Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Програмні засоби для удосконалення систем керування навчанням»

Виконав: студент 2-го курсу
групи ПІ-22мз спеціальності
121 – Інженерія програмного забезпечення
_____Войтенко Д.О._____
Керівник: д.т.н., проф. Романюк О.Н.
«_7_» ___червня_____ 2024 р.
Опонент: д.т.н., проф. Лужецький В.А.
«_7_» ___червня_____ 2024 р.

Допущено до захисту
Завідувач кафедри ПЗ
д.т.н., проф. О. Н. Романюк
«10» _____червня__ 2024 р.

ВНТУ – 2024

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти II-й (магістерський)
Галузь знань 12 – Інформаційні технології
Спеціальність 121 – Інженерія програмного забезпечення
Освітньо-професійна програма – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
д.т.н., професор Романюк О.Н.
_____
« 12 » березня 2024 р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Войтенко Денис Олександрович

1. Тема роботи – Програмні засоби для удосконалення систем керування навчанням.

Керівник роботи: Романюк Олександр Никифорович, д.т.н., завідувач кафедри ПЗ, затверджені наказом вищого навчального закладу від « 11 » березня 2024р. № 81.

2. Строк подання студентом роботи 3 червня 2024р.

3. Вихідні дані до роботи: мова програмування - Rust, бази навчальної роботи, типи сервісів Github/Google Calendar/Google Meet, тип подачі даних - конфігураційні файли.

4. Зміст текстової частини: вступ, обґрунтування розробки, , постановка задачі дослідження, розробка системи керування навчанням, тестування додатку, економічна частина, висновки.

5. Перелік графічного матеріалу: діаграма сутностей, структурна схема додатку, структура даних додатку.

6. Консультанти розділів роботи

11.06.2024 09:36

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Романюк О.Н., д.т.н., професор кафедри ПЗ	11.03.24	03.06.24
5	Кавецький В.В., к.е.н., доцент кафедри ЕПВМ	26.04.24	10.05.24

7. Дата видачі завдання 11 березня 2024р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Обґрунтування розробки	12.03.2024 – 28.03.2024	
2	Розробка методу автоматичної оцінки успішності студентів	29.03.2024 – 11.03.2024	
3	Розробка системи керування навчанням	12.03.2024– 12.04.2024	
4	Тестування роботи системи	13.04.2024– 05.06.2024	
5	Економічна частина	26.04.2024 - 10.05.2024	

Студент Войтенко Д.О.

Керівник магістерської кваліфікаційної роботи Романюк О.Н.

11.06.2024 09:36

ВСТУП

Навчання студентів є складним процесом, яке вимагає постійного нагляду та перевірки виконаних завдання. Студенти мають не тільки отримати завдання а й здавати їх у певних часових рамках та виконувати їх відповідно до вимог, які проходять перевірку викладачем. Також необхідно мати інструменти покарання студентів, які не вистигають у визначені терміни чи виконують завдання не згідно виставлених вимог. Використання систем керування навчанням має полегшити дані складнощі при організації навчання.

Використання систем керування навчанням (LMS) для керування навчальними процесами може значно полегшити як облік студентів групи та й їх оцінювання. Системи керування навчанням можуть зменшити навантаження на викладача та дозволить тому приділяти більше часу на покращення власних знань з відповідним ростом якості навчання.

Системи керування навчанням можуть вести облік результатів виконання студентами визначених завдань та й оброблювати їх. Наприклад після зібрання інформації про виконанні завдання програма може відобразити прогрес студентів у таблиці з розділом по пунктам програми та відповідним прогресом кожного студента.

Системи керування навчанням можуть бути посередником серед спілкуванням ментора зі студентом й аґрегувати у собі різні методи доставки повідомлень.

Системи керування можуть бути використанні для швидкого створення навчальної програми за певними параметрами. Також такі системи можуть надавати рекомендації щодо покращення навчальних програм і сприяти прийняттю кращих рішень.

Важливо також зазначити, що вони забезпечують доступність навчання з будь-якого місця та в будь-який час, що дозволяє студентам навчатися у зручний для них час. В цілому, системи керування навчанням виконують багатофункціональну роль у сучасному освітньому середовищі, допомагаючи підвищити якість та ефективність навчання.

Зв’язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Мета магістерської кваліфікаційної роботи полягає в покращенні процесу навчання за рахунок автоматизації одноманітних процесів та систематизації даних успішності студентів та покращення якості навчання.

Головні задачі дослідження є:

- провести аналіз існуючих систем керування навчанням, які використовуються навчальними закладами;
- запропонувати:
    - метод автоматичного оцінювання успішності студентів на основі часових обмежень та погодження від ментору;
    - метод автоматичного збору інформації про успішність студентів та відображення цієї статистики у зручному для користувача вигляді;
- розробити інтерфейс користувача та інтеграцію зі сторонніми сервісами;
- розробити додаток для показу результату збору інформації;
- провести експериментальні дослідження розроблених засобів.

Об’єкт дослідження – процес розробки системи керуванням навчанням.

Предмет дослідження є методи та засоби розробки системи керування навчанням.

Методи дослідження. У процесі розробки магістерської кваліфікаційної роботи було використано: теорія розробки інтерфейсів користувача, засоби розробки інтерфейсів користувача, методи оцінювання успішності студентів.

Наукова новизна отриманих результатів:

- уперше розроблено метод оцінювання успішності студентів, особливість якого полягає у використанні часових обмежень без

участі ментора у виставлені виконаних завдань кожного студента, що дало можливість покращити якість дистанційного навчання.

Практична цінність отриманих результатів. Практична цінність одержаних результатів полягає в тому, що на основі отриманих у магістерській кваліфікаційній роботі результатів розроблено алгоритми та програмну систему керування навчанням, яка може бути використана для автоматизації процесів навчання та покращення якості навчання за рахунок зменшення обов'язків ментора.

Особистий внесок здобувача. Усі наукові результати викладені у магістерській кваліфікаційній роботі, отримані автором особисто. У роботі [1], написаних у співавторстві, автору належить: метод автоматичного збору статистики успішності студентів.

Апробація роботи матеріали магістерської кваліфікаційної роботи доповідались на Десятій Всеукраїнській науково-практичній інтернет-конференції молодих вчених та студентів «Сучасні інформаційні системи та технології».

Публікації. За тематикою дослідження опубліковано 1 наукова праця у матеріалах конференції.

1 ОБҐРУНТУВАННЯ РОЗРОБКИ

1.1 Аналіз предметної галузі

Системи керування навчанням - це справжні мультитули для освітніх закладів. Вони розроблені для того, щоб оптимізувати кожний етап навчального процесу. Вони можуть адаптуватися до потреб кожного учня, надаючи персоналізовані матеріали та завдання. Також, вони забезпечують величезний обсяг аналітичних даних, які допомагають виявляти тренди та слабкі місця в навчанні [2].

Більше того, вони стимулюють взаємодію між студентами та викладачами через онлайн-форуми та чати. Завдяки автоматизації багатьох рутинних завдань, таких як оцінювання, вони звільняють викладачів від надмірної роботи і дозволяють їм більше уваги приділяти навчанню.

У 1980-х роках почали використовувати сучасні засоби телекомунікацій у освіті. Комп'ютери стали важливим елементом повсякденного життя вищих навчальних закладів, а також інструментом для навчання студентів. Комп'ютерне навчання спрямовувалося на інтеграцію технічних та освітніх засобів. Пізніше тренд змінився на відео-комунікацію, внаслідок чого Університет Хьюстона вирішив проводити телевізійні заняття для своїх студентів приблизно 13-15 годин на тиждень. Заняття відбувалися у 1953 році, а у 1956 році Робін Маккіннон Вуд і Гордон Паск випустили першу систему адаптивного навчання для корпоративного середовища SAKI. Ідея автоматизації навчальних операцій також надихнула експертів Університету Іллінойсу на розробку їхньої програми "Програмована логіка для автоматизованих навчальних операцій" (PLATO), яка дозволяла користувачам обмінюватися контентом незалежно від їхнього місця знаходження. Протягом періоду між 1970 і 1980 роками навчальні заклади швидко розглядали ідею комп'ютеризації курсів, включаючи Західний Інститут Поведінкових Наук з Каліфорнії, який представив перший акредитований онлайн-навчальний ступінь [3].

Історія застосування комп'ютерів в освіті наповнена широко описованими термінами, такими як комп'ютерне управління навчанням (CMI), інтегровані системи навчання (ILS), комп'ютерне навчання (CBI), комп'ютерно-підтримане навчання (CAI) і комп'ютерне навчання (CAL). Ці

терміни описують програми типу "вправи та вправи", більш складні навчальні посібники та більш індивідуалізоване навчання відповідно. У теперішній час термін використовується для опису різних освітніх комп'ютерних застосунків. FirstClass від SoftArc, що використовувався Великобританським Відкритим Університетом у 1990-х і 2000-х роках для надання онлайн-навчання в Європі, був одним з перших Інтернет-орієнтованих LMS [4].

Перша повнофункціональна Система Управління Навчанням (LMS) мала назву EKKO і була розроблена та випущена Норвезькою мережею дистанційної освіти NKI в 1991 році. Три роки потому Мережа Навчання NB з Нью-Брансвіка представила подібну систему, призначену для навчання на основі DOS і виключно для бізнес-студентів [5].

Класичні випадки використання систем управління навчанням (LMS) охоплюють широкий спектр ситуацій, від традиційного освітнього середовища до корпоративного навчання та онлайн-курсів [6].

У навчальних установах, таких як школи, коледжі та університети, LMS використовуються для управління курсами та матеріалами, спільної роботи між студентами та викладачами, а також для оцінювання та ведення журналу. Вони дозволяють викладачам створювати і розповсюджувати навчальний контент, надавати завдання та тести, а також вести облік успішності студентів. Студенти, у свою чергу, можуть взаємодіяти з матеріалами, виконувати завдання та спілкуватися з однокурсниками та викладачами через форуми та чати.

У сфері корпоративного навчання LMS використовуються для навчання персоналу, впровадження нових працівників та навчання на професійних курсах. Вони дозволяють компаніям створювати та впроваджувати навчальні програми, вести моніторинг прогресу навчання та оцінювати результативність працівників [7].

Онлайн-курси та тренінги також широко використовують LMS для надання доступу до навчального матеріалу, організації завдань та тестів, а також для взаємодії між учасниками курсу та інструкторами через форуми та чати.

У кожному з цих випадків LMS виступає як центральний інструмент управління навчальним процесом, забезпечуючи доступ до ресурсів, спільну

роботу та взаємодію між учасниками навчання, а також моніторинг та оцінку результатів.

Основні функції LMS включають в себе:

- Управління користувачами: LMS дозволяють адміністраторам створювати, керувати та редагувати облікові записи користувачів, таких як викладачі і студенти;
- навчальний контент: LMS дозволяють завантажувати, організовувати та надавати доступ до різноманітного навчального контенту, такого як тексти, відео, аудіо, інтерактивні завдання тощо;
- управління курсами: Вони надають засоби для створення та організації навчальних курсів, встановлення графіків та навчальних програм, призначення завдань та тестів;
- оцінювання та звітність: LMS дозволяють викладачам створювати та оцінювати завдання, проводити тести та відстежувати прогрес студентів. Вони також надають інструменти для аналізу даних та створення звітів;
- співпраця та комунікація: LMS надають можливості для спілкування між учасниками навчального процесу, такі як форуми обговорень, чати, електронна пошта тощо;
- моніторинг та аналітика: Вони дозволяють адміністраторам та викладачам відстежувати активність користувачів, успішність студентів, а також аналізувати дані для поліпшення навчального процесу [8].

1.2 Порівняння аналогів

Moodle (рис. 1.1) - це відкрита система управління навчанням, яка користується великою популярністю в освітній сфері. Вона надає широкий спектр можливостей, включаючи створення курсів, завдань, тестів, форумів та блогів. Однією з основних переваг є безкоштовність та активна спільнота користувачів, яка надає підтримку та розвиток системи. Однак, для використання Moodle може знадобитися певний час для оволодіння інтерфейсом та налаштуванням.

Переваги:

- відкритий вихідний код, що дозволяє редагувати та адаптувати платформу під власні потреби;

- багатофункціональність, включаючи форуми, завдання, онлайн-тести, блоги тощо;
- широка спільнота користувачів та розвинена документація.
- Недоліки:
- потребує технічних знань для налаштування та впровадження;
- інтерфейс може здаватися складним для новачків.

Рисунок 1.1 - Логотип сервісу ” Moodle”

Canvas (рис 1.2) - це інша популярна система управління навчанням, яка відома своїм простим та інтуїтивно зрозумілим інтерфейсом. Вона надає багатий функціонал для навчання в онлайн-середовищі, включаючи можливості відеоконференцій та мобільних додатків. Canvas також має розвинуту систему підтримки користувачів. Однак, вартість використання Canvas може бути високою для деяких користувачів.

Переваги:

- інтуїтивний та легкий у використанні інтерфейс;
- широкий спектр функціональних можливостей, включаючи інтеграцію з зовнішніми сервісами;
- активна технічна підтримка.
- Недоліки:
- Високі витрати на ліцензію;
- обмежена можливість налаштування під конкретні потреби.

Рисунок 1.2 - Логотип сервісу ”Canvas”

Blackboard (рис. 1.3) - одна з найстаріших та найбільш відомих систем управління навчанням, яка має велику базу клієнтів у всьому світі. Вона пропонує розширений набір інструментів для віддаленого навчання та співпраці, включаючи функції відеоконференцій та інтерактивних вправ. Однак, інтерфейс користувача Blackboard може бути менш інтуїтивно зрозумілим порівняно з іншими системами.

Переваги:

- Розширені можливості для інтерактивного навчання, включаючи відео та аудіо матеріали, онлайн-тести тощо;
- інтеграція з іншими платформами та сервісами.

Недоліки:

- Специфічний та складний інтерфейс для користувачів;
- обмежена гнучкість у налаштуванні та адаптації.

