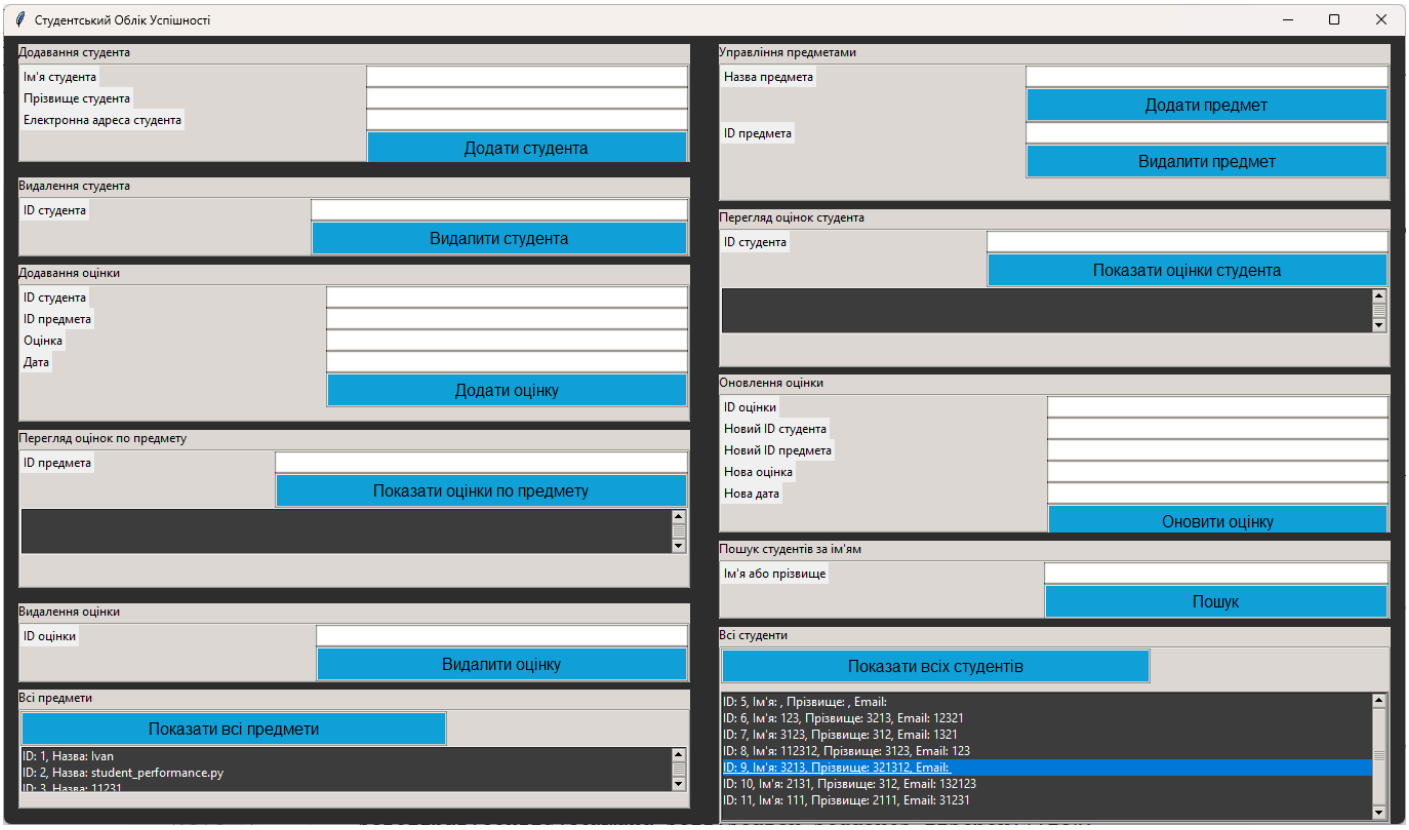

Рисунок 3.1 - Вікно розробленої програми

Код реалізації вікна програми:

```
class Application(Tk):
    def __init__(self):
        super().__init__()
        self.title("Студентський Облік Успішності")

        # Мінімальна роздільна здатність
```

```
        self.minsize(1366, 768)

        self.geometry("1366x768")  # Початкова роздільна здатність

        # Стилізація
        self.style = ttk.Style()
        self.style.theme_use("clam")

        self.style.configure("TLabel", font=("Helvetica", 12), background="#2E2E2E",
foreground="white")
        self.style.configure("TButton", font=("Helvetica", 12), background="#12a3d9",
foreground="black")
        self.style.configure("TEntry", font=("Helvetica", 12), background="#3E3E3E",
foreground="white")
        self.style.configure("TFrame", background="#2E2E2E")
        self.style.configure("TLabelFrame", background="#2E2E2E", font=("Helvetica",
14, "bold"))
        self.configure(bg="#2E2E2E")
```

Дані студентів та їх оцінок зберігаються в базі даних SQLite3 за допомогою SQL-запитів, виконаних через Python.

