ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка застосунку для підтримки самостійної роботи здобувача вищої освіти з використанням мови програмування Python»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Олег ЛЕВЧИК
(підпис)

Виконав: здобувач вищої освіти групи ПД-43

Олег ЛЕВЧИК

Керівник: Ігор АВЕРІЧЕВ
к.е.н.

Рецензент: _____

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
«_____» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Левчик Олегу Ігоровичу

1. Тема кваліфікаційної роботи: «Розробка застосунку для підтримки самостійної роботи здобувача вищої освіти з використанням мови програмування Python» керівник кваліфікаційної роботи к.е.н., доцент кафедри ІПЗ Ігор АВЕРІЧЕВ, затверджені наказом Державного університету інформаційно-комунікаційних технологій від «27» лютого 2024 р. №145.

2. Строк подання кваліфікаційної роботи «28» травня 2024 р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Аналіз предметної галузі.
2. Проектування програмного застосунку.
3. Розробка програмного застосунку.
4. Тестування програмного застосунку.

5. Перелік графічного матеріалу: *презентація*

1. Діаграма варіантів використання.
2. Діаграма послідовності.
3. Діаграма класів.

6. Дата видачі завдання«28» лютого 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	28.02-06.03.2024	
2	Аналіз та дослідження існуючих аналогів	07.03-13.03.2024	
3	Вивчення матеріалів для аналізу існуючих рішень самостійної роботи здобувача	15.03-31.03.2024	
4	Аналіз переваг та недоліків існуючих програмних засобів	01.04-10.04.24	
5	Дослідження функціональних та нефункціональних вимог до застосунку	11.04-04.05.24	
6	Проєктування інтерфейсу користувача	05.05-10.05.24	
7	Оформлення роботи: вступ, висновки, реферат	29.04-05.05.24	
8	Розробка демонстраційних матеріалів	06.05-12.05.24	
9	Попередній захист роботи	13.05-31.05.2024	

Здобувач вищої освіти _____ Олег ЛЕВЧИК
(підпис)

Керівник
кваліфікаційної роботи _____ Ігор АВЕРІЧЕВ
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 60 стор., 3 табл., 36 рис., 36 джерел.

Мета роботи – підтримка самостійної роботи здобувача вищої освіти за використання WEB-застосунку.

Об'єкт дослідження – самостійна робота здобувача вищої освіти.

Предмет дослідження – застосунок для підтримки самостійної роботи здобувача вищої освіти.

Короткий зміст роботи: У роботі проведено аналіз існуючих рішень для самостійної роботи здобувача та спроектовано архітектуру програмного забезпечення. Досліджено переваги та недоліки існуючих програмних засобів. Розроблено користувацький інтерфейс для взаємодії здобувачів з програмним забезпеченням. В результаті розроблено застосунок для підтримки самостійної роботи здобувачів вищої освіти.

КЛЮЧОВІ СЛОВА: КОРИСТУВАЦЬКИЙ ІНТЕРФЕЙС, ДІАГРАМИ КЛАСІВ, БАЗА ДАНИХ, САМОСТІЙНА РОБОТА СТУДЕНТІВ.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД	-	База Даних
AMQP	-	Advanced Message Queuing Protocol
REST	-	Representational State Transfer
UI	-	Інтерфейс користувача
NUI	-	Природний інтерфейс користувача
GUI	-	Графічний інтерфейс користувача
VDI	-	Мережеве налаштування
TUI	-	Відчутний інтерфейс користувача

ВСТУП

В сучасному освітньому середовищі самостійна робота студентів та аспірантів вищих навчальних закладів набуває все більшої уваги. Вона дозволяє здобувачам вищої освіти розвивати свої навички, самостійно вирішувати завдання та відстежувати свій прогрес у навчанні. Однак, для ефективного ведення самостійної роботи здобувачам потрібні зручні та функціональні інструменти, які б допомагали організувати навчальний процес та вести облік завдань. Це підтверджує актуальність розробки застосунку для підтримки самостійної роботи здобувача вищої освіти.

Об’єкт дослідження – самостійна робота здобувача вищої освіти.

Предмет дослідження – застосунок для підтримки самостійної роботи здобувача вищої освіти.

Мета роботи – підтримка самостійної роботи здобувача вищої освіти за використання WEB-застосунку.

Для досягнення поставленої мети в роботі було вирішено наступні завдання:

1. Проаналізувати потреби студентів у підтримці самостійної роботи.
2. Визначити переваги та недоліки існуючих програмних засобів.
3. Розробити функціональні та нефункціональні вимоги до застосунку для самостійної роботи здобувача.
4. Спроектувати архітектуру програмного забезпечення, визначити класи та методи для створення застосунку.
5. Розробити користувацький інтерфейс для взаємодії студентів з програмним забезпеченням.
6. Розробити застосунок для підтримки самостійної роботи здобувачів вищої освіти.
7. Провести модульне та інтеграційне тестування програмного забезпечення.

Розробленний застосунок має практичну новизну завдяки наявності функції сповіщення про дедлайни виконання самостійних робіт.

Робота пройшла апробацію на науково-технічних конференціях «Сучасний стан та перспективи розвитку IoT» (18 квітня 2024 р., Київ, Державний університет інформаційно-комунікаційних технологій), «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях» (24 квітня 2024 р., Київ, Державний університет інформаційно-комунікаційних технологій), за результатами апробації опубліковано тези доповідей [1, 2].

1 ОРГАНІЗАЦІЯ САМОСТІЙНОЇ РОБОТИ СТУДЕНТІВ

1.1 Значення самостійної роботи у вищій освіті

Самостійна робота вищої освіти є ключовим елементом в процесі навчання, особливо в контексті сучасних вимог до підготовки студентів. Вона відображається в різних формах: виконання індивідуальних завдань, дослідницької роботи, написання курсових та дипломних робіт тощо.

Організація самостійної роботи студентів має кілька аспектів, які слід враховувати. По-перше, це розвиток самоорганізації та самодисципліни у студентів. Саме через самостійну роботу вони навчаються ефективно планувати свій час, визначати пріоритети та виконувати поставлені завдання в обумовлений строк.

По-друге, самостійна робота сприяє глибшому засвоєнню навчального матеріалу. Коли студент самостійно працює над певною темою або завданням, він має можливість уважно проаналізувати матеріал, дослідити додаткові джерела і зробити власні висновки. Це сприяє кращому розумінню предмету та його практичному застосуванню.

По-третє, самостійна робота допомагає розвивати критичне мислення та творчість у студентів. Під час самостійної роботи вони змушені аналізувати інформацію, формулювати власні думки, аргументувати свої погляди та шукати нестандартні рішення.

У контексті теми дипломної роботи про розробку застосунку для підтримки самостійної роботи студентів у вищій освіті, важливо розглянути, як цей застосунок може сприяти здобуттю вищої освіти більш ефективним та продуктивним шляхом. Це включає в себе розробку інтерактивних завдань, доступ до навчального матеріалу, можливість взаємодії з викладачами та співстудентами, а також аналіз прогресу та персоналізацію навчального процесу. Такий застосунок

може стати важливим інструментом для підтримки і розвитку самостійної роботи студентів, сприяючи їхньому успіху у навчанні та подальшій кар'єрі.

1.2 Сучасні методи та технології організації самостійної роботи

Сучасні методи та технології організації самостійної роботи студентів у вищій освіті включають в себе різноманітні підходи та інструменти, які сприяють ефективній організації та підтримці навчального процесу. Для реалізації теми дипломної роботи про розробку застосунку для підтримки самостійної роботи студентів, важливо розглянути такі методи та технології:

• Електронні навчальні платформи: Розробка застосунку, що інтегрується з електронними навчальними платформами, дозволяє студентам отримувати доступ до навчального матеріалу, виконувати завдання та отримувати зворотний зв'язок в онлайн-режимі.

• Інтерактивні методи навчання: Застосування інтерактивних методів, таких як відеоуроки, ігрові елементи, онлайн-тести тощо, допомагає стимулювати активну участь студентів у навчальному процесі та забезпечує їхнє зацікавлення.

• Онлайн-курси та відкрите навчання: Розробка модулів для онлайн-курсів та платформ відкритого навчання дозволяє студентам вивчати новий матеріал та розвивати навички у відповідності до власного графіку та потреб.

• Мультимедійні засоби: Використання мультимедійних засобів, таких як анімації, відео та інтерактивні демонстрації, допомагає зробити навчальний матеріал більш доступним та зрозумілим для студентів.

• Адаптивні технології: Розробка застосунків з використанням адаптивних технологій, які аналізують прогрес та потреби студентів для надання персоналізованих рекомендацій та завдань.

Розробка застосунку з використанням мови програмування Python може дозволити імплементувати ці сучасні методи та технології для організації та підтримки самостійної роботи студентів у вищій освіті. Розробка інтерфейсу користувача, баз даних, аналітичних модулів та адаптивних алгоритмів - це лише

деякі з можливих напрямків роботи над застосунком, які можуть сприяти ефективній організації та підтримці самостійної роботи студентів.

Організація самостійної роботи студентів вищих навчальних закладів вимагає від освітніх інституцій забезпечення відповідних умов і ресурсів, що сприяють ефективному самостійному навчанню. Сучасні освітні технології відіграють ключову роль у цьому процесі, оскільки вони дозволяють студентам гнучко керувати своїм навчальним процесом, адаптувати його під особисті освітні потреби та стилі навчання. Ефективна організація самостійної роботи передбачає створення системи, яка забезпечує доступ до навчальних матеріалів, ресурсів та інструментів для самоконтролю. Важливим є забезпечення студентів платформами, де вони можуть отримати доступ до лекцій, електронних книг, наукових статей, відеоматеріалів та інтерактивних завдань. Такі ресурси повинні бути легкодоступними та зручними у використанні, що допоможе студентам ефективніше планувати свій час і зусилля на самостійне вивчення матеріалу. Залучення інтерактивних технологій також відіграє важливу роль у мотивації студентів. Використання онлайн-квізів, самотестувань та віртуальних лабораторій може зробити процес навчання більш захоплюючим та динамічним. Це не тільки підтримує зацікавленість студентів, але й допомагає їм краще засвоювати інформацію через активну участь у навчальному процесі.

Ключовим аспектом самостійної роботи є наявність зворотного зв'язку від викладачів. Система має включати функціонал для подання зворотнього зв'язку студентам щодо їх прогресу, виявлення слабких місць та надання рекомендацій для подальшого вдосконалення. Онлайн платформи можуть автоматизувати цей процес, надаючи викладачам засоби для моніторингу виконання завдань та аналізу результатів тестів у реальному часі. Гнучкість у навчанні — ще одна важлива характеристика сучасної освіти. Студентам потрібна можливість адаптувати навчальний процес під свій індивідуальний графік та особистісні особливості. Це включає можливість вибору між різними форматами навчальних матеріалів, регулювання темпу вивчення та вибір специфічних тем або курсів, які відповідають їхнім кар'єрним цілям та інтересам. Організація самостійної роботи студентів не

обмежується лише технічним забезпеченням. Важливо також створити культуру самостійного навчання в навчальному закладі, де студенти відчувають підтримку з боку викладачів та адміністрації. Залучення студентів до активної участі у формуванні навчального процесу, збір їхніх відгуків і пропозицій, та організація робочих груп чи дискусійних клубів може значно підсилити ефективність самостійного навчання.

Такий комплексний підхід до організації самостійної роботи студентів забезпечує не тільки засвоєння знань, але й розвиток ключових компетенцій, таких як самоменеджмент, критичне мислення та здатність до постійного саморозвитку, що є надзвичайно важливим у сучасному світі.

1.3 Використання Python для створення навчальних застосунків

Однією з ключових тем у контексті розробки навчальних застосунків для підтримки самостійної роботи студентів є використання мови програмування Python. Python є потужним та універсальним інструментом з широким спектром можливостей, які сприяють створенню ефективних та інноваційних навчальних інструментів.

Python відзначається простим синтаксисом, що робить його доступним для вивчення та розуміння навіть для початківців. Це дозволяє швидко створювати навчальні застосунки без великих витрат часу на ознайомлення з складними конструкціями мови програмування.