Рисунок 1.3 - Логотип сервісу ”Blackboard Learn”

Google Classroom (рис. 1.4) - це проста та зручна система управління навчанням, яка інтегрується з іншими сервісами Google, такими як Gmail та Google Drive. Вона особливо підходить для користувачів, які вже використовують інші продукти Google. Однак, в порівнянні з іншими системами, вона може бути обмеженою у функціональності.

Переваги:

- легко інтегрується з іншими Google-продуктами, такими як Gmail та Google Drive;
- простий та зрозумілий для використання вчителями та учнями.

Недоліки:

- може бути обмеженим у функціональності порівняно з іншими системами;
- менші можливості для відстеження прогресу та оцінювання.

Рисунок 1.4 - Логотип сервісу ”Google Classroom”

Schoology (рис.1.5) - це інша популярна система управління навчанням з потужним набором інструментів для співпраці та відстеження прогресу. Вона інтегрується з багатьма сторонніми програмами та сервісами, що полегшує використання в різних навчальних середовищах. Однак, вартість використання Schoology може бути високою для деяких користувачів, а інтерфейс може вимагати трохи часу для оволодіння.

Переваги:

- Простий у використанні інтерфейс, що робить його популярним серед вчителів та студентів;
- можливість створення інтерактивних завдань, відеоуроків та обговорень;
- інтеграція з Google Apps for Education та Microsoft Office 365.

Недоліки:

- Обмежена функціональність порівняно з іншими LMS;
- є обмеження на безкоштовному тарифі, що може призвести до великих витрат для шкіл або навчальних закладів з великою кількістю користувачів.

Рисунок 1.5 - Логотип сервісу ”Schoology”

Результати порівняння аналогів зведено в табл. 1.1.

Таблиця 1.1 - порівняння аналогів систем керування навчанням.

Критерій	Schoology	Blackboard Learn	Canvas	Moodle	Google Classroom	Власна розробка
Зручний інтерфейс	1	0	1	1	1	1
Відкритий вихідний код	0	0	0	1	0	1
Безкоштовна	0	1	0	1	1	1
Інтеграція з стороннім и сервісами	1	1	1	0	0	1
Легкість до змін	0	0	1	1	0	1
Загалом	2	2	3	4	2	5

Відповідно до таблиці з порівняльними характеристиками розробка власної системи керування навчанням є доцільною. Отримана система може позбавити від недоліків, які мають наявні рішення та полегшити інтеграцію з іншими сервісами.

1.3 Аналіз методів розв’язання поставленої задачі

Для розв’язання поставленої задачі існує декілька рішення. Створення системи, яка буде складатися з веб-сторінки та веб-сервера на якій користувач може керувати створенням контенту. Такий підхід вимагає великої кількості ресурсів на розробку.

Альтернативним підходом є написання додатку та інтеграцією його в існуючий сервіс. Зокрема можна використати сервіс Github на якому студенти будуть виконувати практичні заняття та знайомитися з теоретичними матеріалами, а для проведення занять використати Google Meet. Такий підхід

дозволить зменшити вартість проекту та використати сервіси добре знайомі студентам та менторам.

Врахувавши переваги й недоліки цих методів було вирішено використати метод розробки системи керування навчанням за допомогою інтеграції у сторонні сервіси.

1.4 Вибір мови програмування

Вибір мови програмування - це рішення, яке може суттєво вплинути на успіх проекту. Це як вибір інструменту: правильний інструмент допомагає виконати завдання швидше та ефективніше. Кожна мова має свої особливості та переваги, тому важливо розуміти потреби проекту та можливості кожної мови перед прийняттям рішення.

Також, вибір мови програмування - це також компроміс між потребами проекту, навичками розробників та екосистемою інструментів. Деякі мови можуть бути кращими для певних завдань або доменів, тоді як інші можуть мати більшу підтримку та розвиток інструментів.

Розглянемо декілька популярних мов.

Python, його історія починається у кінці 1980-х років, коли Гвідо ван Россум почав розробку цієї мови програмування. Вона була задумана як проста, легка для вивчення та читання мова, яка б могла використовуватися для широкого спектру завдань. У лютому 1991 року вийшла перша версія Python, що вже мала деякі з основних характеристик, які роблять мову популярною до цього дня [9].

Щодо характеристик, Python відомий своєю простотою та зрозумілістю синтаксису, який нагадує звичну англійську мову. Це робить його особливо привабливим для початківців у програмуванні та широкого кола фахівців. Крім того, Python є мовою з високим рівнем абстракції, що дозволяє розробникам зосередитися на розв'язанні проблеми, а не на деталях реалізації.

Однією з ключових переваг Python є його широка екосистема бібліотек та фреймворків, які допомагають вирішувати різноманітні завдання. Наприклад, бібліотека NumPy забезпечує підтримку для чисельних обчислень, а фреймворк Django - для веб-розробки. PyTorch для роботи з нейронними мережами. Крім того, Python має велику та активну спільноту користувачів, яка надає підтримку, допомогу та внесення вкладу у подальший розвиток мови.

Проте, як і у кожної мови програмування, у Python є й свої недоліки. Один з них - це швидкодія. Python, будучи мовою з високим рівнем абстракції, може мати обмежену швидкодію порівняно з мовами низького рівня, такими як C++ або Java. Крім того, Python може бути менш підходящим для деяких завдань, де вимагається велика швидкодія або виконання в реальному часі[10].

C# та його історія розпочалася у 1990-х роках, коли Microsoft вирішила розробити нову мову програмування для роботи з платформою .NET Framework. Робота над мовою почалася у 1999 році, а перша версія C# (C Sharp) була випущена у 2000 році разом з .NET Framework 1.0. З тих пір мова постійно розвивалася, і нові версії випускалися разом з оновленнями платформи .NET [11].

Однією з головних характеристик C# є його схожість з мовою програмування Java. C# була розроблена як мова, що поєднує в собі зручність та продуктивність Java з можливостями Microsoft .NET Framework. Синтаксис мови C# легкий для вивчення та розуміння, що робить його популярним серед розробників різних рівнів навичок.

Однією з важливих переваг C# є його широкий функціональний діапазон. Він може бути використаний для розробки різноманітних додатків, від веб-сайтів і мобільних додатків до десктопних програм та ігор. Більш того, завдяки інтеграції з платформою .NET Framework, C# забезпечує доступ до широкої бібліотеки класів та інструментів для розробки програмного забезпечення [12].

Проте, наявні й деякі недоліки у використанні C#. Одним з них є те, що він більш пов'язаний з платформою Windows, тому розробка крос-платформених додатків може бути складнішою. Крім того, хоча C# і забезпечує високий рівень продуктивності та швидкодії, він може бути менш ефективним для деяких видів додатків порівняно з іншими мовами, такими як C++ або Rust.

У підсумку, C# є потужною та універсальною мовою програмування, яка забезпечує широкі можливості для розробки різноманітних програмних продуктів.

Історія мови програмування C++ почалася у 1979 році, коли Бьярне Страуструп почав розробку цієї мови у Bell Labs. Вона є розширенням мови програмування C з додаванням об'єктно-орієнтованого програмування та інших нововведень. Перша офіційна версія мови була випущена у 1985 році. З

того часу C++ став однією з найпопулярніших мов програмування і використовується для розробки широкого спектру програмного забезпечення, від операційних систем до ігор та вбудованих систем [13].

Однією з головних переваг C++ є його висока швидкодія та ефективність. Мова дозволяє розробникам отримати максимальну продуктивність з використанням мінімальних ресурсів комп'ютера. Крім того, C++ має багато вбудованих бібліотек та інструментів, що дозволяє розробникам швидко створювати складні програми.

Проте, C++ також має свої недоліки. Один з них - це складність вивчення та використання. Синтаксис мови може бути важким для новачків, а деякі концепції, такі як управління пам'яттю, можуть бути складними для розуміння. Крім того, C++ вимагає від розробників більшої уваги до деталей та може бути більш схильним до помилок, зокрема пов'язаних з управлінням пам'яттю та вказівниками.

Ще одним недоліком C++ є його обмежена крос-платформенність порівняно з іншими мовами, такими як Java або Python. Хоча існують засоби для розробки крос-платформених додатків на C++, такі як Qt або Boost, це може бути складніше порівняно з іншими мовами [14].

У підсумку, хоча C++ є потужною та ефективною мовою програмування, що забезпечує високу швидкодію та контроль над ресурсами, вона також має свої недоліки, зокрема складність вивчення та використання, а також обмежену крос-платформенність.

Java - це мова програмування, розроблена компанією Sun Microsystems (згодом придбана Oracle Corporation) у 1990-х роках. Вона була створена як мова, що працює на будь-якій платформі, тобто "write once, run anywhere" (один раз написати, запускати будь-де), завдяки використанню віртуальної машини Java (JVM) [15].

Однією з головних переваг Java є її крос-платформенність. Код, написаний на Java, може запускатися на будь-якій платформі, на якій є встановлена відповідна версія JVM, що робить її ідеальним вибором для розробки програмного забезпечення, яке має бути доступним на різних операційних системах.

Крім того, Java є мовою з високим рівнем безпеки. Вона має вбудовані механізми безпеки, такі як контроль доступу та перевірка додатків під час

виконання (runtime verification), що дозволяє уникнути багатьох типів програмних помилок, таких як переповнення буфера.

Однак, серед недоліків Java можна відзначити те, що вона має великі вимоги до ресурсів системи. JVM може бути великим споживачем оперативної пам'яті та ресурсів процесора, що може вплинути на продуктивність програм. Крім того, запуск та завантаження програм на Java може бути повільним порівняно з іншими мовами програмування, оскільки JVM спочатку компілює програму у проміжний байт-код, а потім виконує його [16].

Ще одним недоліком Java є те, що вона не є такою гнучкою та експресивною, як деякі інші мови програмування, такі як Python або Ruby. Це може робити розробку програм трохи більшою та складнішою, особливо для невеликих проектів або прототипування.

У підсумку, Java є потужною та крос-платформеною мовою програмування з високим рівнем безпеки, але вона також має свої недоліки, зокрема великі вимоги до ресурсів системи та обмежену експресивність порівняно з іншими мовами.

Мова програмування Rust виникла у Mozilla Research у 2006 році як проект для розробки безпечного, швидкого та практичного програмного забезпечення. Офіційно випущена у 2010 році, Rust став популярним варіантом для розробки системного програмного забезпечення, вбудованих систем та високопродуктивних додатків [17].

Однією з ключових переваг Rust є його система типізації та підходи до безпеки. Мова пропонує інноваційні механізми, такі як власництво (ownership), вимірюване в позиційному часі управління пам'яттю, що дозволяють уникати численних класичних помилок програмування, таких як витік пам'яті, вказівники нульового вказівника та перегруження буфера.

Ще однією перевагою Rust є його висока швидкодія та ефективність. Мова володіє системою керування пам'яттю, яка дозволяє розробникам отримати максимальну продуктивність з мінімальними витратами пам'яті та ресурсів процесора.

Проте, при використанні Rust можуть виникати деякі недоліки. Одним з них є відносно великий поріг входження для новачків. Система типізації та підходи до безпеки, хоча і дуже потужні, можуть бути складними для вивчення та використання для тих, хто тільки починає знайомство з мовою.

Ще одним недоліком Rust є те, що, хоча вона пропонує високу продуктивність, її екосистема бібліотек та інструментів може бути менш розвиненою порівняно з деякими іншими мовами, такими як Python або Java. Це може зробити розробку певних типів програм менш швидкою або менш простою.

У підсумку, Rust є потужною та інноваційною мовою програмування, яка пропонує високу продуктивність та безпеку, але вона також може бути складною для новачків та має менш розвинену екосистему порівняно з іншими мовами.

Результати порівняння аналогів зведено в табл. 1.2.

Таблиця 1.2 - порівняння мов програмування.

Критерій	Python	Java	C++	C#	Rust
Швидкодія	0	0	1	0	1
Безпека даних	1	1	0	1	1
Керування пакунками	1	1	0	1	1
Кросплатформовість	1	1	0	1	1
Загалом	3	3	1	3	4

Для розробки додатку, який має працювати з великими об’ємами даних та мусить надійно оброблювати особисті дані користувачів потрібно обрати мову, яка є достатньо швидкою та не містить типових проблем з пам’ятаю як у додатках написані на C чи C++, які були причиною багатьох вразливостей різноманітних систем. У процесі порівняння вибір мови Rust є найбільший доцільним вибором для реалізації такої системи.

1.5 Постановка задачі

На основі технічного завдання та проведеного аналізу предметної галузі, доцільною є розробка системи керування навчанням. Необхідно реалізувати та інтегрувати систему, що буде використана для керування навчальними групами.

Потрібно провести аналіз існуючих інтерфейсів взаємодії системи з студентами та менторами. Проаналізувати доцільність створення власного інтерфейсу.

Потрібно розробити систему для керування навчанням та покращити взаємодію студентів та менторів. Також необхідно зменшити навантаження, яка лягає на ментора за рахунок системи.

Перевірити роботу систему, що відслідковує прогрес студентів та відображає зібрані дані у зручному та зрозумілому представлені як таблиці. Також перевірити на відсутність помилок та відповідність виставленим вимогам.

1.6 Висновок

У розділі було проаналізовано предметну галузь, проведено порівняльний аналіз аналогів, вибрано мову програмування та визначено головні задачі.

У результаті було підтверджено доцільність розробки програмного продукту.

2 РОЗРОБКА МЕТОДУ АВТОМАТИЧНОЇ ОЦІНКИ УСПІШНОСТІ СТУДЕНТІВ

2.1 Постановка вимог для системи керування навчанням

Концепція цього проекту полягає у створенні комплексної системи керування навчанням, що має за мету автоматизувати та оптимізувати процеси навчання в організації. Головною метою системи є полегшення управління навчальними процесами, від ведення реєстрації учасників та курсів до оцінювання та моніторингу успішності. Наприклад, система може включати в себе можливість реєстрації користувачів та надання їм доступу до необхідних курсів, створення та адміністрування навчальних матеріалів, а також ведення статистики проходження курсів та успішності учасників. Очікуваним результатом проекту є створення функціональної та ефективної системи, яка підвищить якість навчання та сприятиме розвитку студентів.