Додавання записів:

```
INSERT INTO students (first_name, last_name, email) VALUES (?, ?, ?);
INSERT INTO subjects (subject_name) VALUES (?);
INSERT INTO grades (student_id, subject_id, grade, date) VALUES (?, ?, ?, ?);
```

Оновлення записів:

```
UPDATE grades
```

```
SET student_id = ?, subject_id = ?, grade = ?, date = ?
WHERE grade_id = ?;
```

Видалення записів:

```
DELETE FROM grades WHERE grade_id = ?;
DELETE FROM students WHERE student_id = ?;
DELETE FROM subjects WHERE subject_id = ?;
DELETE FROM grades WHERE student_id = ?;
DELETE FROM grades WHERE subject_id = ?;
UPDATE students SET student_id = student_id - 1 WHERE student_id > ?;
UPDATE subjects SET subject_id = subject_id - 1 WHERE subject_id > ?;
UPDATE sqlite_sequence SET seq = seq - 1 WHERE name = 'subjects';
```

Отримання даних:

```
SELECT subjects.subject_name, grades.grade, grades.date
FROM grades
JOIN subjects ON grades.subject_id = subjects.subject_id
WHERE grades.student_id = ?;

SELECT students.first_name, students.last_name, grades.grade, grades.date
FROM grades
JOIN students ON grades.student_id = students.student_id
WHERE grades.subject_id = ?;

SELECT student_id, first_name, last_name, email
FROM students
WHERE first_name LIKE ? OR last_name LIKE ?;
```

```
SELECT student_id, first_name, last_name, email
FROM students;

SELECT subject_id, subject_name
FROM subjects;
```

Для отримання даних було реалізовано функції для передачі даних з бази та їх відображення у графічному інтерфейсі.

3.4 Тестування створеного продукту

В ході перевірки функціональності програмного продукту було забезпечено правильну обробку введених даних, проведено різноманітні сценарії для тестування реакції системи на різні ситуації та перевірено коректність та виведення інформації.

В рамках тестування перевірено функціональність на актуальних версіях операційних систем Windows, зокрема, Windows 10 та Windows 11, Linux, MacOS. Слід зазначити, що програмний продукт був націлений на користувачів персональних комп’ютерів, тому тестування на мобільних пристроях, зокрема телефонах, не передбачало та не гарантувало коректної роботи продукту.

Під час тестування інтерфейсу було оцінено зручність та естетика користувацького інтерфейсу, а також перевірено легкість навігації та взаємодії з елементами програмного забезпечення.

Тестування продуктивності включало вимірювання швидкодії та виявлення затримок в роботі.

Під час загальної перевірки придатності, аналізу відгуків користувачів, було зроблено висновок про його надійність.

У тестуванні на помилки ідентифіковано та виправлено виявлені недоліки, а також проведено повторне тестування.

ВИСНОВОК

У результаті виконання даної роботи було розроблено програмне забезпечення для обліку успішності студентів. Програма забезпечує автоматизацію процесу ведення обліку успішності, зручність введення даних.

В роботі описано такі пункти:

1. Постановка задачі та інформаційний огляд
2. Перегляд основ систем обліку успішності в навчальному процесі
3. Огляд подібних програмних продуктів
4. Позитивні аспекти оглянутих програмних продуктів
5. Негативні аспекти оглянутих програмних продуктів
6. Мова програмування Python
7. Середовище розробки PyCharm
8. Python модуль Tkinter
9. Python модуль SQLite3
10. Обґрунтування вибору мови програмування
11. Обґрунтування вибору програмних засобів
12. Опис програмної реалізації
13. Тестування створеного продукту

Для реалізації програми було використано Python, SQLite3 для управління базою даних та бібліотеку Tkinter для створення графічного інтерфейсу користувача. Усі розробки проводились у середовищі PyCharm, що забезпечило зручність та ефективність роботи над проектом. Було розроблено функції для створення, редагування та видалення записів у базі даних, а також для відображення цих даних у графічному інтерфейсі.