Крім того, Python має велику кількість бібліотек та фреймворків, які допомагають в розробці різноманітних функцій у навчальних застосунках. Наприклад, бібліотеки для роботи з графікою, обробки даних, створення веб-додатків тощо дозволяють розширювати можливості навчального застосунку та надавати студентам нові можливості для самостійного навчання.

Однією з основних переваг використання Python є його крос-платформеність. Програми, розроблені на Python, можуть запускатися на різних операційних

системах, таких як Windows, macOS та Linux, що робить їх доступними для широкого кола користувачів.

Python також дозволяє створювати інтерактивні вправи та завдання для студентів, що полегшує їхнє засвоєння матеріалу. За допомогою відповідних бібліотек можна побудувати ігрові елементи, анімації, графіки тощо, що робить навчання більш цікавим та ефективним.

Крім того, Python надає можливості для створення систем штучного інтелекту та машинного навчання для навчання та оцінювання студентів. Такі системи можуть використовуватися для автоматичної перевірки завдань, аналізу прогресу студентів та надання індивідуалізованих рекомендацій щодо подальшого навчання.

Отже, використання Python для розробки навчальних застосунків є важливим та перспективним напрямком, який дозволяє покращити якість навчання та забезпечити студентам зручний та ефективний інструмент для самостійного навчання.

1.4 Інтерактивні елементи для залучення студентів до самостійної роботи

Один із ключових аспектів успішної самостійної роботи студентів полягає в їхньому зацікавленні та залученні до навчального процесу.

Перш за все, інтерактивність може бути втілена у формі відеоуроків, вебінарів або онлайн-лекцій, де студенти можуть взаємодіяти з викладачами або іншими студентами у реальному часі. Це дозволить створити атмосферу співпраці та обміну досвідом, що мотивуватиме студентів до активної участі у навчальному процесі.

Другим важливим інтерактивним елементом є можливість взаємодії з навчальними матеріалами через різноманітні завдання та тести. Наприклад, система може пропонувати студентам інтерактивні квізи з можливістю отримання

негайного зворотного зв'язку щодо правильності відповідей або інтерактивні вправи, які допоможуть їм зрозуміти матеріал краще.

Додатково, інтерактивність може бути забезпечена шляхом використання ігрових елементів у навчанні. Наприклад, можливість отримання балів або нагород за виконання завдань або відповідь на питання може стимулювати студентів до більш активної участі та досягнення кращих результатів.

Крім того, важливою частиною інтерактивних елементів є можливість спілкування та співпраці між студентами. Форуми обговорень, групові завдання або спільні проекти дозволять студентам обмінюватися думками, досвідом та підтримувати один одного у навчанні. Інтерактивність може стати центральним елементом, який сприяє активізації студентів у навчанні. Вона може бути реалізована через різноманітні формати, такі як відеоуроки, вебінари або онлайн-лекції, де студенти можуть взаємодіяти з викладачами або іншими студентами у реальному часі. Такий підхід створює атмосферу співпраці та обміну досвідом, що мотивує студентів до активної участі в навчальному процесі.

Крім того, інтерактивність може забезпечуватися через взаємодію з навчальними матеріалами за допомогою різноманітних завдань та тестів. Наприклад, система може пропонувати студентам інтерактивні квізи з можливістю отримання негайного зворотного зв'язку щодо правильності відповідей або інтерактивні вправи, які допомагають їм краще зрозуміти матеріал.

Додатковим елементом інтерактивності може бути використання ігрових компонентів у навчанні. Наприклад, можливість отримання балів чи нагород за виконання завдань або відповідь на питання може стимулювати студентів до більш активної участі та досягнення кращих результатів.

Також важливою частиною інтерактивних елементів є можливість спілкування та співпраці між самими студентами. Форуми обговорень, групові завдання або спільні проекти дозволяють студентам обмінюватися думками, досвідом та підтримувати один одного у навчанні. Такі формати сприяють поглибленню знань через взаємне взаємодоповнення та допомагають студентам краще засвоювати матеріал.

Узагальнюючи, інтерактивність в навчальному процесі є ключовим фактором, який сприяє активізації студентів, покращує їхнє розуміння матеріалу та мотивує до досягнення високих результатів.

1.5 Персоналізація навчального процесу за допомогою застосунку

Розробка застосунку для підтримки самостійної роботи студентів вищих навчальних закладів є складним і багатогранним завданням. Одним із ключових аспектів успішної реалізації цього завдання є забезпечення персоналізації навчального процесу, тобто створення умов для індивідуального підходу до кожного студента з урахуванням його потреб, здібностей та інтересів.

Персоналізація навчання передбачає створення такого середовища, в якому кожен студент може отримати найефективніший та оптимальний навчальний досвід. Основні аспекти персоналізації навчання через застосунок включають адаптацію навчального матеріалу, розробку індивідуальних навчальних планів, врахування інтересів та стиль навчання кожного студента, а також надання персоналізованого зворотного зв'язку та підтримки.

Одним з основних елементів персоналізації є адаптація навчального матеріалу до потреб та рівня знань кожного студента. За допомогою алгоритмів машинного навчання та аналізу даних застосунок може створювати індивідуалізовані навчальні програми, які враховують сильні та слабкі сторони кожного студента, його попередні знання та потреби.

Додатково, персоналізація може включати створення індивідуальних навчальних планів для кожного студента. Застосунок може аналізувати академічні цілі та індивідуальні потреби студента, враховуючи його часові обмеження та ритми життя, та надавати рекомендації щодо оптимального розподілу часу та задач для досягнення успіху.

Крім того, застосунок може використовувати інтерактивні елементи для залучення студентів до навчання та підвищення їхньої мотивації. Це можуть бути ігрові елементи, відеоуроки, вікторини та інші форми активної участі, які

допоможуть студентам краще засвоювати матеріал та відчувати зацікавленість у навчанні.

Завдяки персоналізації навчання через застосунок студенти отримують можливість навчатися у власному темпі, враховуючи їхні індивідуальні потреби та стиль навчання. Це дозволяє кожному студентові максимально реалізувати свій потенціал та досягти успіху в навчанні.

1.6 Огляд аналогів

1.6.1 Огляд засобів організації самостійної роботи студентів в системі дистанційного навчання Moodle

Організація самостійної роботи студентів у системі дистанційного навчання Moodle включає в себе різноманітні засоби та можливості, які представлені на рис.1.1.

В Moodle кожен курс складається з модулів, які можуть включати лекції, завдання, тести, форуми, відеоматеріали тощо. Це дає студентам можливість самостійно вивчати матеріал, виконувати завдання та практичні вправи у зручний для них час. Викладачі можуть створювати різні типи завдань, такі як написання есе, виконання практичних завдань, розв'язання задач тощо. Студенти можуть завантажувати свої роботи, а викладачі здійснювати оцінювання та надавати зворотний зв'язок. Moodle підтримує створення форумів, де студенти можуть обговорювати теми, обмінюватися думками та досвідом, задавати питання викладачам. Moodle дозволяє завантажувати та споживати відео- та аудіоматеріали. Це може бути запис лекцій, додаткові пояснення матеріалу, інструкції з виконання завдань тощо. Студенти мають можливість переглядати ці матеріали в зручний для них час.

Рис. 1.1 – Засоби підтримки самостійної роботи студента в системі Moodle

В Moodle можна надавати доступ до електронних підручників, наукових статей, веб-ресурсів із додатковою інформацією щодо теми курсу. Також Moodle підтримує автоматизоване оцінювання завдань і тестів, що дозволяє студентам отримувати миттєвий результат та зворотний зв'язок щодо їхніх успіхів та помилок. Крім того, Moodle дозволяє налаштовувати курси та завдання відповідно до потреб кожного студента. Це може включати індивідуальне планування завдань або використання персоналізованих рекомендаційних систем для додаткового навчання.

На рис. 1.2 наведено приклад екранної форми курсу в системі Moodle з завданнями для самостійної роботи студентів.

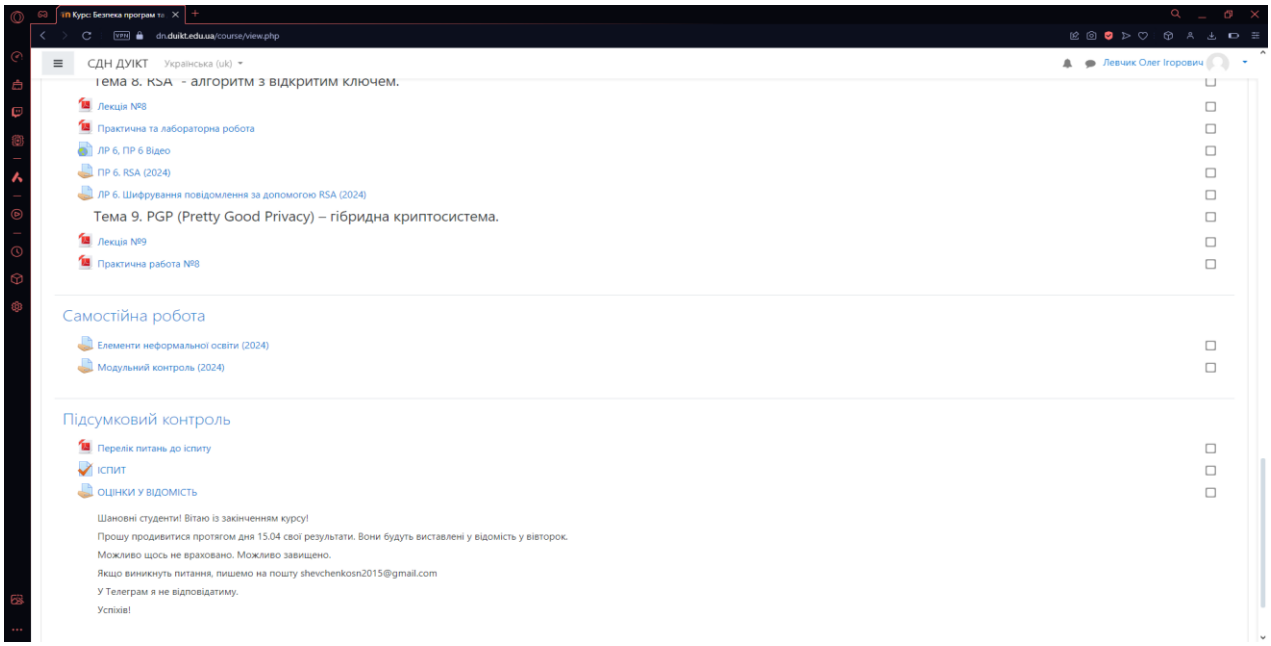


Рис. 1.2 – Сторінка курсу в системі Moodle
з завданнями для самостійної роботи студетнів

1.6.2 Огляд засобів організації самостійної роботи студентів в Google Classroom

Google Classroom організація самостійної роботи студентів забезпечується різноманітними інструментами, які сприяють ефективному навчанню та саморозвитку, які представлені на рисунку 1.3.

Рис. 1.3 – Засоби організації самостійної роботи студентів в Google Classroom

Викладачі можуть створювати різні типи завдань, такі як текстові завдання, завдання з відео чи аудіо матеріалами, тестові завдання тощо. Кожне завдання може мати визначений термін виконання та максимальну кількість балів. Студенти мають можливість відправляти свої відповіді на завдання, будь то текст, прикріплення файлів чи посилання на ресурси. Це дозволяє їм працювати над завданнями відповідно до власного розкладу і в зручному для них форматі. В Google Classroom викладачі можуть оцінювати завдання студентів, залишати коментарі та зворотний зв'язок. У Google Classroom викладачі можуть розміщувати матеріали курсу, такі як презентації, відео, посилання на сторонні ресурси або електронні книги. Це дозволяє студентам самостійно ознайомлюватися з необхідною інформацією та поглиблювати своє розуміння предмету. Також, Google Classroom підтримує функцію обговорення, де студенти можуть спілкуватися, задавати питання, обмінюватися думками та досвідом. Це стимулює активну взаємодію між учасниками курсу та сприяє спільному навчанню. Google Classroom забезпечує доступ до всіх матеріалів і завдань через будь-який пристрій з підключенням до Інтернету. Це дозволяє студентам працювати з курсом з будь-якого місця і в будь-який час.

На рис. 1.4 наведено приклад екранної форми Google Classroom з завданнями для самостійної роботи здобувача вищої освіти.