Ця система має мати можливість інтеграції з різними сервісами та платформами, що дозволяє здешевити процес створення та розгортання. Наприклад, вона може інтегруватися з існуючими системами управління контентом (CMS), електронними платформами для навчання (LMS), а також з різноманітними сервісами хмарних технологій. Це дозволить ефективно використовувати наявні ресурси та інфраструктуру, а також спростить і швидше реалізувати розгортання системи у реальному середовищі. Такий підхід сприятиме зниженню витрат на розробку та підтримку системи, а також забезпечить більш гнучку та масштабовану архітектуру.

2.2 Метод автоматичного оцінювання успішності студентів

Метод оцінювання студентів, який базується на вчасно виконаних завданнях у формі Pull Request на GitHub та отриманих схвалень від менторів групи, надає можливість студентам демонструвати свої навички та здібності у реальному середовищі розробки. В рамках цього методу студенти отримують конкретні завдання, які вони повинні виконати шляхом внесення відповідних змін у вихідний код проекту та створення Pull Request із запропонованими змінами.

Після того як студент завершує виконання завдання та створює Pull Request, ментор групи переглядає його та визначає, чи відповідають внесені

зміни вимогам завдання. Якщо так, ментор може схвалити Pull Request, підтверджуючи успішне виконання завдання. Важливо зауважити, що в цьому методі оцінювання не передбачається присвоєння студентам оцінок а лише письмові коментарі щодо їх роботи.

Основна перевага цього підходу полягає у тому, що він сприяє розвитку самостійності та відповідальності студентів. Студентам доводиться самостійно аналізувати завдання, шукати необхідну інформацію та шляхи його вирішення, що сприяє їхньому професійному зростанню та розвитку навичок. Крім того, відсутність оцінок дозволяє уникнути негативного впливу конкуренції та підвищує спроможність студентів співпрацювати та навчатися разом.

Отже враховуючи всі ці переваги, метод оцінювання студентів на основі вчасно виконаних завдань та отриманих схвалень від менторів групи є ефективним методо організації дистанційного навчання.

2.3 Розробка діаграми сутностей для системи керування навчанням

Діаграма сутностей-зв'язків (ER-діаграма) - це тип блок-схеми, який ілюструє, як "сутності", такі як люди, об'єкти або концепції, пов'язані між собою всередині системи. ER-діаграми найчастіше використовуються для проектування або відлагодки реляційних баз даних у галузях програмної інженерії, бізнес-інформаційних систем, освіти та досліджень. Вони також відомі як ERD або ER моделі і використовують визначений набір символів, таких як прямокутники, ромби, овали та з'єднуючі лінії, щоб зображувати взаємозв'язок між сутностями, відносинами та їх атрибутами. Вони відображають граматичну структуру, де сутності виступають у ролі іменників, а відносини - у ролі дієслів [18].

ER-діаграми пов'язані з діаграмами структури даних (DSDs), які акцентують на взаємозв'язках елементів всередині сутностей, а не на відносинах між сутностями самими. Також ER-діаграми часто використовуються разом з діаграмами потоку даних (DFDs), які уявляють потік інформації для процесів або систем.

Пітер Чен (відомий також як Пітер Пін-Шан Чен), на даний момент викладач в Карнегі-Меллонському університеті в Піттсбурзі, приписується розробці моделювання ER для проектування баз даних у 1970-х роках. Під час

роботи асистента-професора в Школі менеджменту MIT, він опублікував визначну статтю 1976 року під назвою «Модель сутності-відношення: до єдиної точки зору на дані».

У більш широкому контексті зображення взаємозв'язку речей має відомість щонайменше з часів стародавньої Греції, за роботами Арістотеля, Сократа та Платона. Це також спостерігалося в роботах філософів-логіків XIX і XX століть, таких як Чарльз Сандерс Пірс і Готтлоб Фреге.

У 1960-х і 1970-х роках Чарльз Бахман та Е.П.Г. Браун працювали з пристроями, що стали передвестниками підходу Чена. Бахман розробив тип діаграми структури даних, який отримав назву Діаграма Бахмана. Браун опублікував роботи з моделювання систем реального світу. Джеймс Мартін вніс вдосконалення до ERD. Робота Чена, Бахмана, Брауна, Мартіна та інших також сприяла розвитку Мови моделювання об'єктів (UML), широко використовуваної в дизайні програмного забезпечення.

ER діаграми можуть бути використані у багатьох ситуації з найголовніших можна виділити:

Проектування баз даних: Діаграми ER використовуються для моделювання та проектування реляційних баз даних з точки зору логіки та бізнес-правил (у логічній моделі даних) та з точки зору конкретних технологій для впровадження (у фізичній моделі даних). У програмному інженерії діаграма ER часто є початковим кроком у визначенні вимог до проекту інформаційних систем. Також вона використовується для моделювання конкретної бази даних або декількох баз даних. Реляційна база даних має еквівалентну реляційну таблицю і, за необхідності, може бути виражена саме так.

Відлагодження баз даних: Діаграми ER використовуються для аналізу існуючих баз даних з метою виявлення та вирішення проблем у логіці чи розгортанні. Складання діаграми має виявити місце, де є проблеми.

Системи бізнес-інформації: Діаграми використовуються для проектування або аналізу реляційних баз даних, які використовуються в бізнес-процесах. Будь-який бізнес-процес, який використовує дані з сутностями, діями та взаємодією, може отримати вигоду від використання реляційної бази даних. Це може спростити процеси, легше виявляти інформацію та покращувати результати.

Освіта: Бази даних сьогодні - це метод зберігання реляційної інформації для освітніх цілей та подальшого витягування, тому діаграми ER можуть бути цінним інструментом для планування структур даних.

Дослідження: Оскільки багато досліджень спрямовані на структуровані дані, діаграми ER можуть відігравати ключову роль у створенні корисних баз даних для аналізу цих даних.

Основні компоненти діаграми сутностей включають:

Сутності (об'єкти): Це ключові елементи системи, які мають важливий смисловий або функціональний зв'язок з іншими сутностями. Наприклад, у системі керування навчанням можуть бути сутності такі як Студент, Викладач, Курс, Завдання тощо;

Відносини (зв'язки): Це взаємні зв'язки між сутностями, які визначають, як вони пов'язані між собою. Наприклад, у системі керування навчанням може бути відношення "Студент бере участь у Курсі", "Викладач веде Курс", тощо;

Атрибути: Це властивості або характеристики сутностей, які їх характеризують. Наприклад, для сутності Студент можуть бути атрибути Ім'я, Прізвище, Електронна пошта тощо.

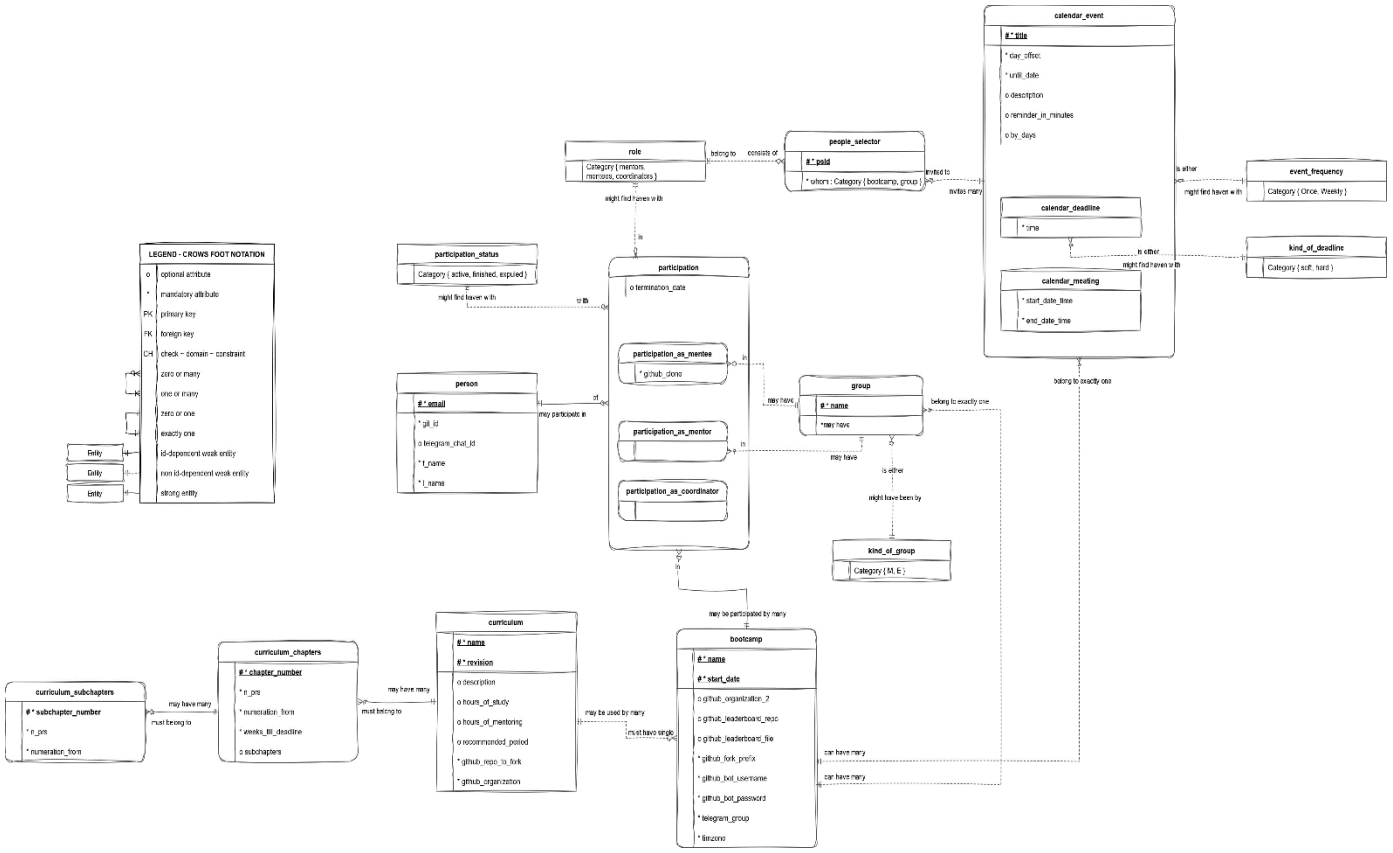

Рисунок 2.1 - Діаграма сутностей

Головні сутності:

- Особа: зібрання спільних властивостей сутностей ментора та студента;
- ментор: особа, яка надає підтримку, поради та наставництво студентам у процесі навчання;
- студент: особа, яка отримує освіту або навчається у навчальному закладі та може брати участь у різних навчальних подіях;
- група: колектив студентів, які навчаються разом та можуть брати участь у спільних навчальних заходах;
- подія: активність або захід, який відбувається у навчальному середовищі та може включати лекції, семінари, практичні заняття тощо;
- буткемп: інтенсивна програма навчання або тренування, спрямована на покращення певних навичок або здобуття нових знань.

2.4 Структури даних додатку

Структура даних додатку - це організація та упорядкування даних, які використовуються в програмному додатку. Це визначає, як дані будуть зберігатися, оброблятися і використовуватися в додатку для досягнення його функціональності і цілей [19].

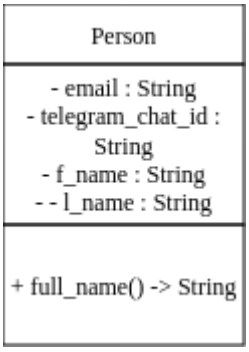

Рисунок 2.2 - Діаграма структури “Person”

Кожен об'єкт типу "Person" має чотири властивості:

- email: Рядок, що містить адресу електронної пошти особи.
- telegram_chat_id: Рядок, що містить ідентифікатор чату в Telegram для особи.
- f_name: Рядок, що містить ім'я особи.
- l_name: Рядок, що містить прізвище особи.

Крім того, в структурі "Person" визначений метод full_name(). Цей метод приймає об'єкт типу "Person" та повертає рядок, який містить повне ім'я особи, об'єднуючи значення "f_name" (ім'я) та "l_name" (прізвище) в один рядок.

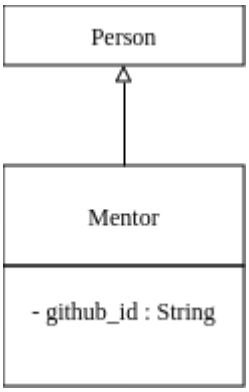

Рисунок 2.3 - Діаграма структури “Mentor”

Кожен об'єкт типу "Person" має одну властивість - github_id, який являє унікальний ідентифікатор на сервісі GitHub.

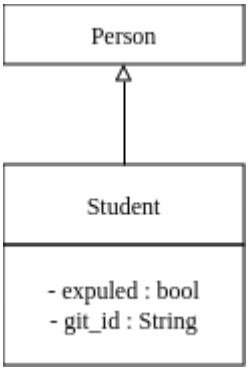

Рисунок 2.3 - Діаграма структури “Student”

Кожен об'єкт типу "Person" має дві властивості:

- Expuled: статус студента, який визначає чи виключений він з курсу;
- git_id: унікальний індентифікатор студента у сервісі GitHub.

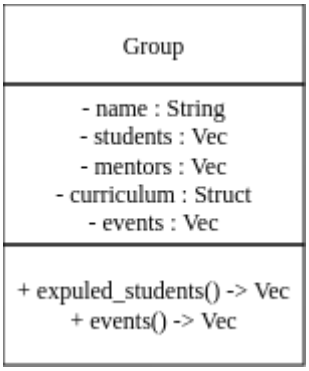

Рисунок 2.4 - Діаграма структури “Group”

Кожен об'єкт типу "Group" має чотири властивості:

- name: назва групи, яка використовується при відображенні прогресу групи;
- students: список студентів групи;
- mentors: список менторів групи;
- curriculum: програма та структура навчальної програми;

- events: список зустрічей групи.

Крім того, в структурі "Group" визначений метод expuled_students() та events(). Перший метод повертає список студентів, які були виключені з групи та не беруть участь у навчанні. Другий метод повертає список активних зустрічей групи.