Тестування програми підтвердило її функціональність та надійність. Було проведено перевірку на різних операційних системах (Windows, Linux, MacOS), що підтвердило її переносимість та коректну роботу на актуальних версіях цих систем.

Інтерфейс програми був оцінений з точки зору зручності користування та естетики, що забезпечило позитивний користувацький досвід. Тестування продуктивності не виявило суттєвих затримок у роботі програми, а тестування на помилки дозволило ідентифікувати та виправити виявлені недоліки, підвищивши надійність програмного продукту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Гаркуша С. В. Методичні рекомендації щодо виконання кваліфікаційної роботи студентів спеціальності 122 Комп’ютерні науки освітня програма «Комп’ютерні науки» ступеня магістра / С. В. Гаркуша, О. В. Ольховська, О. О. Черненко. – Полтава: ПУЕТ, 2023. – 681 с.
2. Електронний журнал обліку успішності студентів як ефективний інструмент моніторингу якості вищої освіти. Титова, Наталія Михайлівна. Київ: НПУ ім. М.П. Драгоманова, 2017. URL: enpuir.npu.edu.ua/handle/123456789/12973.
3. NZ.UA. "Електронний щоденник для шкіл." URL: https://nz.ua.
4. Moodle. "Moodle: Open-source learning platform." URL: https://moodle.org.
5. Blackboard Inc. "Blackboard: Education Technology & Services." URL: https://www.anthology.com/products/teaching-and-learning/learning-effectiveness/blackboard-learn.
6. Google. "Google Classroom: Online Learning Platform." URL: https://classroom.google.com.
7. Instructure. "Canvas: Learning Management System." URL: https://www.instructure.com/canvas
8. Python Software Foundation. "Python 3 Documentation." URL: https://docs.python.org/3/.
9. Elhalid, Osama Burak and Alm Alhelal, Zaynelabdin and HASSAN, SAMER, Exploring the Fundamentals of Python Programming: A Comprehensive Guide for Beginners. October 25, 2023. URL: http://dx.doi.org/10.2139/ssrn.4612765
10. Python, Simon Yuill та Harry Halpin. URL: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=1f2ee3831eebfc97bfafd514ca2abb7e2c5c86bb.
11. JetBrains. "Getting Started with PyCharm." URL: https://www.jetbrains.com/help/pycharm/getting-started.html.
12. Python Software Foundation. "Tkinter — Python Interface to Tcl/Tk. URL: https://docs.python.org/3/library/tkinter.html.
13. Python Software Foundation. "Sqlite3 — DB-API 2.0 interface for SQLite databases" URL: https://docs.python.org/3/library/sqlite3.html