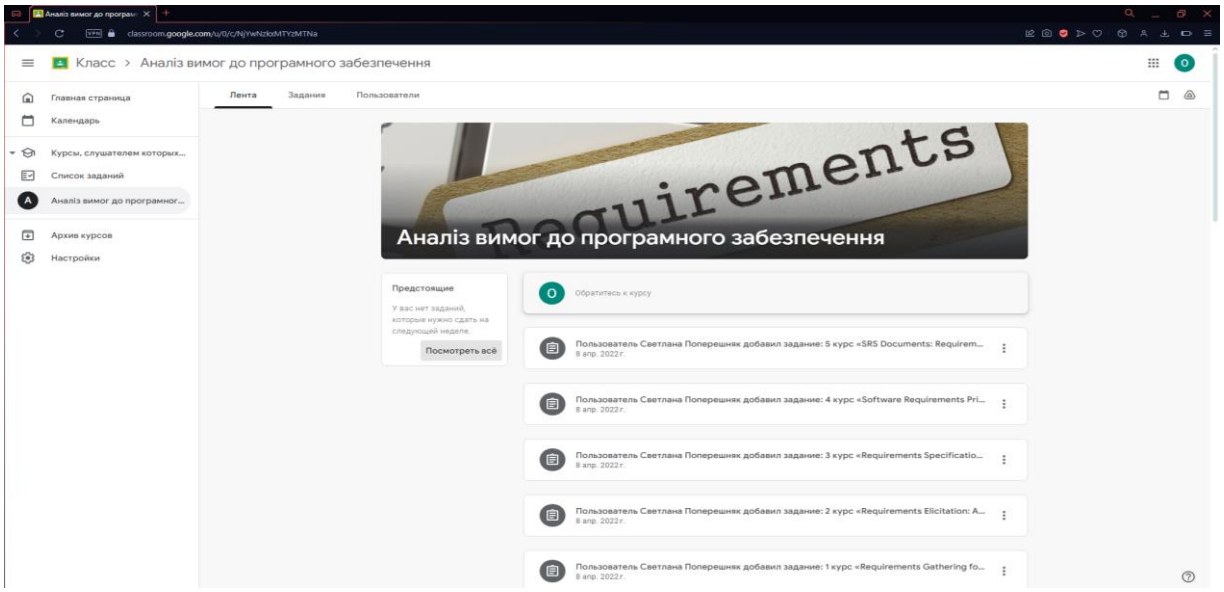


Рис. 1.4 – Сторінка в системі Google Classroom
з завданнями для самостійної роботи здобувача вищої освіти.

1.6.3 Порівняльний аналіз аналогів

Для визначення вимог до застосунку, що дозволять йому конкурувати з аналогічними системами, проведено аналіз окремих показників розглянутих в п.1.6.1 та 1.6.2 систем. Результати порівняння наведено в таблиці 1.1.

Розроблений додаток підтримує відкрите API, що дозволяє розробникам легко інтегрувати нові функції. Він також має інтеграцію з Telegram, чого немає у Moodle та Google Classroom. Реєстрація користувачів, додавання, редагування і видалення завдань у додатку здійснюються через Telegram бот або веб-інтерфейс, тоді як в аналогах це можливо лише через веб-інтерфейс. Додаток не має функції відображення розкладу, але забезпечує відправлення нагадувань про завдання через Telegram бот, що є важливою функцією для студентів. Moodle також має функцію відправлення нагадувань, але без інтеграції з Telegram. Щодо пристроїв, додаток доступний через веб-сайт, а Google Classroom додатково підтримує мобільні пристрої Android та iOS. Для створення курсів розроблений додаток надає доступ на рівні користувача, тоді як у Moodle це здійснюється на рівні адміністратора, а в Google Classroom - на рівні користувача. Також, розроблений додаток має функцію ведення списків оцінок і розрахунку середньої оцінки, що

робить його корисним для студентів, у той час як Moodle та Google Classroom мають схожі функції, але для викладачів.

Таблиця 1.1

Порівняльний аналіз аналогів

Критерій порівняння	**Застосунок для організації самостійної роботи**	**Moodle**	**Google Classroom**
Відкрите API	Так	Обмежені можливості	Обмежені можливості
Інтеграція з Telegram	Так	Немає	Немає
Реєстрація користувачів	Через Telegram бот або через веб-інтерфейс	Через веб-інтерфейс	Через веб-інтерфейс
Додавання завдань	Через Telegram бот або через веб-інтерфейс	Через веб-інтерфейс	Через веб-інтерфейс
Редагування завдань	Через Telegram бот або через веб-інтерфейс	Через веб-інтерфейс	Через веб-інтерфейс
Видалення завдань	Через Telegram бот або через веб-інтерфейс	Через веб-інтерфейс	Через веб-інтерфейс
Відображення розкладу	Немає	Так	Так
Відправлення нагадувань	Через Telegram бот	Так	Так
Пристрої	Website	Website	Android, IOS, Website
Створення курсу	Доступ на рівні користувача	Доступ на рівні адміністратора	Доступ на рівні користувача
Оцінки	Список оцінок, середня оцінка	Формуються списки з оцінками	Список оцінок для викладача

2 ПРОЕКТУВАННЯ ЗАСТОСУНКУ

2.1 Моделювання вимог до програмного забезпечення та визначення архітектури застосунку

Першим кроком у моделюванні застосунку є проведення аналізу потреб користувачів. Для цього були проведені опитування та інтерв'ю з потенційними користувачами — студентами та викладачами. Основними виявленими потребами стали:

- Зручний інтерфейс для управління завданнями.

- Нагадування про дедлайни.

- Інтеграція з популярними месенджерами, такими як Telegram.

- Інструменти для відстеження прогресу та оцінювання результатів.

На основі аналізу потреб користувачів було визначено функціональні та нефункціональні вимоги до застосунку та побудовано діаграму варіантів використання (рис.2.1).

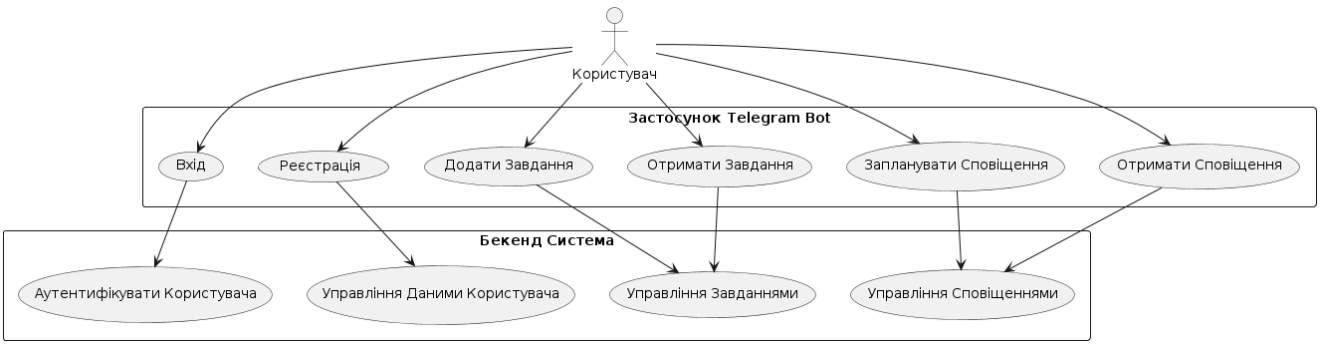


Рис.2.1 - Діаграма варіантів використання

Діаграма варіантів використання показує, як працює веб-застосунок для підтримки самостійної роботи здобувачів вищої освіти на мові Python. Користувач може увійти в систему, зареєструватися, додавати завдання, отримувати список завдань і налаштовувати сповіщення через Telegram бот. Бекенд системи обробляє всі ці дії: аутентифікує користувача, зберігає дані, керує завданнями і забезпечує відправлення сповіщень.

Це допомагає користувачам ефективно керувати своїми завданнями і отримувати важливі нагадування, що робить їхню самостійну роботу більш організованою і продуктивною.

Функціональні вимоги включають:

- Реєстрація та авторизація користувачів через Telegram або веб-інтерфейс.
- Додавання, редагування та видалення завдань.
- Відправлення нагадувань про наближення дедлайнів через Telegram.
- Відображення розкладу завдань та курсів.
- Можливість перегляду та оцінювання виконаних завдань.

Нефункціональні вимоги включають:

- Висока швидкість відгуку системи.
- Інтуїтивно зрозумілий інтерфейс користувача.
- Забезпечення безпеки даних користувачів.
- Масштабованість системи для підтримки великої кількості користувачів.

Архітектура програмного застосунку базується на клієнт-серверній моделі. Це дозволяє розділити обробку даних і користувацький інтерфейс, забезпечуючи гнучкість і масштабованість системи.

Серверна частина (Back-end): реалізована з використанням мови програмування Python та фреймворку Django. Вона відповідає за обробку запитів від клієнтів, взаємодію з базою даних та зовнішніми сервісами, такими як Telegram API.

Клієнтська частина (Front-end): реалізована з використанням JavaScript та HTML/CSS. Вона забезпечує користувацький інтерфейс, через який студенти можуть взаємодіяти із системою, додавати, редагувати та переглядати завдання.

2.2 Моделювання процесів виконання ключових задач користувачів

Для моделювання процесів виконання ключових задач користувачів застосовано діаграми послідовності.

Перша діаграма послідовності аутентифікація та додавання завдання показує як користувач надсилає команду для входу в систему, після чого бот перевіряє цю команду та передає її на модуль аутентифікації.

Модуль аутентифікації перевіряє дані користувача в базі даних, і якщо все в порядку, користувач отримує повідомлення про успішний вхід. Потім користувач може надіслати команду для додавання нового завдання, яку бот також обробляє і додає в базу даних. В результаті користувач отримує підтвердження, що завдання додано.

На рисунку 2.2 представлена діаграма послідовності аутентифікація та додавання завдання.

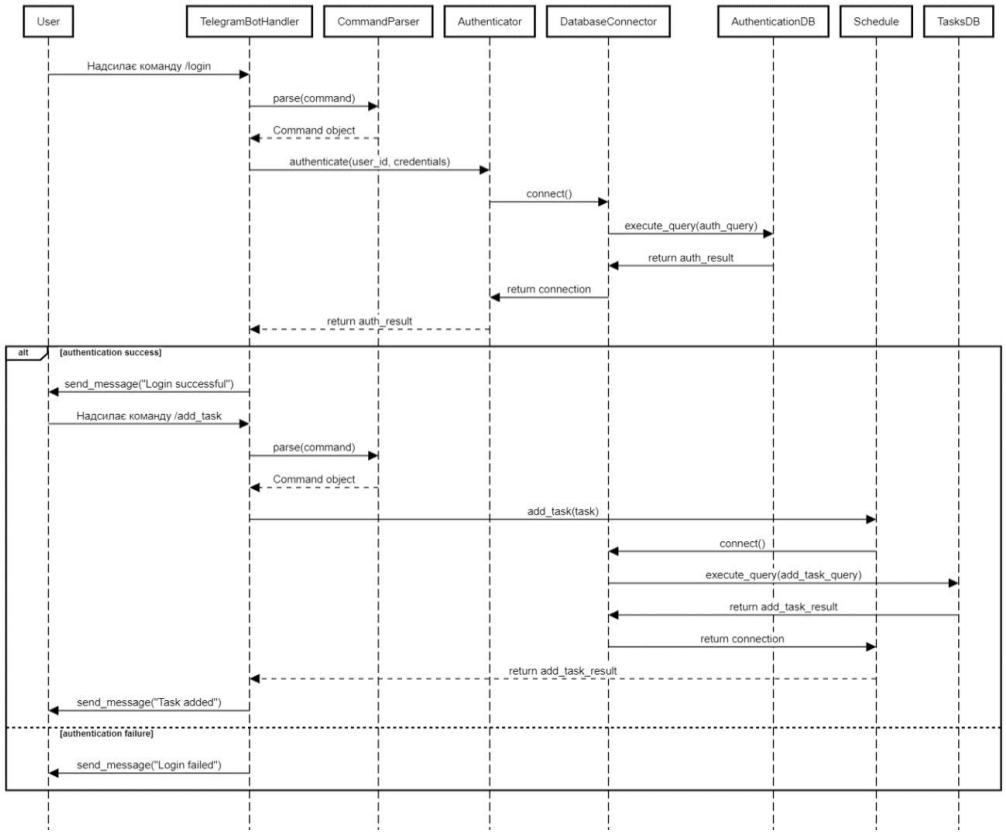


Рис. 2.2 Аутентифікація та додавання завдання

На рисунку 2.3 представлена діаграми послідовності аутентифікація та додавання завдання.