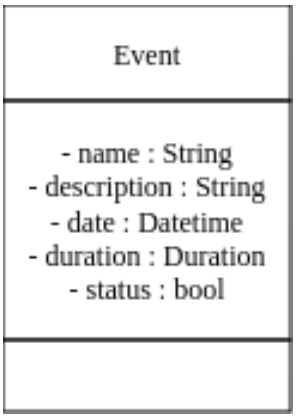

Рисунок 2.5 - Діаграма структури “Event”

Кожен об'єкт типу "Event" має п’ять властивостей:

- name: назва події, яка відображається користувача;
- description: опис події, яка відображає корисну інформацію для студента чи ментора;
- date: дата зустрічі;
- duration: тривалість події;
- status: статус події.

2.5 Блок-схема і структурна схема додатку

Структурна схема додатку - це візуальне представлення компонентів або модулів програмного забезпечення та їх взаємозв'язків. Ця схема допомагає зрозуміти загальну архітектуру додатку, відображаючи, як компоненти взаємодіють між собою для досягнення певного функціоналу [20].

Структурна схема додатку може включати наступні елементи:

- Модулі або компоненти: Це окремі частини програми, які виконують конкретні функції або завдання. Наприклад, це можуть бути модулі для керування користувачами, обробки даних, взаємодії з базою даних тощо.
- Взаємозв'язки між модулями: Вони показують, які модулі взаємодіють між собою для обміну даними або виконання певних операцій. Це може бути виклик методів одного модуля з іншого, передача даних через інтерфейси або використання спільних ресурсів.

- Користувацький інтерфейс: Це компоненти, які відповідають за взаємодію з користувачем. Це може бути веб-інтерфейс, мобільний інтерфейс або графічний інтерфейс користувача.
- База даних: Це модуль, який відповідає за збереження та управління даними, необхідними для роботи додатку. Він може включати схему бази даних та компоненти для доступу до даних.

Вона має містити:

- Інтерфейс користувача за допомогою якого адміністратор може проводити взаємодію зі системою;
- головний додаток learn together, який містить у собі головну логіку додатку та агрегує взаємодію з іншими сервісами;
- cron jobs, які будуть виконувати періодичні завдання для збору та актуалізації даних;
- сторонній сервіс Github та його адаптер для перевірки виконаних завдань студентів;
- сторонній сервіс Telegram та його адаптер для взаємодії з користувачем;
- сторонній сервіс Google Meet для організації відео зустрічей зі студентамиl
- сторонній сервіс Google Calendar для відображення кінцевих термінів здачі завдань студентами та відображення зустрічей.

Побудуємо структурну схему нашого додатку.

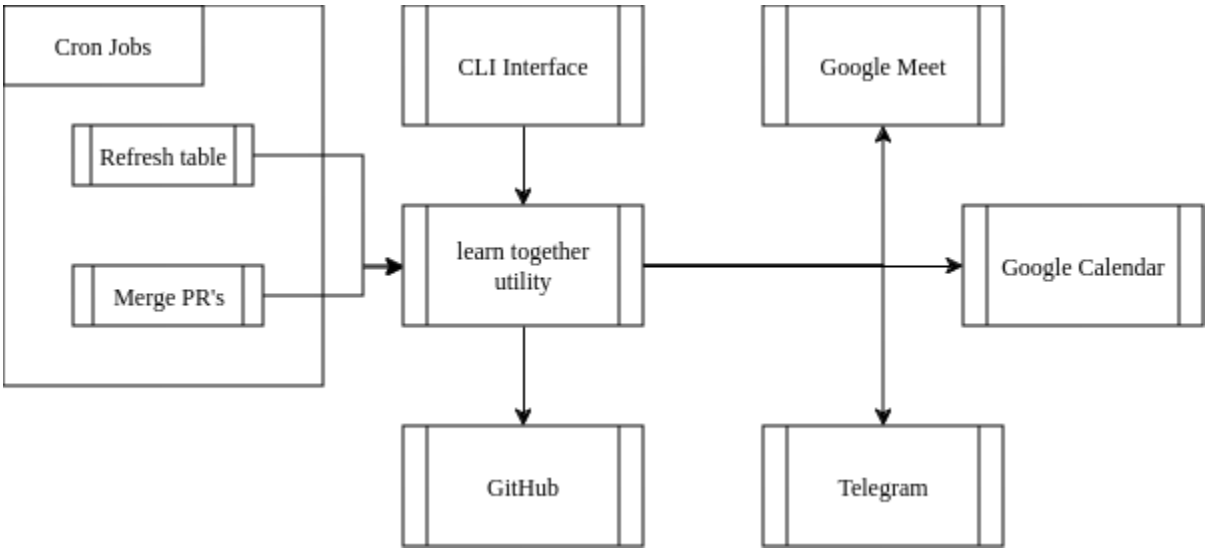

Рисунок 2.6 - Структурна схема додатку

2.6 Висновки

Було виставлено вимоги до розроблюваної системи, проведено аналіз даних додатку. Також було побудовано діаграму сутностей додатку, яка дозволяє розробити структуру бази даних. Ще було розроблено структурну схему додатку, яка відображає всі елементи системи.

У результаті отримати технічну документацію, яка дозволяє приступити до технічної реалізації додатку.

3 РОЗРОБКА СИСТЕМИ КЕРУВАННЯ НАВЧАННЯМ

3.1 Аналіз і обґрунтування вибору інтерфейсу користувача

Вибір правильної технології для реалізації програмного продукту є ключовим етапом, який визначає подальший процес розробки та може спростити або ускладнити його етапи. Важливо ідентифікувати ключові аспекти програмного продукту і обрати інструменти, які найкраще відповідатимуть поставленим завданням.

Для реалізації інтерфейсу потрібно вибрати інструменти, які буду легкі у розробці та надаватимуть досвід, який не буде відрізнятися в залежності від платформи.

Інтерфейс командного рядка (CLI) - це спосіб взаємодії з комп'ютерною програмою або операційною системою шляхом введення текстових команд у спеціальне вікно або термінал. У CLI користувач може вводити команди за допомогою клавіатури, а програма або операційна система відповідає виконанням цих команд [21].

Основні характеристики інтерфейсу командного рядка включають:

Відсутність графічних елементів і використання лише текстових команд.

Можливість виконання різних операцій та керування системою або програмою через введення відповідних команд.

Зручний для автоматизації завдань та роботи з великими обсягами даних.

Широке використання в системному адмініструванні, програмуванні та роботі з серверними системами.

Інтерфейс командного рядка залишається популярним і корисним інструментом для багатьох користувачів, особливо в області програмування та адміністрування комп'ютерних систем.

Інтерфейс користувача з графічним відображенням (GUI) - це спосіб взаємодії з комп'ютерною програмою або операційною системою, який використовує графічні елементи, такі як кнопки, піктограми, вікна та меню, для представлення інформації та надання можливості виконання дій за допомогою миші або клавіатури [22].

Основні характеристики інтерфейсу GUI включають:

Використання графічних елементів для відображення інформації та взаємодії з користувачем.

Можливість виконання різних операцій шляхом натискання кнопок, перетягування об'єктів та інших дій з використанням миші або клавіатури.

Інтуїтивний та простий для користувача інтерфейс, який зазвичай має візуальні підказки та іконки.

Зручність для користувачів, які не знають команд або не мають технічних навичок.

Інтерфейс GUI широко використовується в різних сферах, включаючи операційні системи, програмне забезпечення, веб-додатки та ігри, завдяки своїй зручності та доступності для широкого кола користувачів.

З огляду на особливості створюваної системи доцільно буде вибрати інтерфейс командного рядка, який не тільки простий у реалізацій а як найкраще підходить для автоматизації завдань зі оновлення актуальних даних та відсутність прямої взаємодії користувача з системи.

3.2 Аналіз і обґрунтування вибору засобів для реалізації програмного засобу

Інструменти для розробки CLI (Command Line Interface) додатків - це програми, бібліотеки або набори інструментів, які допомагають програмістам створювати, тестувати і підтримувати програми, які взаємодіють з користувачем через командний рядок або термінал. Ці інструменти надають зручний спосіб для обробки аргументів командного рядка, виведення інформації користувачу, керування курсором та інші корисні можливості. Вони допомагають забезпечити ефективну та приємну розробку CLI додатків, спрощуючи рутинні завдання та надаючи різноманітні інструменти для покращення функціональності програми.

Для створення командних інтерфейсів у Rust існує кілька бібліотек, кожна з яких пропонує свої унікальні можливості та переваги.

Clap (Command Line Argument Parser) є однією з найпопулярніших бібліотек для розробки CLI додатків у Rust. Clap надає потужний та гнучкий інструмент для обробки командного рядка, підтримуючи різні формати аргументів, підкоманди, автодоповнення та генерацію документації. Основною перевагою Clap є його багатофункціональність та зручність використання, що дозволяє швидко створювати складні інтерфейси

командного рядка. Недоліком може бути більша кількість конфігураційних опцій, що може ускладнити використання для простих проектів.

Argh є ще однією бібліотекою для парсингу аргументів командного рядка, яка надає мінімалістичний підхід до створення CLI додатків. Argh використовує атрибути для визначення параметрів та опцій, подібно до StructOpt, але зосереджується на мінімалізмі та простоті використання. Основною перевагою Argh є його легкість та швидкість навчання. Однак, для більш складних випадків використання, Argh може не мати достатньої функціональності.

Gumdrop є ще однією бібліотекою для створення CLI додатків, яка використовує макроси для визначення параметрів та опцій. Gumdrop надає баланс між простотою використання та можливостями конфігурації. Вона підтримує більшість необхідних функцій для створення CLI додатків, таких як підкоманди, опціональні параметри та значення за замовчуванням. Основною перевагою Gumdrop є її гнучкість та зручність використання. Однак, вона менш популярна, ніж Clap чи StructOpt, і має менш активну спільноту.

Docopt є бібліотекою, яка базується на філософії Docopt, що використовує опис аргументів командного рядка у вигляді тексту документації для автоматичного парсингу. Це дозволяє створювати дуже читабельні та зрозумілі специфікації аргументів. Основною перевагою Docopt є її зручність для написання та розуміння документації. Проте, вона може бути менш продуктивною, ніж інші бібліотеки, і менш активно підтримується у Rust-спільноті.

wca - це бібліотека для створення CLI додатків у Rust, яка зосереджується на інтеграції з веб-сервісами. Ця бібліотека дозволяє створювати командні інтерфейси, які можуть взаємодіяти з веб-додатками та API, забезпечуючи простий і ефективний спосіб реалізації таких взаємодій у командному рядку.

Переваги використання wca включають її орієнтованість на інтеграцію з веб-сервісами, що робить її ідеальною для розробки CLI додатків, які повинні взаємодіяти з API або іншими веб-додатками. Вона також відзначається простотою використання та налаштування.

Недоліками можуть бути обмеження функціональності у порівнянні з більш загальними бібліотеками для створення CLI додатків, такими як Clap

або StructOpt. wca більше спеціалізується на веб-інтеграції, тому для деяких сценаріїв використання може знадобитися комбінування з іншими бібліотеками для досягнення потрібної функціональності.

З огляду на особливості розроблюваної системи буде доцільно вибрати wca через легкість інтеграції з іншими додатками.

3.3 Розробка інтерфейсу командного рядку

Для розробки CLI-додатку спершу потрібно визначити список необхідних команд та їх призначення. Це включає в себе розуміння, які функції повинен виконувати додаток і які саме команди будуть потрібні для реалізації цих функцій. Важливо також врахувати можливість автоматизації цих команд, щоб забезпечити зручність використання та інтеграцію з іншими системами чи процесами.

Складемо список необхідних команд та їх властивостей.

Команда “calendar.create” призначена для створення нового календаря для групи. Вона дозволяє користувачам створювати календар з заданим іменем (summary) та надає можливість здійснити пробне виконання команди без фактичного створення календаря (режим dry run).

Команда “calendar.deadlines” призначена для створення подій з дедлайнами та додавання їх у конфігураційний файл. Ця команда автоматизує процес створення подій, які мають чітко визначені кінцеві терміни. Команда має одну властивість “dry”. Це необов'язкова властивість, яка вказує на режим пробного виконання. Якщо встановлено в 1 або не вказано взагалі, команда покаже, що буде виконано, без фактичного створення подій. Якщо встановлено в 0, команда створить події з дедлайнами.

Команда “calendar.list” призначена для отримання списку календарів з Google API та їх відображення. Вона дозволяє користувачам отримувати інформацію про всі доступні календарі та виводити їх на екран. Команда не має додаткових властивостей. Вона просто виконує запит до Google API для отримання списку календарів і відображає результати.

Команда “calendar.select” призначена для вибору календаря для групи. Якщо існує календар з введеною назвою, він буде обраний; якщо ні, команда може створити новий календар з цією назвою і записати його в конфігурацію групи.

Команда “calendar.sync” призначена для пошуку відмінностей між подіями в локальному календарі та Google Calendar API. Ця команда виконує однонаправлену синхронізацію, тобто вона не вносить зміни до локальних конфігураційних файлів, але оновлює події в визначеному календарі Google.

Команда calendar.validate призначена для перевірки коректності даних подій у календарі. Вона аналізує можливі проблеми з даними подій, читає всі групи та перевіряє їхні події, а також перевіряє наявність необхідних полів у конфігураційних файлах.

Команда “consistency.validate” призначена для аналізу потенційних проблем, які можуть виникнути під час роботи програми. Вона перевіряє різні аспекти конфігурацій і даних груп, такі як наявність конфігураційних файлів для заявлених груп та наявність дублікатів студентів.

Команда “curriculum.about” призначена для отримання топології навчального плану. Вона збирає інформацію про топологію навчального плану з файлу TOML і показує, скільки розділів містить навчальний план та скільки завдань є в кожному розділі. Команда не має додаткових властивостей і просто збирає інформацію про топологію навчального плану та виводить її у вигляді таблиці.