ДОДАТОК А. ВИХІДНІ КОДИ

```
import sqlite3
from tkinter import Tk, Label, Entry, Button, Listbox, END, StringVar
from tkinter import ttk

# Створення бази даних і таблиць
def create_db():
    conn = sqlite3.connect('students.db')
    c = conn.cursor()
    c.execute('''
        CREATE TABLE IF NOT EXISTS students (
            student_id INTEGER PRIMARY KEY AUTOINCREMENT,
            first_name TEXT,
            last_name TEXT,
            email TEXT
        )
    ''')
    c.execute('''
        CREATE TABLE IF NOT EXISTS subjects (
            subject_id INTEGER PRIMARY KEY AUTOINCREMENT,
            subject_name TEXT
        )
    ''')
    c.execute('''
        CREATE TABLE IF NOT EXISTS grades (
            grade_id INTEGER PRIMARY KEY,
            student_id INTEGER,
            subject_id INTEGER,
            grade INTEGER,
            date TEXT,
            FOREIGN KEY(student_id) REFERENCES students(student_id),
            FOREIGN KEY(subject_id) REFERENCES subjects(subject_id)
        )
    ''')
    conn.commit()
```

```
    conn.close()

# Функція для додавання студента
def add_student(first_name, last_name, email):
    conn = sqlite3.connect('students.db')
    c = conn.cursor()
    c.execute('''
        INSERT INTO students (first_name, last_name, email) VALUES (?, ?, ?)
    ''', (first_name, last_name, email))
    conn.commit()
    conn.close()

# Функція для додавання предмета
def add_subject(subject_name):
    conn = sqlite3.connect('students.db')
    c = conn.cursor()
    c.execute('''
        INSERT INTO subjects (subject_name) VALUES (?)
    ''', (subject_name,))
    conn.commit()
    conn.close()

# Функція для додавання оцінки
def add_grade(student_id, subject_id, grade, date):
    conn = sqlite3.connect('students.db')
    c = conn.cursor()
    c.execute('''
        INSERT INTO grades (student_id, subject_id, grade, date) VALUES (?, ?, ?, ?)
    ''', (student_id, subject_id, grade, date))
    conn.commit()
    conn.close()

# Функція для оновлення оцінки
def update_grade(grade_id, student_id, subject_id, grade, date):
    conn = sqlite3.connect('students.db')
    c = conn.cursor()
```

```
    c.execute('''
        UPDATE grades
        SET student_id = ?, subject_id = ?, grade = ?, date = ?
        WHERE grade_id = ?
    ''', (student_id, subject_id, grade, date, grade_id))
    conn.commit()
    conn.close()

# Функція для видалення оцінки
def delete_grade(grade_id):
    conn = sqlite3.connect('students.db')
    c = conn.cursor()
    c.execute('''
        DELETE FROM grades WHERE grade_id = ?
    ''', (grade_id,))
    conn.commit()
    conn.close()

# Функція для видалення студента
def delete_student(student_id):
    student_id = int(student_id)  # Переконайтеся, що student_id є цілим числом
    conn = sqlite3.connect('students.db')
    c = conn.cursor()

    # Починаємо транзакцію вручну, щоб уникнути блокування бази даних
    c.execute('BEGIN TRANSACTION')

    try:
        c.execute('''
            DELETE FROM students WHERE student_id = ?
        ''', (student_id,))
        c.execute('''
            DELETE FROM grades WHERE student_id = ?
        ''', (student_id,))
        c.execute('''
            UPDATE students SET student_id = student_id - 1 WHERE student_id > ?
```

```
        ''', (student_id,))
        conn.commit()
    except sqlite3.Error as e:
        print(f"An error occurred: {e}")
        conn.rollback()
    finally:
        conn.close()

# Функція для видалення предмета
def delete_subject(subject_id):
    conn = sqlite3.connect('students.db')
    c = conn.cursor()
    c.execute('''
        DELETE FROM subjects WHERE subject_id = ?
    ''', (subject_id,))
    c.execute('''
        DELETE FROM grades WHERE subject_id = ?
    ''', (subject_id,))
    c.execute('''
        UPDATE subjects SET subject_id = subject_id - 1 WHERE subject_id > ?
    ''', (subject_id,))
    c.execute('''
        UPDATE sqlite_sequence SET seq = seq - 1 WHERE name = 'subjects'
    ''')
    conn.commit()
    conn.close()

# Функція для отримання оцінок студента
def get_student_grades(student_id):
    conn = sqlite3.connect('students.db')
    c = conn.cursor()
    c.execute('''
        SELECT grades.grade_id, subjects.subject_name, grades.grade, grades.date
        FROM grades
        JOIN subjects ON grades.subject_id = subjects.subject_id
        WHERE grades.student_id = ?
```

```
    ''', (student_id,))
    results = c.fetchall()
    conn.close()
    return results