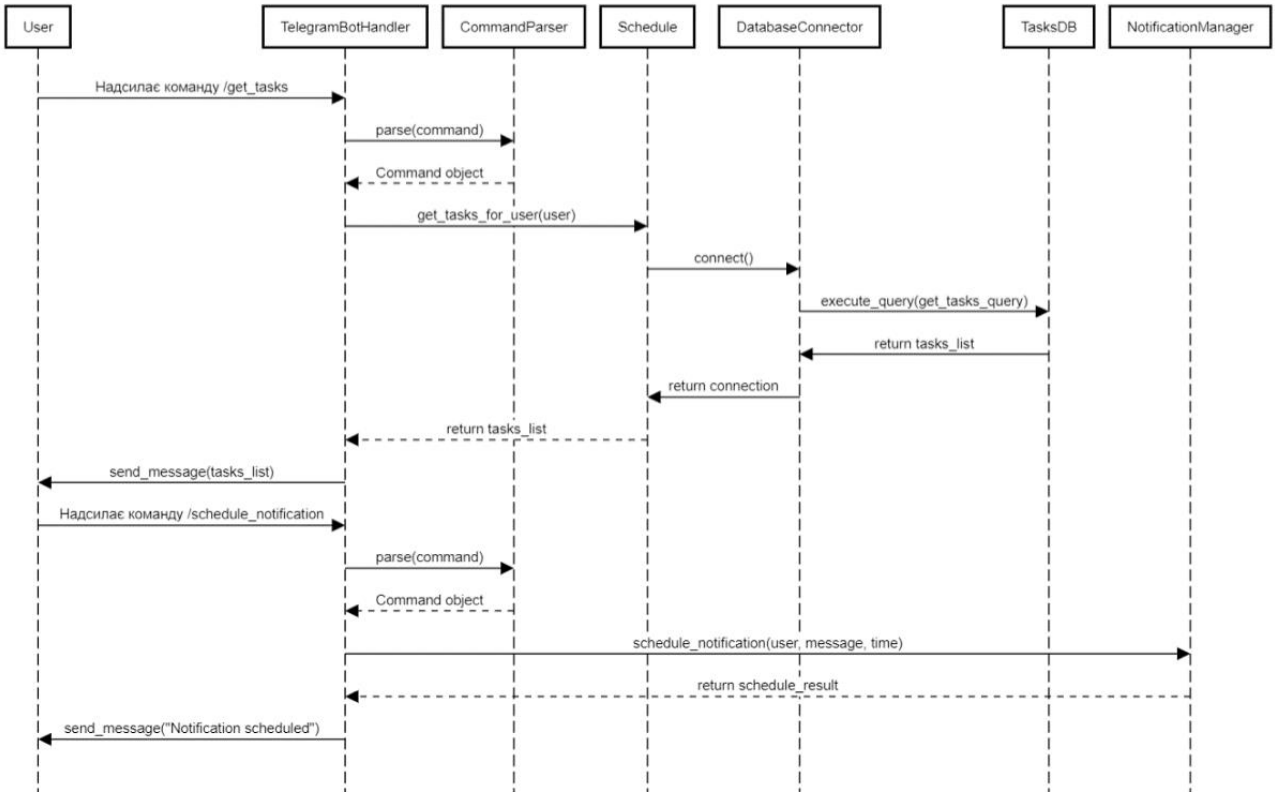


Рис. 2.3 Отримання завдань та планування сповіщень

Діаграма отримання завдань та планування сповіщень показує, як користувач може отримати список своїх завдань та запланувати сповіщення. Користувач надсилає команду для отримання своїх завдань, бот обробляє цю команду, звертається до бази даних і повертає список завдань користувачеві.

Крім того, користувач може надіслати команду для планування сповіщення. Бот приймає цю команду, обробляє її та налаштовує сповіщення на певний час, після чого користувач отримує підтвердження, що сповіщення заплановано.

2.3 Моделювання нефункціональних вимог

До нефункціональних вимог можна віднести вимоги до даних що зберігаються, вимоги до інтерфейсу та безпеки і конфенденційності даних користувачів.

Для зберігання даних застосунку використовується реляційна база даних. Основні таблиці бази даних включають наступне:

- Users: зберігає інформацію про користувачів, такі як імена, електронні адреси та паролі.

- Tasks: зберігає дані про завдання, включаючи опис, дедлайн та статус.

- Courses: зберігає дані про курси, включаючи назву курсу та список завдань, що до нього відносяться.

Проектування бази даних включає нормалізацію даних для мінімізації надлишковості та забезпечення цілісності даних.

Інтерфейс користувача (UI) є ключовим аспектом застосунку, оскільки він визначає, як користувачі взаємодіятимуть із системою. Основні елементи інтерфейсу включають:

- Головна сторінка з оглядом завдань та курсів.

- Форми для додавання та редагування завдань.

- Налаштування для отримання повідомлень через Telegram.

Проектування UI базується на принципах юзабіліті, що забезпечують зручність та інтуїтивність взаємодії. Важливими аспектами є використання зрозумілих іконок, логічне розташування елементів та забезпечення доступності для всіх категорій користувачів.

Безпека даних користувачів є критично важливим аспектом застосунку. Основні заходи безпеки включають:

- Шифрування даних при передачі та зберіганні.

- Використання надійних методів автентифікації та авторизації.

- Регулярне оновлення системи безпеки для захисту від нових загроз.

Конфіденційність даних забезпечується шляхом обмеження доступу до даних тільки авторизованим користувачам та застосуванням політики конфіденційності, що описує, як обробляються персональні дані.

Моделювання програмного застосунку є ключовим етапом розробки, який забезпечує створення ефективного та надійного інструменту для підтримки самостійної роботи здобувачів вищої освіти. Завдяки ретельному аналізу потреб користувачів, визначенню вимог, проектуванню архітектури та бази даних, розробці інтуїтивного інтерфейсу користувача, а також забезпеченню високого рівня безпеки, створений застосунок відповідає всім поставленим завданням та сприяє покращенню організації навчального процесу.

2.2 Проектування інтерфейсу користувача

Проектування інтерфейсу користувача (UI) є однією з найважливіших складових процесу розробки програмного засобу. У контексті створення засобу для підтримки самостійної роботи студентів вищої освіти, інтерфейс відіграє ключову роль у забезпеченні зручності та ефективності його використання. Основна мета проектування інтерфейсу полягає у створенні інтуїтивно зрозумілого та привабливого середовища, яке сприяє легкому доступу до функціоналу та підвищує продуктивність користувачів. Процес проектування інтерфейсу розпочинається з аналізу потреб користувачів. Було проведено детальне дослідження, включаючи опитування та інтерв'ю з потенційними користувачами — студентами та викладачами. На основі отриманих даних було виявлено основні сценарії використання та функції, які повинні бути доступні у засобі. Це дозволило визначити ключові елементи інтерфейсу, такі як сторінка з оглядом завдань, форми для додавання та редагування завдань, а також налаштування для отримання повідомлень через Telegram. Проектування інтерфейсу враховує принципи юзабіліті, що забезпечують зручність та інтуїтивність використання. Важливо, щоб користувачі могли легко орієнтуватися в засобі та швидко знаходити необхідні

функції. Для цього використовується логічне розташування елементів, зрозумілі іконки та прості форми. Особлива увага приділяється забезпеченню послідовності в дизайні, що допомагає користувачам швидко адаптуватися до інтерфейсу. Головна сторінка засобу надає користувачам огляд їхніх завдань та курсів. Вона містить зручні фільтри для сортування завдань за датою, пріоритетом або статусом виконання. Це дозволяє студентам швидко отримувати актуальну інформацію про свої завдання та планувати свій час ефективніше. Додатково, на головній сторінці можуть бути відображені нагадування про наближення дедлайнів та важливі повідомлення від викладачів. Форми для додавання та редагування завдань розроблені з урахуванням мінімізації зусиль, необхідних для введення даних. Вони містять поля для введення назви завдання, опису, дати дедлайну та пріоритету. Використання автозаповнення та випадаючих списків допомагає зменшити кількість помилок при введенні інформації та прискорити процес створення завдань. Важливо, щоб форми були доступні на будь-якому пристрої, тому вони оптимізовані для мобільних пристроїв. Налаштування для отримання повідомлень через Telegram дозволяють користувачам персоналізувати повідомлення, які вони отримують від засобу. Це може включати нагадування про наближення дедлайнів, оновлення статусу завдань або важливі повідомлення від викладачів. Інтерфейс налаштувань повинен бути простим та зрозумілим, щоб користувачі могли легко змінювати свої уподобання. Значну увагу також приділено візуальному дизайну інтерфейсу. Використання привабливих кольорових схем, сучасних шрифтів та графічних елементів підвищує привабливість засобу та робить його використання приємнішим. Водночас, дизайн не повинен бути перевантаженим — важливо зберігати баланс між естетикою та функціональністю. Після створення перших макетів інтерфейсу проводяться юзабіліті-тестування. Це дозволяє отримати зворотній зв'язок від користувачів та виявити можливі проблеми у дизайні на ранніх етапах розробки. На основі результатів тестування вносяться корективи, що покращують зручність використання та ефективність інтерфейсу. Тестування проводиться ітеративно, що дозволяє поступово вдосконалювати інтерфейс.

Проектування інтерфейсу також включає забезпечення доступності для всіх категорій користувачів, включаючи людей з обмеженими можливостями. Це означає використання відповідних контрастів кольорів, забезпечення підтримки екранних читачів та інших допоміжних техннологій. Інтерфейс повинен бути зручним для використання як на комп'ютерах, так і на мобільних пристроях, тому адаптивний дизайн є обов'язковим. У процесі проектування інтерфейсу використовуються різноманітні інструменти та технології. Це включає прототипування з використанням інструментів типу Figma або Adobe XD, розробку з використанням HTML, CSS та JavaScript, а також фреймворків, таких як React або Angular. Використання сучасних технологій дозволяє забезпечити високу продуктивність та якість інтерфейсу. Проектування інтерфейсу користувача є безперервним процесом, який продовжується навіть після випуску засобу. Отримання зворотного зв'язку від користувачів та аналіз їхньої поведінки допомагають виявляти нові потреби та можливості для поліпшення. Регулярні оновлення та вдосконалення інтерфейсу забезпечують його відповідність сучасним стандартам та вимогам користувачів. Таким чином, проектування інтерфейсу користувача — це комплексний процес, що об'єднує в собі аналіз потреб користувачів, розробку і тестування з метою створення ефективного, зручного та привабливого середовища для користувачів програмного засобу.

Крім основних аспектів проектування інтерфейсу, важливо також звернути увагу на адаптацію до сучасних трендів у дизайні. Наприклад, в останні роки популярність здобувають темні режими інтерфейсів, які не тільки зменшують навантаження на очі користувачів, але й підвищують читабельність та зручність в умовах низької освітленості. Такі режими також можуть допомагати зекономити енергію на мобільних пристроях з OLED-екранами.

Крім того, важливим елементом сучасного дизайну є анімації та мікроінтеракції, які зроблюють взаємодію з програмою більш приємною та емоційно насиченою. Це може включати анімовані переходи між сторінками, плавні зміни стану елементів при взаємодії користувача, а також інші ефекти, що роблять використання програмного засобу більш інтуїтивним та цікавим.

Необхідно також зазначити значення психологічних аспектів у дизайні інтерфейсу. Використання правильних кольорів, типографії та композиції може впливати на емоційний стан користувача та сприйняття інформації. Наприклад, використання спокійних та приємних кольорів може знижувати рівень стресу у користувачів під час використання програми.

Таким чином, інтеграція сучасних тенденцій дизайну, включаючи темні режими, анімації та урахування психологічних аспектів, покращує не лише зовнішній вигляд інтерфейсу, але й підвищує загальний комфорт та задоволення від використання програмного засобу.