Команда “events.watch” призначена для спостереження за подіями, що надходять з вказаних репозиторіїв або від студентів. Вона створює вебхук у вказаному репозиторії, якщо він ще не існує, і виводить всі події, що були надіслані з моменту створення вебхуку. Команда не має додаткових властивостей і просто створює вебхук та виводить всі надіслані події з моменту створення вебхуку.

Команда “forks.made” призначена для створення репозиторіїв з шаблону в залежності від поточних студентів або фільтра груп. Команда створює репозиторії з шаблону в залежності від поточних студентів або фільтра груп.

Команда “forks.sync” призначена для синхронізації гілок (forks) студентів та їх подальшої синхронізації з оригінальними гілками.

Команда “gcp.login” призначена для входу в систему з вашим обліковим записом Google, щоб отримати доступ до календаря. Ця команда запускає процес авторизації, який дозволяє вашій програмі отримувати доступ до календаря Google. Вона відкриває сторінку авторизації Google у вашому веб-браузері, де ви можете увійти до свого облікового запису Google, якщо це

необхідно. Після цього вас перенаправлять на сторінку з результатом авторизації. Якщо процес авторизації успішний, токен оновлення буде записано у конфігураційний файл групи.

Команда “group” призначена для встановлення назви групи в програмі. Ця команда дозволяє встановлювати фільтр за назвою групи для програми. Програма має три фільтри: студенти, наставники та групи. Якщо вибрано лише групу, будуть використовуватися студенти або наставники з вибраної групи. Якщо вибрано лише одного або кількох студентів або наставників, то будуть використовуватися лише вони.

Команда “group.add” використовується для додавання нової групи до списку груп та запам'ятовування каталогу, в якому знаходиться файл конфігурації. Ця команда дозволяє додавати нову групу до списку груп та запам'ятовувати каталог, в якому знаходиться файл конфігурації.

Команда “group.del” використовується для видалення групи зі списку груп. Ця команда не впливає на конфігураційний файл групи. Ця команда дозволяє видаляти групу зі списку груп. Параметр команди - це назва групи, яку потрібно видалити.

Команда “group.messanger.send” використовується для надсилання повідомлення в груповий чат у Telegram. Ця команда приймає текстове повідомлення та назву групи. Відправляє отримане повідомлення у чат Telegram, вказаний користувачем.

Команда “mentors.invite” використовується для надсилання запрошень на співпрацю менторам у репозиторії студентів. Ця команда надсилає запрошення менторам у репозиторії студентів, які призначені їм. Якщо ви вказали одного студента у фільтрі, запрошення буде надіслано усім менторам, яким він призначений. Якщо ви вказали одного ментора, запрошення буде надіслано до всіх репозиторіїв студентів, які призначені цьому ментору.

Команда “prs.merge” використовується для злиття пул-запитів студентів. Ця команда зливає пул-запити одного або кількох студентів. Студенти, які використовуються, залежать від поточного фільтра. Якщо ви вказали одного студента у фільтрі, то запити будуть злиті усіх студентів. Якщо ви вказали кількох студентів, то запити будуть злиті для них усіх.

Команда “student.email.send” використовується для надсилання електронних листів студентам. Ця команда надсилає електронні листи

студентам залежно від поточного фільтра. Ви можете вказати повідомлення як аргумент команди.

Команда “student.prs” призначена для отримання списку затверджених запитів на злиття (Pull Requests, PRs) для одного студента. Ця команда показує всі затверджені запити на злиття студентів в залежності від поточного фільтра. Ви можете вказати параметри, такі як показ тільки затверджених PRs, відображення прогресу студента за завданнями, фільтрація за розділом та завданням, а також за датою оновлення.

Команда “students.invitations” призначена для відображення списку запрошень студентам у репозиторії. Ця команда показує список запрошень студентам до репозиторіїв для бота. Відображаються такі дані як номер запрошення, ім'я студента, користувач, який надіслав запрошення , ідентифікатор репозиторію , повне ім'я репозиторію та чи є репозиторій приватним.

Команда “students.invite” призначена для надсилання запрошень студентам до їхнього репозиторію. Ця команда надсилає запрошення студентам у їхній репозиторій. Студенти використовуються залежно від поточного фільтра. Ви також можете переглянути, хто прийняв або не прийняв запрошення. При спробі надіслати запрошення знову ви побачите список тих, хто вже прийняв запрошення, і тих, кому програма все ще намагається надіслати запрошення (тому що вони ще не прийняли його).

Команда “students.progress” призначена для відображення прогресу студентів у виконанні завдань. Ця команда дозволяє вам переглянути загальний прогрес навчальної групи, показуючи кількість завдань, виконаних кожним студентом. За замовчуванням інвалідування вимкнене, що означає, що інформація синхронізується з кешем, а кеш оновлюється за потреби. Студенти використовуються залежно від поточного фільтра.

Команда “students.progress.board” призначена для оновлення дошки прогресу студентів у репозиторії. Ця команда дозволяє оновити дошку прогресу студентів у репозиторії, показуючи кількість завдань, виконаних кожним студентом. Оновлення відбувається для всіх груп.

Команда validate призначена для аналізу можливих проблем з даними програми. Результат виводу команди буде містити звіти про можливі проблеми з різних аспектів програми. Це може включати проблеми з

консистентністю даних, проблеми з налаштуваннями менторів та студентів, а також інші види валідації, які були визначені для програми.

Команда exit призначена для завершення роботи програми. Використання цієї команди призведе до закриття програми.

3.4 Автоматизація виконання команд

Cron - це системний інструмент для автоматизації виконання завдань на ОС Linux та UNIX. Він дозволяє запускати команди або сценарії на підставі розкладу, заданого користувачем. Основна концепція полягає в тому, що ви можете налаштувати cron job, щоб ваша система виконувала певні завдання заздалегідь визначеним чином [23].

Автоматизація виконання завдань на Linux за допомогою cron jobs - це досить поширений спосіб.

Напишемо скрипт для автоматичного злиття ПР'ів студентів рисунок 3.1.

```
1  cd /home/ghuba/git/learn_together
2  cargo run -- .prs.merge dry:0
```

Рисунок 3.1 - Скрипт автоматичного запуску злиття ПР'ів

Команда "cd" переходить до каталогу "/home/ghuba/git/learn_together". Це зручно, якщо ви хочете виконати наступну команду у вказаному каталозі.

Команда cargo run -- .prs.merge dry:0 запускає проект за допомогою Cargo, менеджера пакетів для мови програмування Rust. Параметр -- слугує для передачі додаткових аргументів програмі, що викликається. У цьому випадку, . означає, що ми запускаємо поточний проект, а .prs.merge dry:0 - це аргументи, що передаються програмі.

Напишемо крипт, для оновлення таблиці прогресу студентів рисунок 3.2.

```
1  cd /home/ghuba/git/learn_together
2  cargo run --  .students.progress.board unclassified:1 emoji:1 short_name:1 generation_time:1 chapter:2 leaving_message:1 in_alt:1
```

Рисунок 3.2 - Скрипт автоматичного оновлення дошки прогресу

Команда "cd" переходить до каталогу "/home/ghuba/git/learn_together".

Команда “.students.progress.board unclassified:1 emoji:1 short_name:1 generation_time:1 chapter:2 leaving_message:1 in_alt:1” запускає процес оновлення дошки прогресу студентів у навчальному плані.

3.5 Висновки

У третьому розділі було обґрунтовано вибір інтерфейсу користувача та технологій, які будуть використовуватися під час розробки системи та наведено їх особливості. У результаті було обрано командний інтерфейс користувача та бібліотеку wca для його розробки.

4 ТЕСТУВАННЯ ДОДАТКУ

4.1 Методи тестування програмного забезпечення

Тестування програмного забезпечення є важливим етапом у розробці, який дозволяє виявити та виправити помилки, покращити якість і надійність продукту. Існує кілька основних методів тестування, кожен з яких має свої переваги та недоліки.

Юніт-тестування, або модульне тестування, зосереджується на перевірці окремих компонентів або модулів коду. Цей метод дозволяє ізолювати та тестувати кожен модуль окремо, що допомагає швидко виявити помилки на ранніх етапах розробки. Основною перевагою юніт-тестування є можливість раннього виявлення багів, що знижує витрати на їх виправлення у майбутньому. Крім того, юніт-тести легко автоматизуються і забезпечують високу покритість коду. Проте цей метод може мати обмежену ефективність при тестуванні інтеграції модулів або виявленні помилок, пов'язаних із взаємодією різних частин системи [24].

Інтеграційне тестування спрямоване на перевірку взаємодії між різними модулями або компонентами системи. Після того, як окремі модулі пройшли юніт-тестування, вони об'єднуються та тестуються разом, щоб забезпечити коректну роботу всієї системи в цілому. Інтеграційне тестування допомагає виявити проблеми, пов'язані з інтерфейсами та взаємодією компонентів. Перевагою цього методу є здатність виявляти баги на рівні взаємодії модулів, що може бути пропущено під час юніт-тестування. Однак інтеграційне тестування може бути складним і витратним, особливо при великій кількості компонентів і складних зв'язках між ними.

Системне тестування перевіряє всю систему в цілому, включаючи всі її модулі та інтеграції. Це тестування забезпечує повну перевірку функціональності, продуктивності, безпеки та інших аспектів системи. Основною перевагою системного тестування є можливість оцінити, чи відповідає система всім вимогам і очікуванням користувачів. Цей метод дозволяє виявити проблеми, які можуть виникнути при реальній експлуатації ПЗ. Недоліком системного тестування є його висока вартість і тривалість, оскільки воно вимагає значних ресурсів і часу для повного охоплення всіх аспектів системи [25].

Системне тестування перевіряє всю систему в цілому, включаючи всі її модулі та інтеграції. Це тестування забезпечує повну перевірку функціональності, продуктивності, безпеки та інших аспектів системи. Основною перевагою системного тестування є можливість оцінити, чи відповідає система всім вимогам і очікуванням користувачів. Цей метод дозволяє виявити проблеми, які можуть виникнути при реальній експлуатації ПЗ. Недоліком системного тестування є його висока вартість і тривалість, оскільки воно вимагає значних ресурсів і часу для повного охоплення всіх аспектів системи.

Регресійне тестування є важливою складовою процесу тестування після внесення змін або додавання нових функцій у ПЗ. Воно перевіряє, чи не з'явилися нові помилки в існуючому функціоналі після змін. Регресійне тестування допомагає забезпечити стабільність системи та запобігти зниженню якості ПЗ. Перевагою цього методу є можливість швидко виявити проблеми, що виникають через зміни в коді. Проте повторне проведення тих самих тестів може бути трудомістким і дорогим, особливо якщо тестування виконується вручну. Автоматизація регресійного тестування може знизити витрати, але вимагає початкових інвестицій у створення та підтримку тестових скриптів.

Тестування продуктивності перевіряє, як система працює під навантаженням, включаючи тестування швидкості, масштабованості та стабільності. Це тестування допомагає визначити, чи відповідає ПЗ вимогам щодо продуктивності і як воно справляється з високим навантаженням. Перевагою цього методу є можливість виявлення вузьких місць і потенційних проблем продуктивності до випуску продукту. Проте тестування продуктивності може бути складним і дорогим, оскільки вимагає спеціалізованих інструментів і знань.

Фаззинг, або fuzz-тестування, є технікою тестування програмного забезпечення, яка полягає у подачі на вхід програми великої кількості випадково згенерованих або мутованих даних з метою виявлення помилок, вразливостей або некоректної поведінки. Цей метод особливо ефективний для виявлення помилок, які важко знайти за допомогою традиційних методів тестування.

Основною перевагою fuzz-тестування є його здатність автоматично генерувати великий обсяг тестових даних, що дозволяє перевірити програму на різні, часто несподівані сценарії. Це допомагає виявити не тільки функціональні помилки, але й проблеми безпеки, такі як буферні переповнення, SQL-ін'єкції, XSS-атаки та інші вразливості. Завдяки автоматизації, fuzz-тестування може виконуватись безперервно, забезпечуючи високий рівень покриття коду.

Недоліком fuzz-тестування є те, що воно може генерувати величезну кількість тестових випадків, з яких лише невелика частина може виявитися корисною для виявлення реальних проблем. Це може призвести до великої кількості "шуму" — незначних помилок або попереджень, які не є критичними. Крім того, fuzz-тестування не завжди може визначити точну причину виявленої помилки, що вимагає додаткового аналізу та діагностики.

Також важливо враховувати, що fuzz-тестування, хоча й ефективне для виявлення багатьох типів помилок, не може замінити інші методи тестування. Воно є доповненням до юніт-тестування, інтеграційного тестування, тестування продуктивності та інших методів, які забезпечують всебічну перевірку якості програмного забезпечення.

Комбінування різних методів тестування є ключовим для створення комплексної та ефективної стратегії, оскільки різні методики виявляють помилки, які особливі лише для них, забезпечуючи повне та всебічне тестування. Різні етапи тестування дозволяють знаходити помилки на всіх етапах розробки додатку постійно покращуючи його якість.

З огляду на особливість розроблюваного продукту було прийнято обрати інтеграційне для перевірки працездатності додатку.

Інтеграційне дає змогу перевірити роботу додатку у взаємодії з усією системою. Оскільки розроблювана система має працювати у складній взаємодії з безліччю сторонніх сервісів потрібно бути впевненим, що ця інтеграція є надійною.

4.2 Інтеграційне тестування системи

Для перевірки працездатності системи потрібно провести тестування взаємодії з Google Calendar де будуть створюватися події для студентів, Github

на якому студенти мають здавати виконанні завдання та перевірку взаємодії системи з даними користувачів.

Перевірка створення події у календарі рисунок 4.1.

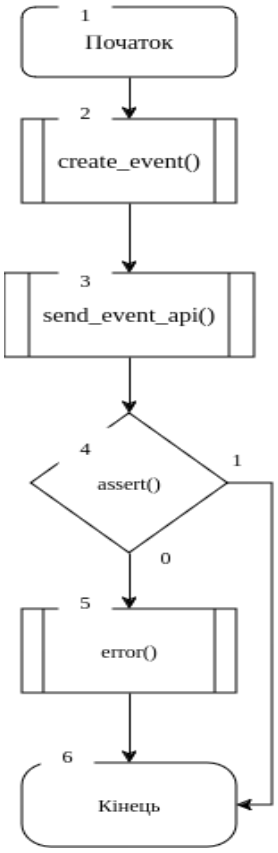

Рисунок 4.1 - Тестування функції створення події

Створюються тестові дані, та викликається метод створення нової події у календарі. “assert” переконується, що подію було успішно створення за допомогою визначеного id. Перевірка гарантує що події успішно створюються у календарі.