# Функція для отримання оцінок по предмету
def get_subject_grades(subject_id):
    conn = sqlite3.connect('students.db')
    c = conn.cursor()
    c.execute('''
        SELECT students.first_name, students.last_name, grades.grade, grades.date
        FROM grades
        JOIN students ON grades.student_id = students.student_id
        WHERE grades.subject_id = ?
    ''', (subject_id,))
    results = c.fetchall()
    conn.close()
    return results

# Функція для пошуку студентів за ім'ям
def search_students(name):
    conn = sqlite3.connect('students.db')
    c = conn.cursor()
    c.execute('''
        SELECT student_id, first_name, last_name, email
        FROM students
        WHERE first_name LIKE ? OR last_name LIKE ?
    ''', ('%' + name + '%', '%' + name + '%'))
    results = c.fetchall()
    conn.close()
    return results

# Функція для отримання всіх студентів
def get_all_students():
    conn = sqlite3.connect('students.db')
    c = conn.cursor()
```

```
    c.execute('''
        SELECT student_id, first_name, last_name, email
        FROM students
    ''')
    results = c.fetchall()
    conn.close()
    return results

# Функція для отримання всіх предметів
def get_all_subjects():
    conn = sqlite3.connect('students.db')
    c = conn.cursor()
    c.execute('''
        SELECT subject_id, subject_name
        FROM subjects
    ''')
    results = c.fetchall()
    conn.close()
    return results

# Графічний інтерфейс користувача
class Application(Tk):
    def __init__(self):
        super().__init__()
        self.title("Студентський Облік Успішності")

        # Мінімальна роздільна здатність
        self.minsize(1366, 768)

        self.geometry("1366x768")  # Початкова роздільна здатність

        # Стилізація
        self.style = ttk.Style()
        self.style.theme_use("clam")
```

```
        self.style.configure("TLabel", font=("Helvetica", 12), background="#2E2E2E",
foreground="white")
        self.style.configure("TButton", font=("Helvetica", 12), background="#12a3d9",
foreground="black")
        self.style.configure("TEntry", font=("Helvetica", 12), background="#3E3E3E",
foreground="white")
        self.style.configure("TFrame", background="#2E2E2E")
        self.style.configure("TLabelFrame", background="#2E2E2E", font=("Helvetica", 14,
"bold"))
        self.configure(bg="#2E2E2E")

        # Додавання студента
        self.add_student_frame = ttk.LabelFrame(self, text="Додавання студента")
        self.add_student_frame.place(relx=0.01, rely=0.01, relwidth=0.48, relheight=0.15)

        Label(self.add_student_frame, text="Ім'я студента").grid(row=0, column=0,
sticky="w")
        self.first_name = ttk.Entry(self.add_student_frame, foreground="black")
        self.first_name.grid(row=0, column=1, sticky="ew")

        Label(self.add_student_frame, text="Прізвище студента").grid(row=1, column=0,
sticky="w")
        self.last_name = ttk.Entry(self.add_student_frame, foreground="black")
        self.last_name.grid(row=1, column=1, sticky="ew")

        Label(self.add_student_frame, text="Електронна адреса студента").grid(row=2,
column=0, sticky="w")
        self.email = ttk.Entry(self.add_student_frame, foreground="black")
        self.email.grid(row=2, column=1, sticky="ew")

        ttk.Button(self.add_student_frame, text="Додати студента",
command=self.add_student_gui).grid(row=3, column=1, sticky="ew")

        # Управління предметами
        self.manage_subject_frame = ttk.LabelFrame(self, text="Управління предметами")
        self.manage_subject_frame.place(relx=0.51, rely=0.01, relwidth=0.48, relheight=0.20)
```

```
        Label(self.manage_subject_frame, text="Назва предмета").grid(row=0, column=0,
sticky="w")
        self.subject_name = ttk.Entry(self.manage_subject_frame, foreground="black")
        self.subject_name.grid(row=0, column=1, sticky="ew")

        ttk.Button(self.manage_subject_frame, text="Додати предмет",
command=self.add_subject_gui).grid(row=1, column=1, sticky="ew")

        Label(self.manage_subject_frame, text="ID предмета").grid(row=2, column=0,
sticky="w")
        self.delete_subject_id = ttk.Entry(self.manage_subject_frame, foreground="black")
        self.delete_subject_id.grid(row=2, column=1, sticky="ew")

        ttk.Button(self.manage_subject_frame, text="Видалити предмет",
command=self.delete_subject_gui).grid(row=3, column=1, sticky="ew")

        #self.subject_listbox = Listbox(self.manage_subject_frame, bg="#3E3E3E",