3 РОЗРОБКА ПРОГРАМНОГО ЗАСТОСУНКУ

3.1 Інструменти та засоби розробки

Для реалізації проекту було використано кілька ключових програмних засобів. PyCharm — це інтегроване середовище розробки (IDE) для Python, яке значно спрощує процес написання, тестування та налагодження коду. Python є основною мовою програмування для розробки логіки нашого застосунку. Його обрали через його простоту, універсальність та широку екосистему бібліотек і фреймворків. Python дозволяє писати чистий і зручний для підтримки код, що сприяє прискоренню циклів розробки та полегшенню співпраці між членами команди. Для створення графічного інтерфейсу користувача (GUI) використовується фреймворк Customtkinter, побудований на основі стандартної бібліотеки tkinter Python. Його сучасний та візуально привабливий набір елементів інтерфейсу дозволяє легко налаштувати вигляд програми з мінімальним кодом. До Customtkinter в проекті використовуються додаткові бібліотеки Python для розширення функціональності і спрощення розробки, такі як бібліотека os для операцій з файлами. Для веб-розробки на Python використовується фреймворк Django, який допомагає швидко створювати безпечні та масштабовані веб-застосунки. JavaScript використовується для надання інтерактивності веб-застосункам, створення динамічних елементів на веб-сторінках. HTML5 і CSS3 використовуються для створення структури і стилю веб-застосунку: HTML5 для розмітки контенту, а CSS3 — для його візуального оформлення. SQLite — легка база даних, яка ідеально підходить для зберігання даних нашого застосунку, особливо для невеликих і середніх проектів. Використані інструменти дозволяють створити ефективний і зручний у використанні веб-застосунок для підтримки самостійної роботи студентів. Для реалізації проекту було використано кілька ключових програмних засобів. PyCharm, як інтегроване середовище розробки для мови програмування Python, значно полегшує процес написання, тестування і

налагодження коду. Python обрано як основну мову програмування завдяки його простоті, універсальності і багатій екосистемі бібліотек і фреймворків, що сприяє швидкій розробці і спрощує співпрацю в команді. Для створення графічного інтерфейсу користувача (GUI) використовується фреймворк Customtkinter, побудований на основі стандартної бібліотеки tkinter Python. Цей фреймворк забезпечує сучасний та естетичний набір елементів інтерфейсу, таких як кнопки, мітки, поля введення та індикатори виконання, з можливістю легкої настройки їх візуального вигляду. Простий у використанні API та обширна документація дозволяють швидко створювати адаптивні та інтерактивні користувацькі інтерфейси з мінімальними зусиллями. Окрім Customtkinter, в проекті використовуються інші бібліотеки Python для розширення функціональності і спрощення розробки. Наприклад, бібліотека os широко використовується для роботи з файлами, що робить її незамінною у випадках, коли потрібно взаємодіяти з операційною системою. Для веб-розробки використовується фреймворк Django, що дозволяє ефективно створювати безпечні та масштабовані веб-застосунки на Python. JavaScript використовується для додавання інтерактивності на веб-сторінках, створення динамічних елементів і покращення користувацького досвіду. HTML5 і CSS3 використовуються для створення структури і стилю веб-застосунку: HTML5 для розмітки контенту, а CSS3 — для його візуального оформлення та створення привабливого зовнішнього вигляду. SQLite використовується як легка база даних для зберігання даних у проекті. Вона ідеально підходить для невеликих і середніх обсягів даних, забезпечуючи зручну і швидку роботу з інформацією. Ці інструменти не лише дозволяють ефективно розробляти інноваційні технології, але й підтримують самостійну роботу студентів у процесі навчання та виконання проектів.

3.2 Розробка класів системи

Основними елементами системи є класи, які забезпечують роботу з користувачами, аутентифікацію та управлінням правами доступу. Діаграма класів

допомагає зрозуміти, які дані потрібно зберігати, як ці дані пов'язані між собою та які методи будуть використовуватись для маніпуляції даними. Діаграма класів показує, як працюють основні компоненти системи аутентифікації та управління доступом для застосунку, що підтримує самостійну роботу здобувачів освіти на мові Python. На рисунку 3.1 представлено UML діаграма класів, яка демонструє взаємодію між компонентами системи, відповідальними за роботу з telegram api та управління сповіщеннями в застосунку для підтримки самостійної роботи здобувачів освіти.

Основні класи включають нижченаведені (рис.2.2):

- User: представляє студента або викладача, що користується системою.

- Task: містить інформацію про конкретне завдання, включаючи його опис, дедлайн та статус виконання.

- Course: об'єднує завдання в рамках одного навчального курсу.

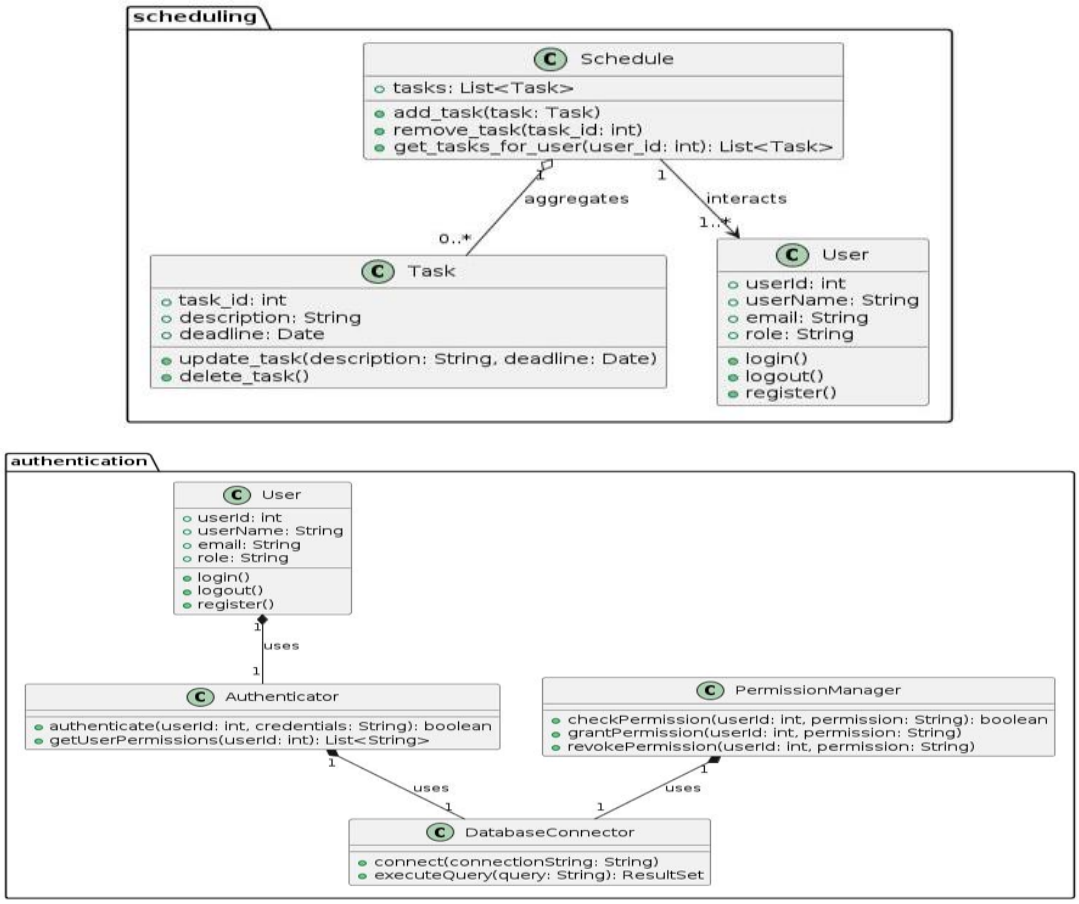


Рис. 3.1 Діаграма класів

Клас `User` представляє користувача системи і містить інформацію про користувача, таку як ідентифікатор користувача (`user_id`), ім'я користувача (`username`), електронну пошту (`email`) та роль (`role`). Цей клас також має методи для входу, виходу з системи та реєстрації нового користувача.

Клас `Authenticator` відповідає за перевірку достовірності користувачів. Він використовує методи для аутентифікації користувача за його ідентифікатором та обліковими даними, а також для отримання прав доступу користувача. Щоб виконати ці операції, `Authenticator` використовує об'єкт класу `DatabaseConnector`, який забезпечує виконання запитів.

Для управління правами доступу в системі використовується клас `PermissionManager`. Він перевіряє, надає і відкликає права доступу користувачів. Як і у випадку з аутентифікацією, цей клас також використовує `DatabaseConnector` для взаємодії з базою даних.

Клас `DatabaseConnector` відповідає за бази даних.

Усі ці класи взаємодіють між собою, щоб забезпечити надійну аутентифікацію користувачів та ефективне управління правами доступу в застосунку, який підтримує самостійну роботу здобувачів вищої освіти на мові Python.

В основі цієї системи лежить клас `TelegramBotHandler`, який взаємодіє з Telegram API для обміну повідомленнями з користувачами. Він містить інформацію про ключ API та список користувачів. Основні методи цього класу включають отримання повідомлень, відправлення повідомлень користувачам та розбір команд з повідомлень.

Клас `CommandParser` використовується для обробки команд, які отримує `TelegramBotHandler`. Він має єдиний метод `parse`, який приймає команду як вхідний параметр і повертає об'єкт типу `Command`. Цей клас допомагає розібрати текст команди і перетворити його в структуру, з якою зручно працювати далі.

Клас `NotificationManager` відповідає за управління чергою сповіщень. Він містить методи для планування сповіщень, де вказується користувач, повідомлення

та час відправлення, а також для відправлення всіх запланованих сповіщень. Цей клас допомагає забезпечити своєчасне інформування користувачів про важливі події та нагадування.

Клас `Schedule` містить список завдань (`tasks`), які потрібно виконати. Він надає методи для додавання нового завдання (`add_task`), видалення існуючого завдання (`remove_task`) та отримання списку завдань для конкретного користувача (`get_tasks_for_user`).

Клас `Task` представляє окреме завдання. Він містить інформацію про завдання, таку як ідентифікатор завдання (`task_id`), опис (`description`) та кінцевий термін виконання (`deadline`). Цей клас також надає методи для оновлення інформації про завдання (`update_task`) та видалення завдання (`delete_task`).

На діаграмі показано, що клас `Schedule` агрегує об'єкти класу `Task`, тобто містить список завдань. Також видно асоціацію між класами `Schedule` та `User`, що означає, що розклад може взаємодіяти з користувачами, наприклад, для отримання завдань, призначених конкретному користувачу.

3.3 Реалізація програмного застосунку

Наступним кроком оберемо та опишимо для реалізації програмного застосунку самостійної роботи здобувачів освіти на мові Python, клас FileSelector.

Опис класу FileSelector представлена на рисунку 3.7.

```
class FileSelector:
    def __init__(self, root):
        self.root = root
        self.selected_files = []
        self.selected_folders = []
        self.selected_destination = ""


        self.btn_select_files = tk.Button(root, text="Выбрать файлы", command=self.select_files)
        self.btn_select_files.pack()

        self.btn_select_folders = tk.Button(root, text="Выбрать папки", command=self.select_folders)
        self.btn_select_folders.pack()

        self.btn_select_destination = tk.Button(root, text="Выбрать путь назначения", command=self.select_destination)
        self.btn_select_destination.pack()
```

Рис. 3.2 Опис класу FileSelector

Наступний метод, який ми використовуємо для реалізації програмного застосунку має назву self.schedule та представлений на рисунку 3.8.

```
def set_schedule(self):
    cal = DateEntry(self.root, width=12, background='darkblue',
                    foreground='white', borderwidth=2, year=datetime.datetime.now().year)
    cal.pack(padx=10, pady=10)

    time_str = simpledialog.askstring(parent=self.root)
    try:
        time_obj = datetime.datetime.strptime(time_str, "%H:%M").time()
        scheduled_datetime = datetime.datetime.combine(cal.get_date(), time_obj)
        self.schedule = scheduled_datetime
        messagebox.showinfo("Schedule Set", f"Schedule set for {self.schedule}")
    except ValueError:
        messagebox.showerror("Error", "Invalid time format")
```

Рис. 3.3 Метод self.schedule

Представлений метод спочатку відкриває календар, де користувач може вибрати дату. Після цього через діалог запитується час. Якщо формат часу вірний, зберігається обране значення дати та часу в self.schedule. Цей метод отримує інформацію про дату і час.