Перевірка створення події у календарі без визначеного нами id рисунок 4.2.

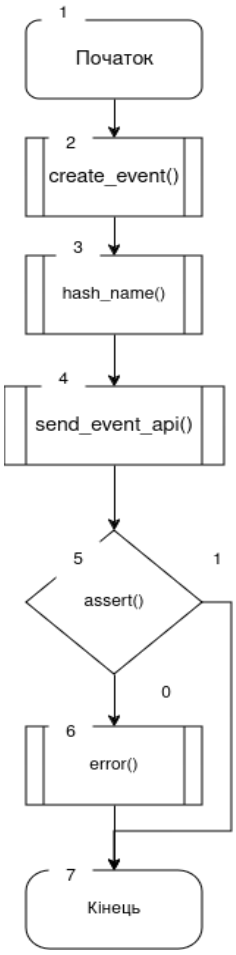

Рисунок 4.2 - Тестування функції створення події з хешованим id

Створюється подія, яка не містить у собі поля “id”, яке необхідно при створення події у Google Calendar. Створюється новий id для якого за основу береться назва та опис події, а “assert” переконується, що подію було успішно створення за допомогою згенерованого id. Перевірка гарантує що події успішно створюються у календарі навіть за відсутності викзначеного ключа.

Перевірка метода синхронізації календаря рисунок 4.3.

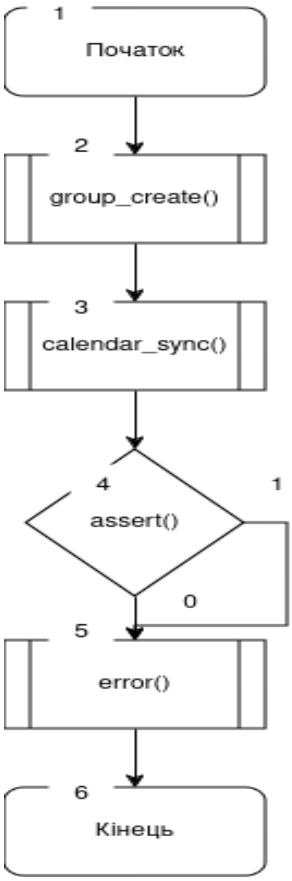

Рисунок 4.3 - Тестування функції знаходження різниці

Створюються тестові дані. Вони складаються зі групи для якої є події, та календаря у якому містяться події, які необхідно оновити, створити та видалити. Цей тест гарантує, що при спробі оновити події групи, буде знайдено різницю у календарі локальному, та тому, що зберігається у Google Calendar. Старі не актуальні події, мають бути видалені з нього, а події, які мають змінену кількість учасників, чи назву мають бути відповідно оновленні. Решта з події не мають зазнати ніяких змін.

Перевірка метода отримання інформації про репозиторій рисунок 4.4.

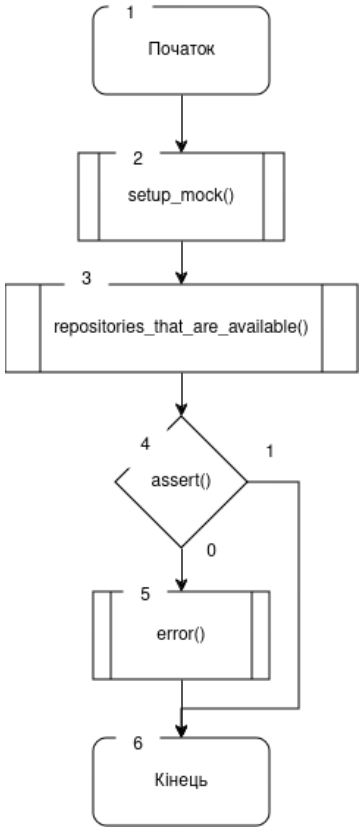

Рисунок 4.4 - Тестування інформації про репозиторій

Тестовий випадок перевіряє, що при надані тестових даних метод “repositories_that_are_available” поверне правильну інформації про репозиторій для якого був зроблений запит. Зокрема власником цього репозиторія має бути студент для якого зробили запит.

Перевірка метода отримання PR’ів студента рисунок 4.5.

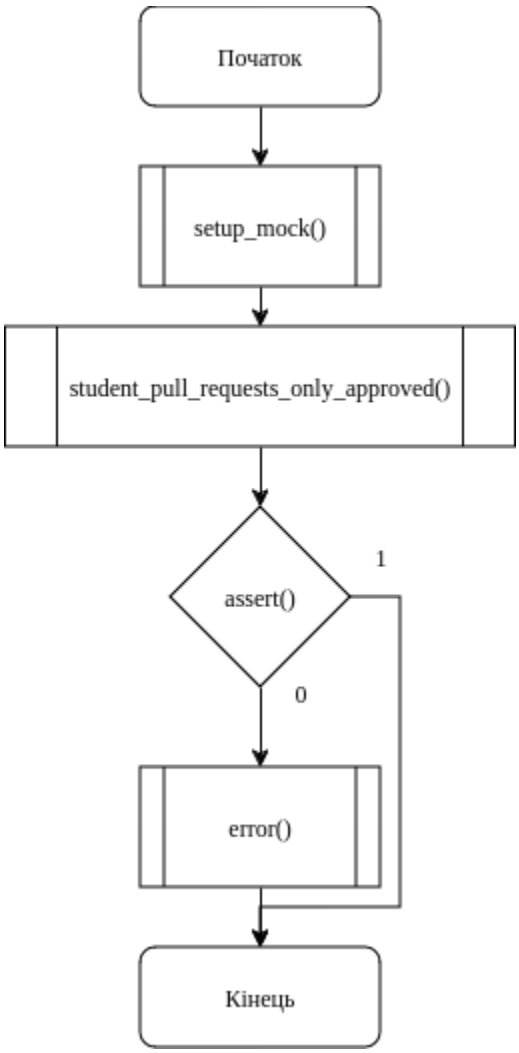

Рисунок 4.5 - Тестування отримання PR’ів студента

Тестовий випадок перевіряє, що метод “student_pull_requests_only_approved” отримує данні, їх оброблює та повертає коректний список зі PR’ами студента, які отримали перевірку ментора.

Лістинг тесту перевірки роботи глобального конфігураційного файлу рисунок 4.5.

```rust
#[ test ]
fn group_add_and_delete() -> anyhow::Result< () >
{
  const PATH : &str = "lrt_test.lt.toml";
  let group_name = GroupName( "lrt_test".into() );

  let dir =  DIR.get_or_init( ||
  {
    tmp_dir( std::env::current_dir()
    .expect( "failed to get current dir").join( FOLDER_NAME ) ).expect( "Failed to get tmp dir")
  });

  let path = GroupConfigPath::try_from( AbsolutePath::new( dir.to_path_buf().join( PATH ) )? )?;

  let add_name = GlobalConfig::< TestPath >::global_config_group_add( path ).unwrap();

  assert_eq!( add_name, group_name );

  let state = GlobalConfig::< TestPath >::global_config_read().unwrap();
  assert!( state.groups.contains_key( &group_name ) );

  let delete_name = group_delete::< GlobalConfig< TestPath > >( &add_name );
  assert!( delete_name.is_ok() );

  let state = GlobalConfig::< TestPath >::global_config_read().unwrap();
  assert!( !state.groups.contains_key( &group_name ) );

  Ok( () )
}
```

Рисунок 4.5- Лістинг тесту глобального конфігураційного файлу

Цей тест перевіряє функціональність додавання та видалення групи у конфігурації за допомогою бібліотеки GlobalConfig. Він створює тимчасову директорію для зберігання файлу конфігурації, визначає шлях та ім'я групи, додає цю групу до конфігурації, перевіряє, чи група була додана успішно, а потім видаляє цю групу та перевіряє, чи вона була успішно видалена. Цей процес відбувається в межах одного тестового випадку, що забезпечує впевненість у правильному функціонуванні обох операцій.

Тест перевірки перевірки навчального плану та правильності його зчитування з конфігураційного файлу рисунок 4.6.

```rust
#[ test ]
fn curriculum_about_1() -> anyhow::Result< () >
{
  const PATH : &str = "1_lrt_2022.lt.toml";
  const CACHE : &str = "approved_pr_list.json";

  let ( group_context, _tmp_dir ) = build_ctx( PATH, CACHE )?;

  assert_eq!( group_context.sum_tasks_in_chapter( 4 )?, 7 );
  assert_eq!( group_context.sum_all_tasks( )?, 36 );

  Ok(())
}
```

Рисунок 4.6 - Лістинг тесту перевірки навчального плану

Тест `curriculum_about_1` перевіряє правильність читання топології навчального плану. Основною метою є перевірка, чи функції “all_tasks_in_chapter()” і “all_tasks_in_curriculum()” правильно рахують кількість завдань у розділі та всьому навчальному плані. Важливим аспектом є те, що в топології є завдання з нульовою кількістю вправ, і якщо це не враховується під час обчислень, результат буде некоректним. Тест використовує наступну топологію розділів: “[[1,0], [10,0],[5,0], [13,0], [10,1]]”.

Тестовий сценарій починається з визначення шляху до файлу конфігурації та кешу. Далі створюється контекст групи та тимчасова директорія за допомогою функції “build_ctx”, яка ініціалізує необхідні параметри на основі наданих файлів. Після цього перевіряється кількість завдань у четвертому розділі, яка має дорівнювати 7, та загальна кількість завдань у навчальному плані, яка повинна бути 36. Використовуючи функції “sum_tasks_in_chapter()” та “sum_all_tasks()”, тест підтверджує, що всі завдання підраховані правильно, враховуючи завдання з нульовою кількістю вправ.

Таким чином, цей тест забезпечує перевірку коректного підрахунку завдань у навчальному плані, особливо враховуючи специфічні випадки, коли кількість вправ дорівнює нулю.

4.3 Висновки

У цьому розділі було розглянуто методи тестування програмного забезпечення та було обрано модульне тестування та інтеграційне тестування для перевірки системи керування навчання.

Проведені тестування підтвердили, що робота програми відповідає поставленим вимогам.

5 ЕКОНОМІЧНА ЧАСТИНА

Для успішного прийняття та впровадження науково-технічних розробок важливо, щоб вони відповідали сучасним вимогам як у науково-технічному прогресі, так і в економічному аспекті. Тому необхідно проводити оцінку економічної доцільності отриманих результатів науково-дослідної діяльності.

Магістерська кваліфікаційна робота на тему "Програмні засоби для удосконалення систем керування навчання" відноситься до науково-технічних проектів, спрямованих на можливе впровадження на ринок. Це означає, що розробка має потенціал для комерційного використання, і цей напрямок вважається пріоритетним, оскільки результати роботи можуть бути корисні для широкого кола користувачів і принести економічний виграш. Проте для досягнення цієї мети потрібно залучити потенційного інвестора, який би здійснив фінансування впровадження цієї розробки та виведення її на ринок.

Для реалізації даного сценарію робіт потрібно виконати наступні етапи:

Провести комерційний аудит науково-технічної розробки для встановлення її науково-технічного рівня та комерційного потенціалу.

Розрахувати витрати на здійснення науково-технічної розробки, включаючи витрати на дослідницьку роботу, матеріали, обладнання та інші ресурси.

Провести аналіз економічної ефективності впровадження науково-технічної розробки, враховуючи очікуваний ринковий попит, прибуток від продажу продукту та інші фінансові показники. Обґрунтувати економічну доцільність комерціалізації потенційним інвестором.

Ці кроки допоможуть забезпечити успішне впровадження науково-технічної розробки та максимізувати її комерційний потенціал.

5.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Мета проведення комерційного і технологічного аудиту дослідження за темою " Програмні засоби для удосконалення систем керування навчання" полягає в оцінюванні науково-технічного рівня та комерційного потенціалу розробки, яка створена в результаті науково-технічної діяльності.

Для оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується використовувати 5-ти бальну систему оцінювання за 12-ма критеріями, наведеними в табл. 5.1.

Таблиця 5.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					
	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах
Ринкові переваги (недоліки)					
	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів

	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
0	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві

1	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
2	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь- які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно звести до таблиці.

Таблиця 5.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	5	5	4
2. Ринкові переваги (наявність аналогів)	5	4	4
3. Ринкові переваги (ціна продукту)	4	4	4
4. Ринкові переваги (технічні властивості)	5	4	4
5. Ринкові переваги (експлуатаційні витрати)	3	3	3
6. Ринкові перспективи (розмір ринку)	3	4	3
7. Ринкові перспективи (конкуренція)	4	3	4
8. Практична здійсненність (наявність фахівців)	3	3	3
9. Практична здійсненність (наявність фінансів)	3	3	4
10. Практична здійсненність (необхідність нових матеріалів)	4	4	4
11. Практична здійсненність (термін реалізації)	3	3	4
12. Практична здійсненність (розробка документів)	3	3	4
Сума балів	45	43	45
Середньоарифметична сума балів СБс	44		

За результатами розрахунків, наведених в таблиці 5.2, можна зробити висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому важливо використовувати рекомендації, наведені в таблиці 5.3, для об'єктивної оцінки цих параметрів.

Таблиця 5.3 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ , розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

Згідно проведених досліджень, рівень комерційного потенціалу розробки за темою "Програмні засоби для удосконалення систем керування навчання" становить 44 бали. Згідно з таблицею 5.3, такий результат свідчить про високу комерційну важливість проведення цих досліджень.

5.2 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов'язані з проведенням науково-дослідної роботи на тему "Програмні засоби для удосконалення систем керування навчання", розглядаються, групуються та обліковуються за відповідними статтями під час планування та розрахунку собівартості проекту.