fg="white")
        #self.subject_listbox.grid(row=4, columnspan=2, sticky="ew")
        #scrollbar = ttk.Scrollbar(self.subject_listbox, orient="vertical")
        #scrollbar.config(command=self.subject_listbox.yview)
        #scrollbar.pack(side="right", fill="y")

        # Видалення студента
        self.delete_student_frame = ttk.LabelFrame(self, text="Видалення студента")
        self.delete_student_frame.place(relx=0.01, rely=0.18, relwidth=0.48, relheight=0.1)

        Label(self.delete_student_frame, text="ID студента").grid(row=0, column=0,
sticky="w")
        self.delete_student_id = ttk.Entry(self.delete_student_frame, foreground="black")
        self.delete_student_id.grid(row=0, column=1, sticky="ew")

        ttk.Button(self.delete_student_frame, text="Видалити студента",
command=self.delete_student_gui).grid(row=1, column=1, sticky="ew")
```

```
        # Додавання оцінки
        self.add_grade_frame = ttk.LabelFrame(self, text="Додавання оцінки")
        self.add_grade_frame.place(relx=0.01, rely=0.29, relwidth=0.48, relheight=0.2)

        Label(self.add_grade_frame, text="ID студента").grid(row=0, column=0, sticky="w")
        self.student_id = ttk.Entry(self.add_grade_frame, foreground="black")
        self.student_id.grid(row=0, column=1, sticky="ew")

        Label(self.add_grade_frame, text="ID предмета").grid(row=1, column=0, sticky="w")
        self.subject_id = ttk.Entry(self.add_grade_frame, foreground="black")
        self.subject_id.grid(row=1, column=1, sticky="ew")

        Label(self.add_grade_frame, text="Оцінка").grid(row=2, column=0, sticky="w")
        self.grade = ttk.Entry(self.add_grade_frame, foreground="black")
        self.grade.grid(row=2, column=1, sticky="ew")

        Label(self.add_grade_frame, text="Дата").grid(row=3, column=0, sticky="w")
        self.date = ttk.Entry(self.add_grade_frame, foreground="black")
        self.date.grid(row=3, column=1, sticky="ew")

        ttk.Button(self.add_grade_frame, text="Додати оцінку",
command=self.add_grade_gui).grid(row=4, column=1, sticky="ew")

        # Перегляд оцінок студента
        self.view_student_grades_frame = ttk.LabelFrame(self, text="Перегляд оцінок
студента")
        self.view_student_grades_frame.place(relx=0.51, rely=0.22, relwidth=0.48,
relheight=0.2)

        Label(self.view_student_grades_frame, text="ID студента").grid(row=0, column=0,
sticky="w")
        self.view_student_id = ttk.Entry(self.view_student_grades_frame, foreground="black")
        self.view_student_id.grid(row=0, column=1, sticky="ew")

        ttk.Button(self.view_student_grades_frame, text="Показати оцінки студента",
command=self.show_student_grades).grid(row=1, column=1, sticky="ew")
```

```
        self.student_grades_listbox = Listbox(self.view_student_grades_frame, bg="#3E3E3E",
fg="white")
        self.student_grades_listbox.grid(row=2, columnspan=2, sticky="ew")
        scrollbar = ttk.Scrollbar(self.student_grades_listbox, orient="vertical")
        scrollbar.config(command=self.student_grades_listbox.yview)
        scrollbar.pack(side="right", fill="y")

        # Перегляд оцінок по предмету
        self.view_subject_grades_frame = ttk.LabelFrame(self, text="Перегляд оцінок по
предмету")
        self.view_subject_grades_frame.place(relx=0.01, rely=0.5, relwidth=0.48,
relheight=0.2)

        Label(self.view_subject_grades_frame, text="ID предмета").grid(row=0, column=0,
sticky="w")
        self.view_subject_id = ttk.Entry(self.view_subject_grades_frame, foreground="black")
        self.view_subject_id.grid(row=0, column=1, sticky="ew")

        ttk.Button(self.view_subject_grades_frame, text="Показати оцінки по предмету",
command=self.show_subject_grades).grid(row=1, column=1, sticky="ew")

        self.subject_grades_listbox = Listbox(self.view_subject_grades_frame, bg="#3E3E3E",
fg="white")
        self.subject_grades_listbox.grid(row=2, columnspan=2, sticky="ew")
        scrollbar = ttk.Scrollbar(self.subject_grades_listbox, orient="vertical")
        scrollbar.config(command=self.subject_grades_listbox.yview)
        scrollbar.pack(side="right", fill="y")

        # Оновлення оцінки
        self.update_grade_frame = ttk.LabelFrame(self, text="Оновлення оцінки")
        self.update_grade_frame.place(relx=0.51, rely=0.43, relwidth=0.48, relheight=0.2)

        Label(self.update_grade_frame, text="ID оцінки").grid(row=0, column=0, sticky="w")