На рисунку 3.27 представлено календар завдань веб-застосунку.

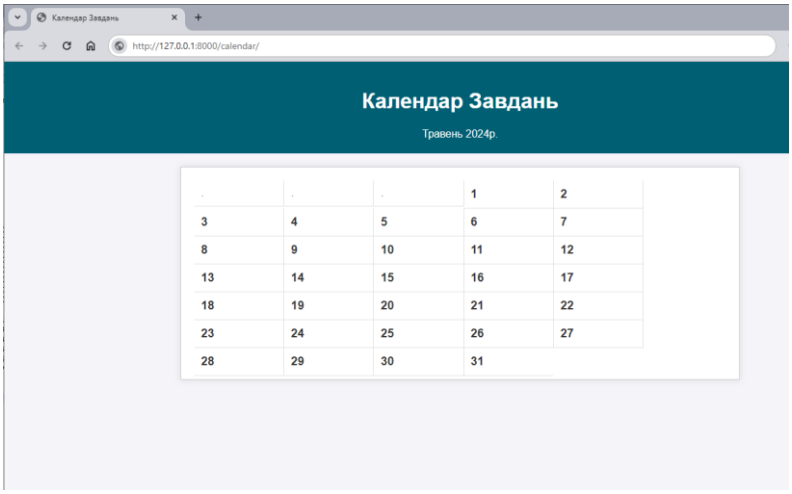


Рис. 3.4 Календар завдань

Для реалізації програмного застосунку в роботі було представлено екранні форми які зображені на рис. 3.4-3.7.

На рисунку 3.5 представлено форма Календар Завдань з опцією додавання завдання, де студенти можуть бачити всі свої заплановані завдання. Вибравши певний день, вони можуть отримати детальну інформацію про завдання, включаючи опис та дедлайн.

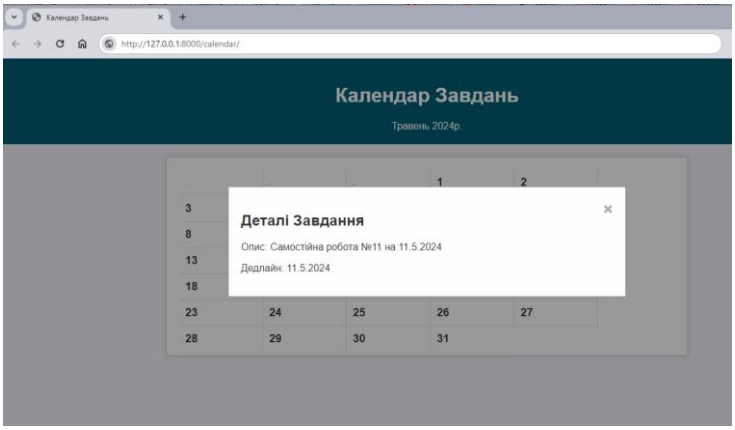


Рис. 3.5 Список завдань

На рисунку 3.6 представлено Мій Dashboard.

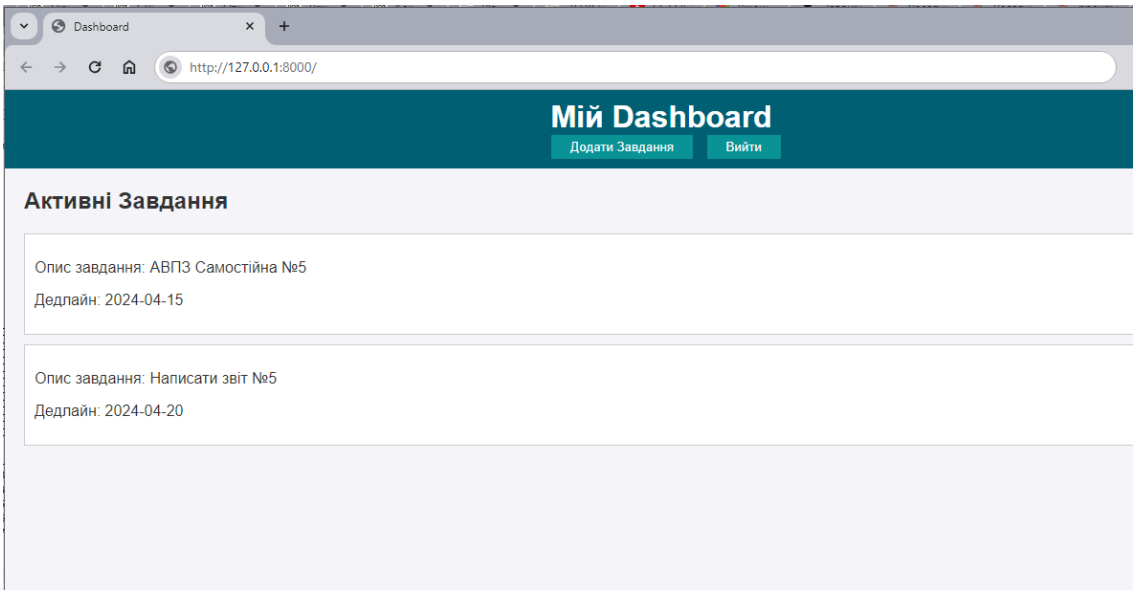


Рис. 3.6 Мій Dashboard

Друга форма – Список Завдань – це інтерфейс для додавання нових завдань. Тут студенти можуть ввести опис завдання та його дедлайн. Це спрощує процес введення нових завдань і забезпечує, що вся інформація про завдання буде в одному місці.

Рисунок 3.7 друга форма – Список Завдань – це інтерфейс для додавання нових завдань.

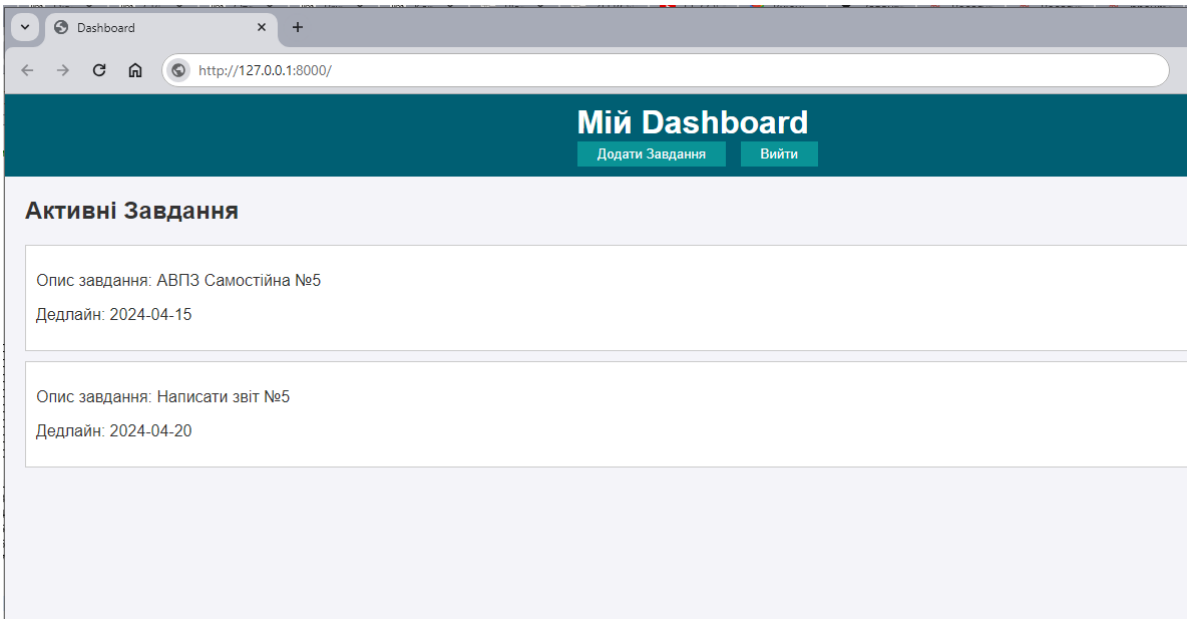


Рис. 3.7 Активні завдання

За допомогою Мій Dashboard студенти можуть переглядати всі свої активні завдання, а також додавати нові завдання або виходити з системи. Кожне завдання містить опис і дедлайн, що допомагає студентам організовувати свою роботу ефективніше.

Представлені екранні форми надають студентам зручний інтерфейс для планування та управління самостійною роботою.

4 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАСТОСУНКУ

4.1 Аналіз застосувань

Тестування програмного додатку описує методологію тестування, тестових кейсів та результатів пов’язаних з додатком для резервного копіювання та моніторингку резервних копій, використовуючи проект коду як зразок.

Методологія тестування яка прийнята для програми відповідає комплексному підходу. Кожен рівень тестування служить певній меті і допомагає виявити різні типи проблем або дефектів у програмному забезпеченні.

Тестування інтеграції з базою даних або файловою системою: перевірити чи коректно програма зчитує і записує дані в файлову систему або в базу даних.

Тестування інтеграції з зовнішніми сервісами: якщо програма інтегрується з хмарними сервісами для зберігання даних, важливо переконатися, що ці інтеграції працюють належним чином.

Приклади системного тестування.

Системне тестування здійснюється на рівні цілої системи і включає в себе перевірку програмного забезпечення в умовах, максимально наближених до реальних. Цей тип тестування оцінює систему її очікуваної функціональності, безпеки, надійності та взаємодії з іншими системами або компонентами.

Функціональне тестування: перевірка всіх функцій програми, таких як створення бекапів, відновлення даних, планування задач та ведення журналів подій.

Тестування виконання: оцінка продуктивності програми при різних обсягах даних або під високим навантаженням.

Тестування безпеки: перевірка системи на вразливості, що можуть вплинути на конфіденційність, цілісність або доступність даних.

Тестування сумісності: забезпечення того, що програма працює правильно на різних платформах, операційних системах або в різних мережевих середовищах.

Автоматизація тестування: потрібно розглянути можливість автоматизації рутинних тестів, особливо для функціонального і регресійного тестування, це значно прискорить процес розробки та забезпечить більш стабільні релізи.

Використання тестових стендів: обов’язкове створення спеціальних тестових середовищ допомогає проводити інтеграційне та системне тестування без ризику для реальних даних або оперативних систем. Розглядаючи та використовуючи ці методи тестування, можемо забезпечити надійність та якість застосунку, мінімізувати помилки та надати користувачам високоякісний продукт.

4.2 Розробка тест кейсів

Тестові випадки (тест-кейси) були створені для перевірки основних функцій розробленого застосунку для підтримки самостійної роботи здобувача вищої освіти з використанням мови програмування Python. Кожен тест-кейс спрямований на перевірку відповідності вимогам системи та її коректної роботи у різних сценаріях використання.

Залежно від очікуваного результату, тест-кейси поділяються на дві категорії:

- Позитивні тест-кейси використовують лише коректні дані для перевірки правильності виконання функцій застосунку. Вони підтверджують, що програма вірно обробляє правильні вхідні дані та повертає очікувані результати.
- Негативні тест-кейси включають як коректні, так і некоректні дані (з принаймні одним некоректним параметром). Ці тест-кейси призначені для перевірки поведінки системи в умовах, коли вхідні дані не відповідають очікуванням. Вони перевіряють, як система обробляє помилкові ситуації та чи забезпечує вона належні повідомлення про помилки користувачеві.

Кожен тест-кейс має структуровану форму з наступними розділами:

- Action (дія або послідовність дій, що виконуються під час тестування),
- Expected Result (очікуваний результат),
- Test Result (фактичні результати роботи функції, часто позначені як "passed/failed/blocked").

Наступні тест-кейси призначені для тестування ключових функцій розробленого застосунку. Результати тестування наведені у таблиці 4.1.

Таблиця 4.1.