5.2.1 Витрати на оплату праці

До статті "Витрати на оплату праці" включаються витрати на виплату основної та додаткової заробітної плати таким категоріям працівників, як керівники відділів, лабораторій, секторів та груп, наукові працівники, інженерно-технічний персонал, конструктори, технологи, креслярі, копіювальники, лаборанти, робітники, студенти, аспіранти та інші працівники,

які безпосередньо зайняті у виконанні конкретної теми. Величина цих витрат обчислюється на підставі посадових окладів, відрядних розцінок та тарифних ставок відповідно до відповідних систем оплати праці.

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою [26]:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \quad (5.1)$$

де k – кількість посад дослідників залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дні;

T_p – середнє число робочих днів в місяці, T_p=21 день.

$$З_о = 30000{,}00 \cdot 60 / 21 = 81818{,}18 \text{грн.}$$

Результат обрахунків занесемо до таблиці.

Таблиця 5.4 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник проекту	30000	1392,86	60	85714,28
Інженер-розробник програмного забезпечення	25000	1173,33	60	71428,57
Всього				157142,85

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт НДР розраховуємо за формулою:

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i, \quad (5.2)$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника під час виконання роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}}, \quad (5.3)$$

де M_M – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо *Мм*=8000,00 грн;

K_i – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду (табл. Б.2, додаток Б) [27];

K_c – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об'єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

T_p – середнє число робочих днів в місяці, приблизно T_p = 22 дні;

$t_{зм}$ – тривалість зміни, год.

$C_1 = 8000,00 \cdot 1,10 \cdot 1,65 / (21 \cdot 8) = 86,43$ грн.
$З_{р1} = 86,43 \cdot 10,00 = 864,35$ грн.

Таблиця 5.5 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
Встановлення допоміжного обладнання	10	2	1,1	86,43	864,35
Тренування системи	4	4	1,5	106,30	1063,08

Інсталяція програмного забезпечення	7	4	1,5	106,30	1063,08
Всього					2990,51

Додаткова заробітна плата дослідників та робітників

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (5.4)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 11%.

$З_{дод}$ = (81818,18 + 2990,51) · 11 / 100% = 9328,95 грн.

5.2.2 Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (5.5)$$

де $Н_{зп}$ – норма нарахування на заробітну плату. Приймаємо 22%.

Зн = (81818,181+ 2990,51 + 9328,95) · 22 / 100% = 20710,28 грн.

5.2.3 Сировина та матеріали

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень.

Витрати на матеріали (*М*), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою:

$$M = \sum_{j=1}^{n} H_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} B_j \cdot Ц_{вj} \quad , \qquad (5.6)$$

де H_j – норма витрат матеріалу *j*-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу *j*-го найменування, грн/кг;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ = *1,1 … 1,15*);

B_j – маса відходів *j*-го найменування, кг;

$Ц_{вj}$ – вартість відходів *j*-го найменування, грн/кг.

$М_1 = 3 \cdot 210{,}00 \cdot 1{,}1 - 0{,}000 \cdot 0{,}00 = 693{,}0$ грн.

Проведені розрахунки зведемо до таблиці.

Таблиця 5.6 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за од	Норма витрат, од	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Папір канцелярський офісний Crystal (A4-500)	220	3	0	0	726
Канцелярське приладдя (набір офісного працівника)	175	2	0	0	385
Картридж для принтера HP LaserJet M1132 MFP	1100	1	0	0	1210
Диск оптичний VEKO-10 (CD-R)	15	3	0	0	49,5
Диск оптичний VEKO-W (CD-RW)	20	3	0	0	66
Всього					2436,5

5.2.4 Розрахунок витрат на комплектуючі

Витрати на комплектуючі *($К_в$)*, які використовують при проведенні НДР на тему «Програмні засоби для удосконалення систем керування навчання» відсутні.

5.2.5 Спецустаткування для наукових (експериментальних) робіт

До статті «Спецустаткування для наукових (експериментальних) робіт» належать витрати на виготовлення та придбання спецустаткування необхідного для проведення досліджень, також витрати на їх проектування, виготовлення, транспортування, монтаж та встановлення.

Витрати на спецустаткування, які використовують при проведенні НДР на тему «Програмні засоби для удосконалення систем керування навчання» відсутні.

5.2.6 Програмне забезпечення для наукових (експериментальних) робіт

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних для проведення досліджень, також витрати на їх проектування, формування та встановлення.

Балансову вартість програмного забезпечення розраховуємо за формулою:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \quad , \qquad (5.8)$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу даного виду, грн;

$C_{прг.i}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, (K_i = *1,10…1,12);*

k – кількість найменувань програмних засобів.

$B_{прг}$ = *11600,00 · 2 · 1,1 = 25984 грн.*

Отримані результати зведемо до таблиці:

Таблиця 5.8 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
ОС Windows 10	2	11600	25984
Прикладний пакет Microsoft Office 2019	2	5500	12320
Всього			38304

5.2.7 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12}, \quad (5.9)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$A_{обл}$ = *(35000,00 · 3) / (5 · 12) = 1750 грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 5.9 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Персональний комп'ютер	30 000	5	3	1500,00
Персональний комп'ютер	35 000	5	3	1750,00
Робоче місце дослідника	20000	5	3	1000,00
Оргтехніка	7000	4	3	437,50
Приміщення лабораторії	250000	20	3	3125,00
ОС Windows 10	25984	3	3	2165,33
Прикладний пакет Microsoft Office 2019	12320	3	3	1026,67
				11004,50

5.2.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію (B_e) розраховуємо за формулою:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot K_{вті}}{\eta_i}, \quad (5.10)$$

де W_{yi} – встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_e$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_e = 7,50$ грн;

$K_{впі}$ – коефіцієнт, що враховує використання потужності, $K_{впі} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

$B_е = 0{,}3 \cdot 480{,}0 \cdot 7{,}50 \cdot 0{,}95 / 0{,}97 = 1057{,}73$*грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 5.10 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Персональний комп'ютер	0,3	480	1057,73
Персональний комп'ютер	0,3	480	1057,73
Робоче місце дослідника	0,15	480	528,87
Оргтехніка	0,45	10	33,05
Всього			2677,38

5.2.9 Службові відрядження

До статті "Службові відрядження" дослідної роботи включаються витрати, пов'язані з організацією відряджень штатних працівників, працівників, які працюють за договорами цивільно-правового характеру, аспірантів, що зайняті проведенням досліджень. Ці витрати також охоплюють відрядження, пов'язані з випробуванням машин та приладів, а також участь у наукових з'їздах, конференціях та нарадах, які мають прямий зв'язок із виконанням конкретних досліджень.

Витрати за статтею «Службові відрядження» розраховуємо як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{св} = (З_о + З_р) \cdot \frac{H_{св}}{100\%}, \quad (5.11)$$

де $H_{св}$ – норма нарахування за статтею «Службові відрядження», приймемо $Н_{св} = 20\%$.

$B_{св}$ = (81818,18 + *2990,51)* · *20 / 100%* = 16961,73 *грн.*

5.2.10 Витрати на роботи, які виконують сторонні підприємства, установи і організації

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» розраховуємо як 30…45% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{сп} = (З_о + З_р) \cdot \frac{Н_{сп}}{100\%}, \quad (5.12)$$

де $Н_{сп}$ – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації», приймемо $Н_{сп}$= 35%.

$B_{сп}$ = (81818,18 + *2990,51)* · *35 / 100%* = 29683,04 *грн.*

5.2.11 Інші витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$І_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (5.13)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 50%.

$І_в$ = *(81818,18 + 2990,51)* · *50 / 100%* = 42404,34 *грн.*

5.2.12 Накладні (загальновиробничі) витрати

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов'язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов'язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов'язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (5.14)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 100%.

$B_{нзв}$ = *(81818,18 + 2990,51) · 100 / 100%* = 84808,68 *грн.*

Витрати на проведення науково-дослідної роботи розраховуємо як суму всіх попередніх статей витрат за формулою:

$$B_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + B_{спец} + B_{прс} + A_{обл} + B_е + B_{ов} + B_{сп} + I_в + B_{нзв}. \quad (5.15)$$

$B_{заг}$= 81818,18 + 2990,51+ 9328,95 + 20710,28 +2436,50+ 38304+ 11004,50+2677,38+ 29683,04 + 29683,04 + 42404,34 + 84808,68 = 355849.39 *грн.*

Загальні витрати *ЗВ* на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{B_{заг}}{\eta}, \quad (5.16)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η =0,9.

$$ЗВ = 355849.39 * 0,9 = 320264.45 \text{ грн.}$$

5.3 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором

У ринкових умовах узагальнюючим позитивним результатом для потенційного інвестора від можливого впровадження результатів науково-технічної розробки є збільшення чистого прибутку.

Результати дослідження передбачають комерціалізацію протягом 3-х років після введення на ринок.

У цьому випадку майбутній економічний ефект буде формуватися на основі наступних даних:

ΔN – збільшення кількості споживачів продукту, у періоди часу, що аналізуються, від покращення його певних характеристик;

1-й рік – 100 користувачів/рік;

2-й рік – 200 користувачів/рік;

3-й рік – 300 користувачів/рік.

N – кількість споживачів, які використовували аналогічний продукт у рік до впровадження результатів нової науково-технічної розробки, приймемо рівною 850 користувачам.

$Ц_б$ – вартість програмного продукту у році до впровадження результатів розробки, приймемо 35000 грн;

$\pm\Delta Ц_о$ – зміна вартості програмного продукту від впровадження результатів науково-технічної розробки, приймемо 5000,00 грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 3-х років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою:

$$\Delta П_i = (\pm \Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (5.17)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2024 році ставка податку на додану вартість складає 20%, а коефіцієнт λ *=0,8333*;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту. Приймемо ρ =30%;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2024 році ϑ *=18%*;

Збільшення чистого прибутку 1-го року:

$\Delta П_1$ = (850·5000,00+40000·100)·0,83·0,3·(1-0,18/100%)= 1690573,5 грн.

Збільшення чистого прибутку 2-го року:

$\Delta П_2$ = (850·5000,00+40000·(100+200))·0,83·0,3·(1-0,18/100%)= 3329917,5 грн.

Збільшення чистого прибутку 3-го року:

$\Delta П_3$ = (850·5000,00+40000·(100+200+300))·0,83·0,3·(1-0,18/100%)= 5788933,5 грн.

Приведена вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \quad (5.18)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =0,25;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$ПП$ = 1690573,5 /(1+0,25)^1+ 3329917,5 /(1+0,25)^2+ 5788933,5 /(1+0,25)^3= 6447539,95 грн.

Величина початкових інвестицій PV , які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{інв} \cdot ЗВ, \quad (5.19)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$ =2;

$ЗВ$ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 608629,56 грн.

$PV = k_{інв} \cdot ЗВ$ = 2· 608629,56 = 1217259,113 грн.

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV \tag{5.20}$$

де $ПП$ – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 6447539,95грн;

PV – теперішня вартість початкових інвестицій, 1217259,113 грн.

$E_{абс} = ПП - PV$ = 6447539,95 - 1217259,113 = 5230280,84грн.

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1 \tag{5.21}$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, 5230280,84грн;

PV – теперішня вартість початкових інвестицій, 1217259,113 грн;

$T_{ж}$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 3 роки.

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1$$ = (1+5230280,84/ 1217259,113)$^{1/3}$= 0,74.

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$:

$$\tau_{мін} = d + f, \quad (5.22)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2024 році в Україні d =0,1;

f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,25.

$\tau_{мін}$ = 0,11+0,25 = 0,36 < 0,74 свідчить про те, що внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою «Розробка систем керування навчання» доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_в}, \quad (5.23)$$

де $E_в$ – внутрішня економічна дохідність вкладених інвестицій.

$T_{ок}$ = 1 / 0,74 = 1,35 року.

$T_{ок}$ < 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

5.4 Висновки

Згідно проведених досліджень, рівень комерційного потенціалу розробки за темою "Програмні засоби для удосконалення систем керування навчання" становить 44 бали, що свідчить про високу комерційну важливість проведення цих досліджень.

Також термін окупності складає 1,35 року, що менше ніж 3 роки, що свідчить про комерційну привабливість науково-технічної розробки і може зацікавити потенційного інвестора у фінансуванні впровадження цієї розробки та виведенні її на ринок.

Отже, можна зробити висновок про доцільність проведення науково-дослідної роботи за темою "Розробка системи керування навчання".

ВИСНОВКИ

У магістерській кваліфікаційній роботі було проведено аналіз предметної галузі систем керування навчанням у навчанні.

Проведено аналіз стану предметної галузі та досліджено існуючі аналоги. Було обґрунтовано вибір мови програмування для реалізації розроблюваного додатку та поставлені задачі для системи.

Розроблено метод оцінювання успішності студентів, особливість якого полягає у використанні часових обмежень без участі ментора у виставлені виконаних завдань кожного студента, що дало можливість покращити якість дистанційного навчання.

Розроблено алгоритми та програмні засоби для систем керування навчанням.

Розроблено інтерфейс додатку для якого було вибрано інтерфейс командного рядку, та бібліотеку “wca” для його реалізації. Також було автоматизовано найбільш поширені задачі системи.

Проведено інтеграційне тестування системи та її взаємодії з іншими сервісами. Отримані результати підтвердили ефективність і стабільність системи.