        self.update_grade_id = ttk.Entry(self.update_grade_frame, foreground="black")
        self.update_grade_id.grid(row=0, column=1, sticky="ew")
```

```
        Label(self.update_grade_frame, text="Новий ID студента").grid(row=1, column=0,
sticky="w")
        self.update_student_id = ttk.Entry(self.update_grade_frame, foreground="black")
        self.update_student_id.grid(row=1, column=1, sticky="ew")

        Label(self.update_grade_frame, text="Новий ID предмета").grid(row=2, column=0,
sticky="w")
        self.update_subject_id = ttk.Entry(self.update_grade_frame, foreground="black")
        self.update_subject_id.grid(row=2, column=1, sticky="ew")

        Label(self.update_grade_frame, text="Нова оцінка").grid(row=3, column=0,
sticky="w")
        self.update_grade = ttk.Entry(self.update_grade_frame, foreground="black")
        self.update_grade.grid(row=3, column=1, sticky="ew")

        Label(self.update_grade_frame, text="Нова дата").grid(row=4, column=0, sticky="w")
        self.update_date = ttk.Entry(self.update_grade_frame, foreground="black")
        self.update_date.grid(row=4, column=1, sticky="ew")

        ttk.Button(self.update_grade_frame, text="Оновити оцінку",
command=self.update_grade_gui).grid(row=5, column=1, sticky="ew")

        # Видалення оцінки
        self.delete_grade_frame = ttk.LabelFrame(self, text="Видалення оцінки")
        self.delete_grade_frame.place(relx=0.01, rely=0.72, relwidth=0.48, relheight=0.1)

        Label(self.delete_grade_frame, text="ID оцінки").grid(row=0, column=0, sticky="w")
        self.delete_grade_id = ttk.Entry(self.delete_grade_frame, foreground="black")
        self.delete_grade_id.grid(row=0, column=1, sticky="ew")

        ttk.Button(self.delete_grade_frame, text="Видалити оцінку",
command=self.delete_grade_gui).grid(row=1, column=1, sticky="ew")

        # Пошук студентів за ім'ям
        self.search_students_frame = ttk.LabelFrame(self, text="Пошук студентів за ім'ям")
```

```
        self.search_students_frame.place(relx=0.51, rely=0.64, relwidth=0.48, relheight=0.1)

        Label(self.search_students_frame, text="Ім'я або прізвище").grid(row=0, column=0,
sticky="w")
        self.search_student_name = ttk.Entry(self.search_students_frame, foreground="black")
        self.search_student_name.grid(row=0, column=1, sticky="ew")

        ttk.Button(self.search_students_frame, text="Пошук",
command=self.search_students_gui).grid(row=1, column=1, sticky="ew")

        self.search_students_listbox = Listbox(self.search_students_frame, bg="#3E3E3E",
fg="white")
        self.search_students_listbox.grid(row=2, columnspan=2, sticky="ew")
        scrollbar = ttk.Scrollbar(self.search_students_listbox, orient="vertical")
        scrollbar.config(command=self.search_students_listbox.yview)
        scrollbar.pack(side="right", fill="y")

        # Відображення всіх студентів
        self.view_all_students_frame = ttk.LabelFrame(self, text="Всі студенти")
        self.view_all_students_frame.place(relx=0.51, rely=0.75, relwidth=0.48, relheight=0.25)

        ttk.Button(self.view_all_students_frame, text="Показати всіх студентів",
command=self.show_all_students).grid(row=0, column=0, sticky="ew")

        self.all_students_listbox = Listbox(self.view_all_students_frame, bg="#3E3E3E",
fg="white")
        self.all_students_listbox.grid(row=1, columnspan=2, sticky="ew")
        scrollbar = ttk.Scrollbar(self.all_students_listbox, orient="vertical")
        scrollbar.config(command=self.all_students_listbox.yview)
        scrollbar.pack(side="right", fill="y")
        # Налаштування параметрів розміщення лістбоксу
        self.all_students_listbox.place(relx=0, rely=0.25, relwidth=1, relheight=0.75)
        # Відображення всіх предметів
        self.view_all_subjects_frame = ttk.LabelFrame(self, text="Всі предмети")
        self.view_all_subjects_frame.place(relx=0.01, rely=0.83, relwidth=0.48, relheight=0.15)
```

```
        ttk.Button(self.view_all_subjects_frame, text="Показати всі предмети",
command=self.show_all_subjects).grid(row=0, column=0, sticky="ew")

        self.all_subjects_listbox = Listbox(self.view_all_subjects_frame, bg="#3E3E3E",
fg="white")
        self.all_subjects_listbox.grid(row=1, columnspan=2, sticky="ew")
        scrollbar = ttk.Scrollbar(self.all_subjects_listbox, orient="vertical")
        scrollbar.config(command=self.all_subjects_listbox.yview)
        scrollbar.pack(side="right", fill="y")

        # Пропорційне масштабування колонок і рядків