№	OK/NOK	Дії	Очікуваний результат
1	OK	1. Відкрити сторінку для створення нового завдання. 2. Ввести опис завдання та встановити дедлайн. 3. Додати необхідні документи або матеріали. 4. Натиснути кнопку "Зберегти".	Завдання створюється та зберігається у системі.
2	OK	1. Відкрити сторінку для перегляду списку завдань. 2. Відкрити деталі конкретного завдання.	Список завдань відображається коректно. Відображаються всі деталі та прикріплені матеріали.
3	OK	1. Відкрити сторінку для завантаження виконаного завдання.	Завантажене завдання

		2. Прикріпити необхідні коментарі або додаткові матеріали. 3. Натиснути кнопку "Завантажити".	зберігається у системі.
4	ОК	1. Відкрити сторінку для перегляду завантажених завдань. 2. Відкрити деталі конкретного завантаженого завдання.	Завантажені завдання відображаються коректно. Показуються всі деталі та прикріплені коментарі.
5	ОК	1. Відкрити сторінку для оцінювання завантаженого завдання. 2. Встановити оцінку та за необхідності додати коментар. 3. Натиснути кнопку "Зберегти оцінку".	Оцінка завантаженого завдання зберігається у системі.
6	ОК	1. Відкрити сторінку для перегляду оцінок завантажених завдань.	Оцінки відображаються коректно для кожного завантаження.
7	ОК	1. Відкрити сторінку для редагування завдання.	Можливість внесення змін до

		2. Відредагувати опис, додати/видалити матеріали або змінити дедлайн. 3. Натиснути кнопку "Зберегти".	існуючого завдання. Зміни зберігаються коректно у системі.
8	ОК	1. Відкрити сторінку для видалення завдання. 2. Підтвердити видалення через діалогове вікно.	Можливість видалення завдання з системи. Завдання успішно видаляється з системи.

ВИСНОВКИ

В процесі виконання кваліфікаційної роботи, було досягнуто наступних результатів:

1. Проаналізувано потреби студентів у підтримці самостійної роботи.
2. Визначено переваги та недоліки існуючих програмних засобів.
3. Розроблено функціональні та нефункціональні вимоги до застосунку для самостійної роботи здобувача.
4. Спроектувано архітектуру програмного забезпечення, визначено класи та методи для створення застосунку.
5. Розроблено користувацький інтерфейс для взаємодії студентів з програмним забезпеченням.
6. Розроблено застосунок для підтримки самостійної роботи здобувачів вищої освіти.
7. Проведено модульне та інтеграційне тестування програмного забезпечення.
8. Робота пройшла апробацію:

Левчик О.І., Аверічев І.М. Розробка застосунку для підтримки самостійної роботи здобувача вищої освіти на мові PYTHON. Всеукраїнська науково-технічна конференція «Сучасний стан та перспективи розвитку IoT», 18 квітня 2024 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2024. С.96-97.

Левчик О.І., Аверічев І.М. Інтеграція сучасних інновацій в освітні навчальні платформи. Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях», 24 квітня 2024 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.:ДУІКТ, 2024. С.44-46.

На основі отриманих даних та досліджень, розроблено застосунок для підтримки самостійної роботи студентів, який враховує всі проаналізовані потреби та можливості, щоб стати ефективним інструментом у навчанні.

ПЕРЕЛІК ПОСИЛАНЬ

1. Левчик О.І., Аверічев І.М. Розробка застосунку для підтримки самостійної роботи здобувача вищої освіти на мові PYTHON. Всеукраїнська науково-технічна конференція «Сучасний стан та перспективи розвитку IoT», 18 квітня 2024 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2024. С.96-97.

2. Левчик О.І., Аверічев І.М. Інтеграція сучасних інновацій в освітні навчальні платформи. Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях», 24 квітня 2024 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.:ДУІКТ, 2024. С.44-46.

3. Балабанов О. С. Аналітика великих даних: принципи, напрямки і задачі (огляд). *Проблеми програмування*. 2019. No 2. С. 47-68. URL: http://dspace.nbuv.gov.ua/xmlui/bitstream/handle/123456789/161487/05-Balabanov .pdf;jsessionid=D60607206FD710CE8DBEAE17F2658C87?sequence=1.

4. Barney N. Amazon Web Services [Електронний ресурс] / Nick Barney // Techtarget. – 2022. – Режим доступу до ресурсу: https://www.techtarget.com/searchaws/definition/Amazon-Web-Services.

5. T.K. B. Machine learning algorithms for social media analysis: A survey [Електронний ресурс] / B. T.K., R. Chandra Sekhara, B. Annushree // ScienceDirect. – 2021. – Режим доступу до ресурсу: https://www.sciencedirect.com/science/article/abs/pii/S1574013721000356.

6. Документація «AMQP (Advanced Message Queuing Protocol)» [Електронний ресурс] - Режим доступу: https://www.amqp.org/ 18.10.2023

7. Документація «RabbitMQ»:[Електронний ресурс] - Режим доступу: https://www.rabbitmq.com/ 18.10.2023

8. Article «Why Google Stores Billions of Lines of Code in a Single Repository»: - Rachel Potvin and Josh Levenberg [Електронний ресурс] - Режим доступу: https://dl.acm.org/doi/pdf/10.1145/2854146 18.10.2023

9. Клієнт-серверна архітектура [Електронний ресурс] // QATestLab. – 2020. – Режим доступу до ресурсу: https://training.qatestlab.com/blog/technical articles/client-server-architecture/.

10. Агрегація в MongoDB: зменшення сукупного трубопроводу та карти [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: https://uk.myservername.com/aggregation-mongodb.

11. The top programming languages [Електронний ресурс] // GitHub. – 2022. – Режим доступу до ресурсу: https://octoverse.github.com/2022/top programming-languages.

12. "Шаблони проектування сервісів: Фундаментальні рішення для SOAP/WSDL та RESTful веб-сервісів" Роберт Деньо

13. Рейтинг мов програмування 2023 [Електронний ресурс] // Редакція DOU. – 2023. – Режим доступу до ресурсу: https://dou.ua/lenta/articles/language rating-2023/.

14. "Data Structures and Algorithms in Python", Michael T. Goodrich and Roberto Tamassia, 4th edition, Wiley, 2022.

15. "OSPF: The Basics", Juniper Networks, 2023.

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка застосунку для підтримки самостійної роботи здобувача вищої освіти з використанням мови програмування Python

Виконав студент 4 курсу
групи ПД-43
Левчик Олег Ігорович
Керівник роботи
к.е.н., доцент кафедри ІПЗ Аверічев Ігор Миколайович

Київ – 2024

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи** – підтримка самостійної роботи здобувача вищої освіти за використання WEB-застосунку.
- **Об'єкт дослідження** – самостійна робота здобувача вищої освіти.
- **Предмет дослідження** – застосунок для підтримки самостійної роботи здобувача вищої освіти.

2

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Проаналізувати потреби студентів у підтримці самостійної роботи.
2. Визначити переваги та недоліки існуючих програмних засобів.
3. Розробити функціональні та нефункціональні вимоги до застосунку для самостійної роботи здобувача.
4. Спроектувати архітектуру програмного забезпечення, визначити класи та методи для створення застосунку.
5. Розробити користувацький інтерфейс для взаємодії студентів з програмним забезпеченням.
6. Розробити застосунок для підтримки самостійної роботи здобувачів вищої освіти.
7. Провести модульне та інтеграційне тестування програмного забезпечення.

3

АНАЛІЗ АНАЛОГІВ

Критерій порівняння	Застосунок для самостійних робіт	Moodle	Google Classroom
Відкрите API	Так	Обмежені можливості	Обмежені можливості
Інтеграція з Telegram	Так	Немає	Немає
Реєстрація користувачів	Через Telegram бот або через веб-інтерфейс	Через веб-інтерфейс	Через веб-інтерфейс
Додавання завдань	Через Telegram бот або через веб-інтерфейс	Через веб-інтерфейс	Через веб-інтерфейс
Редагування завдань	Через Telegram бот або через веб-інтерфейс	Через веб-інтерфейс	Через веб-інтерфейс
Видалення завдань	Через Telegram бот або через веб-інтерфейс	Через веб-інтерфейс	Через веб-інтерфейс
Відображення розкладу	Немає	Так	Так
Відправлення нагадувань	Через Telegram бот	Так	Так
Пристрої	Website	Website	Android, IOS, Website
Створення курсу	Доступ на рівні користувача	Доступ на рівні адміністратора	Доступ на рівні користувача
Оцінки	Список оцінок, середня оцінка	Формуються списки з оцінками	Список оцінок для викладача

4

ВИМОГИ ДО ЗАСТОСУНКУ

Функціональні:

1.Реєстрація та аутентифікація користувачів.

2. Додавання, редагування та видалення завдань через Telegram бот.

3. Відображення списку завдань користувача через веб-інтерфейс.

4. Відображення розкладу завдань у вигляді календаря.

5. Відправлення нагадувань про дедлайни завдань.

6. Управління ролями користувачів (студент, викладач).

Нефункціональні:

1. Підтримка різних операційних систем (Windows, Mac, Linux).

2. Захист персональних даних користувачів.

3. Використання офіційного Telegram API для взаємодії з ботом.

4. Інтуїтивно зрозумілий та зручний інтерфейс користувача.