        for i in range(2):
            self.add_student_frame.grid_columnconfigure(i, weight=1)
            self.manage_subject_frame.grid_columnconfigure(i, weight=1)
            self.add_grade_frame.grid_columnconfigure(i, weight=1)
            self.view_student_grades_frame.grid_columnconfigure(i, weight=1)
            self.view_subject_grades_frame.grid_columnconfigure(i, weight=1)
            self.update_grade_frame.grid_columnconfigure(i, weight=1)
            self.delete_grade_frame.grid_columnconfigure(i, weight=1)
            self.search_students_frame.grid_columnconfigure(i, weight=1)
            self.view_all_students_frame.grid_columnconfigure(i, weight=1)
            self.view_all_subjects_frame.grid_columnconfigure(i, weight=1)
            self.delete_student_frame.grid_columnconfigure(i, weight=1)

        self.show_all_subjects()  # Показуємо всі предмети після створення інтерфейсу

    def add_student_gui(self):
        add_student(self.first_name.get(), self.last_name.get(), self.email.get())
        self.first_name.delete(0, END)
        self.last_name.delete(0, END)
        self.email.delete(0, END)
        self.show_all_students()

    def add_subject_gui(self):
        add_subject(self.subject_name.get())
        self.subject_name.delete(0, END)
```

```
        self.show_all_subjects()

    def add_grade_gui(self):
        add_grade(self.student_id.get(), self.subject_id.get(), self.grade.get(), self.date.get())
        self.student_id.delete(0, END)
        self.subject_id.delete(0, END)
        self.grade.delete(0, END)
        self.date.delete(0, END)

    def show_student_grades(self):
        student_id = self.view_student_id.get()
        grades = get_student_grades(student_id)
        self.student_grades_listbox.delete(0, END)
        for grade in grades:
            self.student_grades_listbox.insert(END, f"ID оцінки: {grade[0]}, Предмет:
{grade[1]}, Оцінка: {grade[2]}, Дата: {grade[3]}")

    def show_subject_grades(self):
        subject_id = self.view_subject_id.get()
        grades = get_subject_grades(subject_id)
        self.subject_grades_listbox.delete(0, END)
        for grade in grades:
            self.subject_grades_listbox.insert(END, f"Студент: {grade[0]} {grade[1]}, Оцінка:
{grade[2]}, Дата: {grade[3]}")

    def update_grade_gui(self):
        update_grade(self.update_grade_id.get(), self.update_student_id.get(),
self.update_subject_id.get(), self.update_grade.get(), self.update_date.get())
        self.update_grade_id.delete(0, END)
        self.update_student_id.delete(0, END)
        self.update_subject_id.delete(0, END)
        self.update_grade.delete(0, END)
        self.update_date.delete(0, END)

    def delete_grade_gui(self):
        delete_grade(self.delete_grade_id.get())
```

```
        self.delete_grade_id.delete(0, END)

    def search_students_gui(self):
        name = self.search_student_name.get()
        students = search_students(name)
        self.search_students_listbox.delete(0, END)
        for student in students:
            self.search_students_listbox.insert(END, f"ID: {student[0]}, Ім'я: {student[1]},
Прізвище: {student[2]}, Email: {student[3]}")

    def show_all_students(self):
        students = get_all_students()
        self.all_students_listbox.delete(0, END)
        for student in students:
            self.all_students_listbox.insert(END, f"ID: {student[0]}, Ім'я: {student[1]},
Прізвище: {student[2]}, Email: {student[3]}")

    def show_all_subjects(self):
        subjects = get_all_subjects()
        self.all_subjects_listbox.delete(0, END)
        for subject in subjects:
            self.all_subjects_listbox.insert(END, f"ID: {subject[0]}, Назва: {subject[1]}")

    def delete_student_gui(self):
        student_id = self.delete_student_id.get()
        if student_id:
            try:
                delete_student(student_id)
                self.delete_student_id.delete(0, END)
                self.show_all_students()  # Оновлюємо список всіх студентів після видалення
            except Exception as e:
                print(f"An error occurred: {e}")

    def delete_subject_gui(self):
        delete_subject(self.delete_subject_id.get())
        self.delete_subject_id.delete(0, END)
```

```
        self.show_all_subjects()  # Оновлюємо список всіх предметів після видалення

if __name__ == "__main__":
    create_db()
    app = Application()
    app.mainloop()
```