5

ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ

6

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

PyCharm

JavaScript

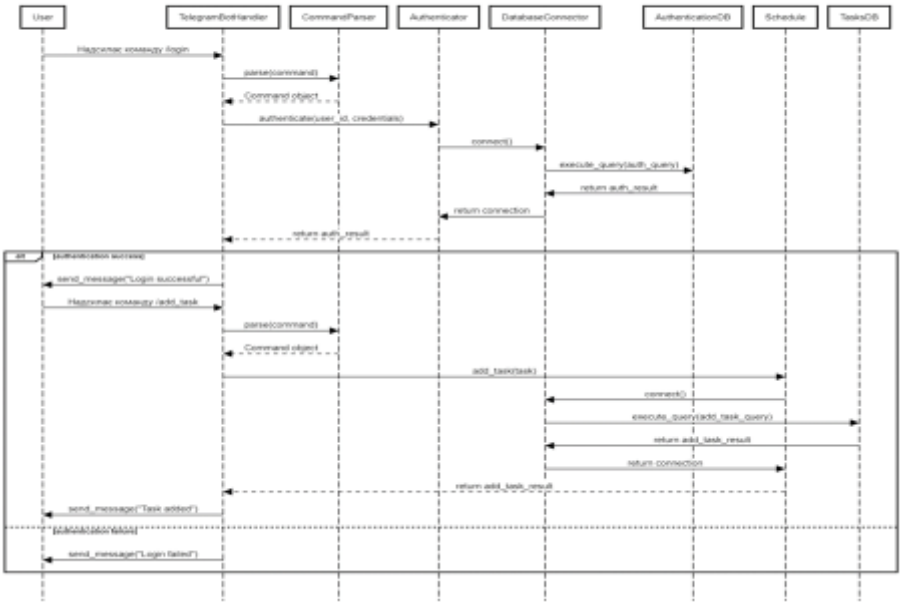
HTML CSS

SQLite

python

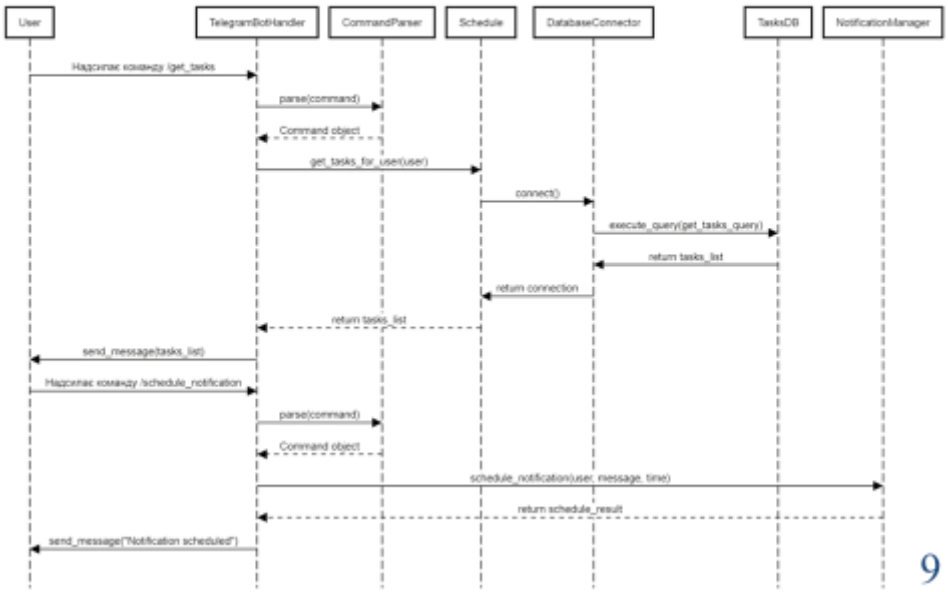
7

ДІАГРАМА ПОСЛІДОВНОСТІ
АУТЕНТИФІКАЦІЯ ТА ДОДАВАННЯ ЗАВДАННЯ



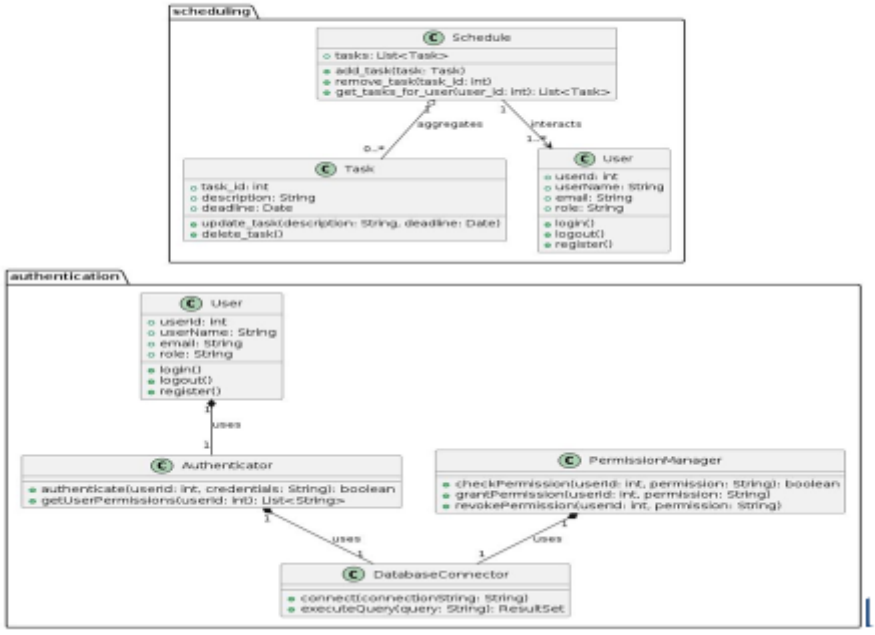
8

ДІАГРАМА ПОСЛІДОВНОСТІ
ОТРИМАННЯ ЗАВДАНЬ ТА ПЛАНУВАННЯ СПОВІЩЕНЬ



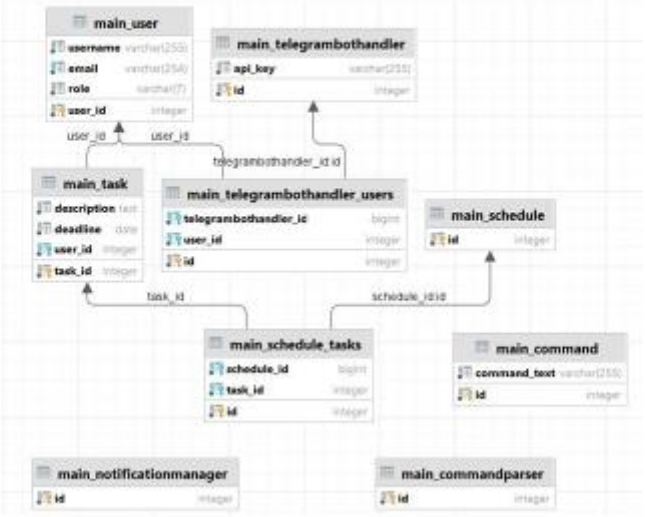
9

ДІАГРАМИ КЛАСІВ



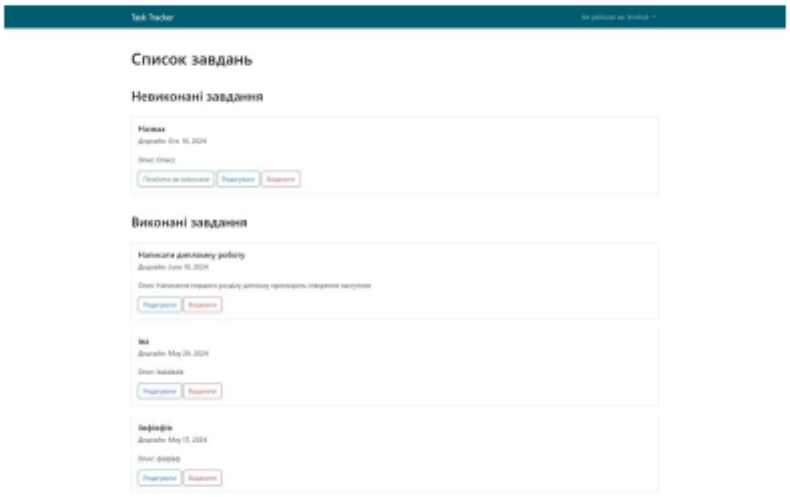
10

СХЕМА БАЗИ ДАННИХ



11

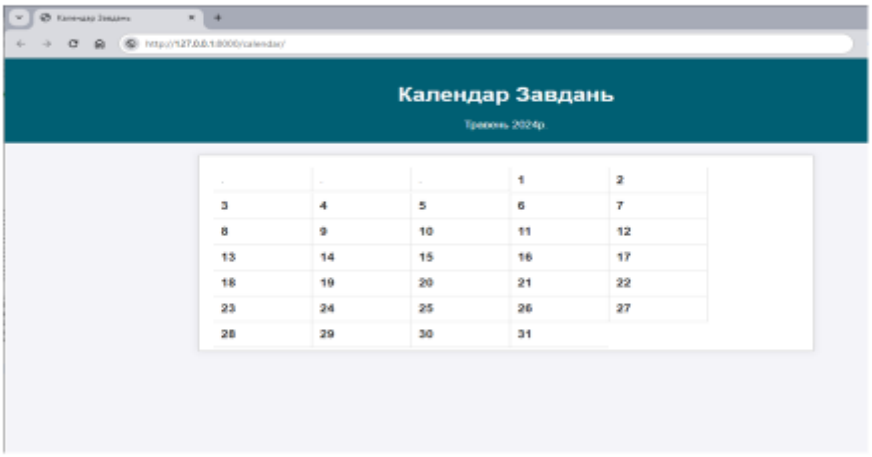
ЕКРАННІ ФОРМИ



Список завдань

12

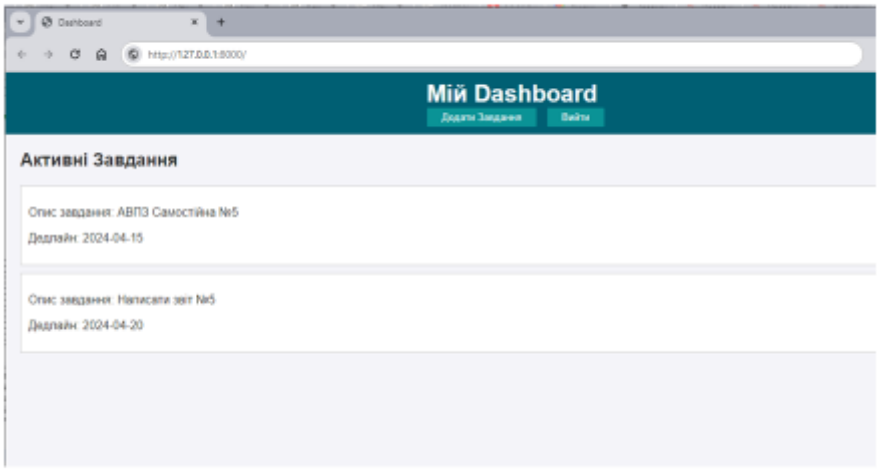
ЕКРАННІ ФОРМИ



Календар завдань

13

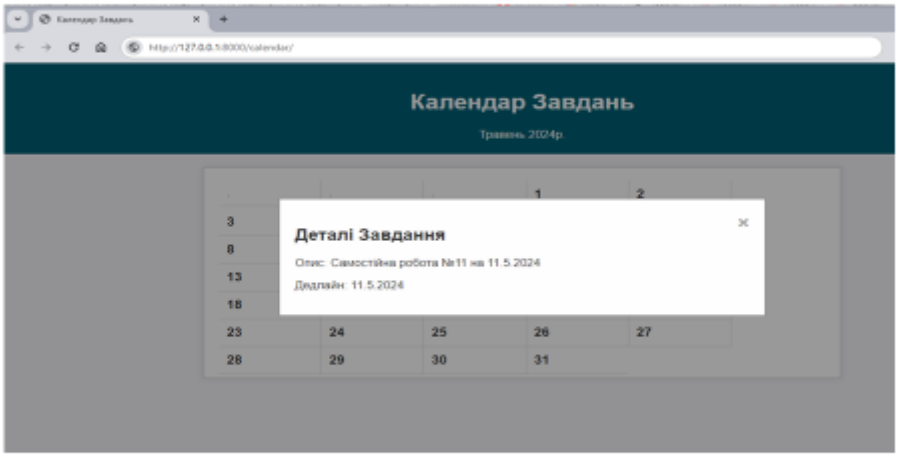
ЕКРАННІ ФОРМИ



Активні завдання

14

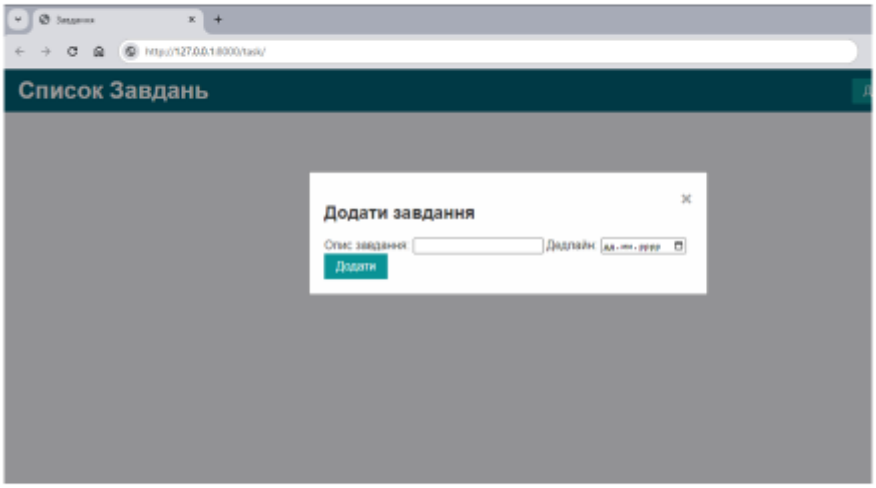
ЕКРАННІ ФОРМИ



Деталі завдання

15

ЕКРАННІ ФОРМИ



Додати завдання

16

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

- Левчик О.І., Аверічев І.М. Розробка застосунку для підтримки самостійної роботи здобувача вищої освіти на мові PYTHON. Всеукраїнська науково-технічна конференція «Сучасний стан та перспективи розвитку IoT», 18 квітня 2024 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2024. С.96-97.
- Левчик О.І., Аверічев І.М. Інтеграція сучасних інновацій в освітні навчальні платформи. Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях», 24 квітня 2024 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.:ДУІКТ, 2024. С.44-46.

17

ВИСНОВКИ

1. Проаналізовано потреби студентів у підтримці самостійної роботи.
2. Визначено переваги та недоліки існуючих програмних засобів.
3. Розроблено функціональні та нефункціональні вимоги до застосунку для самостійної роботи здобувача.
4. Спроектовано архітектуру програмного забезпечення, визначено класи та методи для створення застосунку.
5. Розроблено користувацький інтерфейс для взаємодії студентів з програмним забезпеченням.
6. Розроблено застосунок для підтримки самостійної роботи здобувачів вищої освіти.
7. Проведено модульне та інтеграційне тестування програмного забезпечення.