Отримані в магістерській роботі наукові та практичні результати можна використати для розробки системи керування навчання та основі автоматизованого оцінювання успішності студентів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Войтенко Д. О., Романюк О. Н., Системи Управління Навчанням. Матеріали Десята Всеукраїнська науково-практична інтернет-конференція молодих вчених та студентів «Сучасні інформаційні системи та технології»
2. Організація дистанційного навчання. Створення електронних навчальних курсів та електронних тестів. Вишнівський В.В., Гніденко М.П., Гайдур Г.І., Ільїн О.О. 2014р. 140с
3. Синергетичні принципи освіти та науки. О.В. Чалий. 2000р. 253с
4. Дистанційний навчальний процес:Навчальний посібник. Биков В.Ю, Кухаренко В.М. 2005р. 101с.
5. Дистанційне навчання: Умови застосування. Кухаренко В.М., Рибалко О.В., Сиротенко Н.Г. 2002р. 320с.
6. Дистанційне навчання у схемах. Кухаренко В. М., Сиротенко Н. Г. 2001р. 64с.
7. Використання системи електронного навчання MOODLE для контролю і оцінювання навчальної діяльності студентів ВНЗ. Ю.В. Триус, І.В. Стеценко, Л.П. Оксамитна, В.М. Франчук, І.В. Герасименко. 2010р. 200с.
8. Посібник з підготовки та організації електронного навчання. Катерняк І. 2016р. 48 с.
9. Think Python: How to Think Like a Computer Scientist. Downey, Allen B. 2012р. 270с.
10. Learning Python (5th ed.). Lutz, Mark. 2013р. 1214с.
11. C# Language Pocket Reference. Peter Drayton, Ben Albahari, Ted Neward. 2006 р. 128 с.

12. Microsoft Visual C#.NET: Language Reference. Microsoft Press, 2002p. 393 c.
13. A history of C++: 1979-1991. Stroustrup, Bjarne. 1996p. 8c.
14. Modern C++ Design: Generic Programming and Design Patterns Applied. Andrei Alexandrescu. 2001p. 323c.
15. Java's 20 Years Of Innovation. Binstock, Andrew. 2015p. 1c.
16. Program Development in Java: Abstraction, Specification, and Object-oriented Design. Barbara Liskov, John Guttag. 2001p. 443 c.
17. Programming Rust: Fast, Safe Systems Development. Jim Blandy, Jason Orendorff, Leonora F. S. Tindall. 2021p. 711c.
18. Software Pioneers: Contributions to Software Engineering. M. Broy. 2002 p. 728 стор.
19. Structured Analysis and System Specification. Tom DeMarco. 1979p. 352c.
20. Structure Charts and Basic Programming. Sarah Brooks. 1981p. 112c.
21. Shell Scripting: Expert Recipes for Linux, Bash and more. Parker, Steve. 2011p. 262c.
22. Graphical user interfaces. Wendy L. Martinez. 2011p. 150c.
23. The Open Group Base Specifications Issue 7, 2018 edition. The Open Group. 2018p. 600c.
24. Software Testing. Milind G. Limaye. 2009p. 523c.
25. Pragmatic Software Testing: Becoming an Effective and Efficient Test Professional. Rex Black. 2011p. 384c.
26. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. Вінниця : ВНТУ, 2021. 42 с.

27. Кавецький В. В. Економічне обґрунтування інноваційних рішень: практикум / В. В. Кавецький, В. О. Козловський, І. В. Причепа. Вінниця : ВНТУ, 2016. 113 с.

ДОДАТКИ

81

Додаток А - Технічне завдання

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ

д.т.н., проф. О. Н. Романюк

«12» березня 2024 р.

Технічне завдання

на магістерську кваліфікаційну роботу

«Програмні засоби для удосконалення систем керування навчанням»

за спеціальністю

121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи:

д.т.н., проф. О.Н. Романюк

" 12 " березня 2024 р.

Виконав:

студент гр.ІП-22мз Д.О. Войтенко

" 12 " березня 2024 р.

Вінниця – 2024 року

11.06.2024 09:41

1. Найменування та галузь застосування

Магістерська кваліфікаційна робота: «Програмні засоби для удосконалення систем керування навчанням».

Галузь застосування – освітній процес.

2. Підстава для розробки.

Підставою для виконання магістерської кваліфікаційної роботи (МКР) є індивідуальне завдання на МКР та наказ № 81 від 11 березня 2024 ректора по ВНТУ про закріплення тем МКР.

3. Мета та призначення розробки.

Мета магістерської кваліфікаційної роботи полягає в покращенні процесу навчання за рахунок автоматизації одноманітних процесів та систематизації даних успішності студентів та покращення якості навчання.

Призначення роботи – розробка методу автоматичної оцінки успішності студентів.

4 Вихідні дані для проведення МКР

Перелік основних літературних джерел, на основі яких буде виконуватись МКР.

1. Організація дистанційного навчання. Створення електронних навчальних курсів та електронних тестів. Вишнівський В.В., Гніденко М.П., Гайдур Г.І., Ільїн О.О. 2014р. 140с
2. Синергетичні принципи освіти та науки. О.В. Чалий. 2000р. 253с

3. Дистанційний навчальний процес:Навчальний посібник. Биков В.Ю, Кухаренко В.М. 2005р. 101с.

4. Дистанційне навчання: Умови застосування. Кухаренко В.М., Рибалко О.В., Сиротенко Н.Г. 2002р. 320с.

5. Дистанційне навчання у схемах. Кухаренко В. М., Сиротенко Н. Г. 2001р. 64с.

5. Технічні вимоги

Вихідні дані до роботи: Мова програмування - Rust, бази навчальної роботи, типи сервісів Github/Google Calendar/Google Meet, тип подачі даних - конфігураційні файли.

6. Конструктивні вимоги.

Конструкція пристрою повинна відповідати естетичним та ергономічним вимогам, повинна бути зручною в обслуговуванні та керуванні.

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

– пояснювальна записка до МКР;

– технічне завдання;

– лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки:

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи
1	Обґрунтування розробки	12.03.2024 – 28.03.2024
2	Розробка методу автоматичної оцінки успішності студентів	29.03.2024 – 11.03.2024
3	Розробка програмних засобів для покращення систем керування навчанням	12.03.2024– 12.04.2024
4	Тестування роботи системи	13.04.2024– 05.06.2024
5	Економічна частина	26.04.2024 - 10.05.2024

10. Порядок контролю та прийняття.

Виконання етапів магістерської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи. Прийняття магістерської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком.

Додаток Б. ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи: «Програмні засоби для удосконалення систем керування навчання»

Тип роботи: магістерська кваліфікаційна робота

Підрозділ : кафедра програмного забезпечення, ФІТКІ, ПІ – 22мз

Науковий керівник:

Unicheck	
Оригінальність	
Схожість	

Аналіз звіту подібності

■ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

 Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

 Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Черноволик Г. О.

Опис прийнятого рішення: **допустити до захисту**

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи _____ Войтенко Д.О.

Керівник роботи _____ Романюк О. Н.

Додаток В. Лістинг команд CLI

```
#![ warn( rust_2018_idioms ) ]
#![ warn( missing_debug_implementations ) ]
#![ warn( missing_docs ) ]

//!
//! Learning progress analysis tool.
//!

#![ doc = include_str!( concat!( env!( "CARGO_MANIFEST_DIR" ), "/", "Readme.md" ) ) ]

use rustyline::DefaultEditor;

use lt::*;


use wca::{ GrammarConverter, Command };
use wtools::Itertools;


fn generate_help_content( grammar : &GrammarConverter, command : Option< &Command > ) -> String
{
  if let Some( command ) = command
  {
    let name = &command.phrase;
    let hint = if command.long_hint.is_empty() { &command.hint } else { &command.long_hint };
    let subjects = command.subjects.iter().enumerate().fold( String::new(), | acc, ( number, subj ) | format!( "{acc} <subject_{number}:{:?}>", subj.kind ) );
    let full_subjects = command.subjects.iter().enumerate().map( | ( number, subj ) | format!( "subject_{number} - {} [{:?}]", subj.hint, subj.kind ) ).join( "\n\t" );
```

```
        let properties = if command.properties.is_empty() { " " } else { "
<properties> " };
        let full_properties = command.properties.iter().sorted_by_key( | ( name, _ )
| *name )
        .map( | ( name, value ) |
        {
          if value.optional
          {
            format!( "?{name} - {} [{:?}]", value.hint, value.kind )
          }
          else
          {
            format!( "{name} - {} [{:?}]", value.hint, value.kind )
          }

        } ).join( "\n\t" );

        format!( "{name}{subjects}{properties}- {hint}\n{}{}",
        if command.subjects.is_empty() { String::new() } else
{ format!( "\nSubjects:\n\t{}", &full_subjects ) },
        if command.properties.is_empty() { String::new() } else
{ format!( "\nProperties:\n\t{}",&full_properties ) }, )

      }
      else
      {
        grammar.commands
        .iter()
        .sorted_by_key( | ( name, _ ) | *name )
        .map
        ( | ( name, cmd ) |
          {
            cmd.iter().fold
            ( String::new(), | _, cmd |
```

```
                {
                    let subjects = cmd.subjects.iter().fold( String::new(), | _, subj | format!( " <{:?}>", subj.kind ) );
                    let properties = if cmd.properties.is_empty() { " " } else { " <properties> " };
                    let hint = if cmd.hint.is_empty() { &cmd.long_hint } else { &cmd.hint };

                    let left = format!( ".{name}{subjects}{properties}" );
                    format!( "{left:<55} - {hint}" )

                }
              )
            }
          )
          .fold
          ( String::new(), | acc, cmd |
            {
              format!( "{acc}\n{cmd}" )
            }
          )
        }
      }

      // /// Command request from user
      // fn command_line_asking( request : &str ) -> String
      // {
      //   let mut response = String::new();
      //   print!( "{request} : " );
      //   io::stdout().flush().ok();
      //   io::stdin().read_line( &mut response ).ok();
      //   response.trim().to_string()
      // }
```

```
fn main()
{
  let editor = std::rc::Rc::new( std::cell::RefCell::new( DefaultEditor::new().expect( "Failed to init editor" ) ) );
  let _ = editor.borrow_mut().load_history( &dirs::home_dir().unwrap_or_default().join( ".lt_history.txt" ) );

  let lamda_editor = std::rc::Rc::clone( &editor );

  let ca = wca::CommandsAggregator::former()
  .grammar( grammar_form() )
  .executor( executor_form() )
  .callback
  (
    move | input, _program |
    {
      let _ = lamda_editor.borrow_mut().add_history_entry( input );
      let _ = lamda_editor.borrow_mut().save_history( &dirs::home_dir().unwrap_or_default().join( ".lt_history.txt" ) );
    } )
  .help( generate_help_content)
  .build();

  let args = std::env::args().skip( 1 ).collect::< Vec< String > >();
  if args.is_empty()
  {
    loop
    {
      let command = editor.borrow_mut().readline( ">" ).expect( "Failed to readline" );
```

```
            match ca.perform( command.clone() )
            {
              Ok( _ ) => (),
              Err( e ) => println!( "{e}" )
            }
            if command == ".exit"
            {
              return;
            }
          }
        }
        else
        {
          ca.perform( args.join( " " ).as_str() ).unwrap();
        }
      }

       #![cfg(not(doctest))]
      #![ doc = include_str!( concat!( env!( "CARGO_MANIFEST_DIR" ), "/", "Readme.md" ) ) ]

      pub use commands::form::{ grammar_form, executor_form };

      /// Commands of utility.
      mod commands;
      pub(crate) mod global_filter;

      use wca::{Args, Props, Type, wtools::BasicError};
      use lt_logic::endpoint::calendar;

      pub( crate ) const CALENDAR_DEADLINES_PHRASE : &str = "calendar.deadlines";
```

```
/// Declare command
pub(crate) fn calendar_deadlines_command() -> wca::Command
{
  wca::Command::former()
  .hint("Create deadline events")
  .long_hint
  (
    "Create and add to config file deadline events.

    ### Example

    ```
    .calendar.deadlines dry:0
    ```

    ### Config requirement

    To execute this command, you need to set some fields.
    You can set these fields according to the examples:
    chapters_soft_deadlines = [\"06/07/2023\", \"20/07/2023\", \"03/08/2023\", \"24/08/2023\", \"07/09/2023\", \"28/09/2023\", \"05/10/2023\"]"
)
  .phrase(CALENDAR_DEADLINES_PHRASE)
  .property("dry", "dry testing. It displays what the command will do without actually executing it ", Type::Number, true)
  .form()

}

pub(crate) fn calendar_deadlines((_, props) : (Args, Props)) -> wca::wtools::Result< () >
{

  let dry = match props.get_owned("dry")
```

```
        {
          Some( 1 ) | None => true,
          Some( 0 ) => false,
          Some( i ) =>  Err( BasicError::new( format!( "Error getting dry property: {i:?}"  ) ) )?
        };

        let result = calendar::deadline_events_create_endpoint( dry )?;
        println!( "{result}" );

        Ok( () )
      }

      use lt_logic::endpoint::calendar::calendar_events_wrapper;
      use wca::{ Args, Props, Type, wtools::BasicError };

      use crate::global_filter::groups_get;

      pub( crate ) const CALENDAR_EVENTS_PHRASE : &str = "calendar.events";

      /// Declarate command
      pub( crate ) fn calendar_events_command() -> wca::Command
      {
        wca::Command::former()
        .hint( "Get list of calendar events" )
        .long_hint( "This command will get bootcamp events from calendar and display them

        ### Example

        ```
        .calendar.events filter:M1
        ```
```

```
        ### Config requirement

        To execute this command, you need to set some fields.
        You can set these fields according to the examples:
        `calendar.id = \"c_350e9bbebaa4a9344e0cb0c088bedeebec5bc77bd3b8b35c6da693cdf7befb6f@group.calendar.google.com\"`
        `gcp_cloud.refresh_token = \"1//0cD5VgXzPyu1QCgYIARAAAAwSNwF-L5IrqzKw9ma4z_qtUI543FKrRAceIYi4LnEUGoBrWYe4njK2npTKXF4-tWwg0VnBdduh6zQ\"`
        `gcp_cloud.client_secret = \"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\"`
        `gcp_cloud.client_id = \"238696131479-r7qk24crlsr5mdnqb55hkim6vgqvusfo.apps.googleusercontent.com\"`" )
        .phrase( CALENDAR_EVENTS_PHRASE )
        .property( "filter", "free text search terms to find events that match these terms in any field, except for extended properties", Type::String, true )
        .form( )

      }

      pub( crate ) fn calendar_events( ( _, props ) : ( Args, Props ) ) -> wca::wtools::Result< () >
      {

        let filter : String = props.get_owned( "filter" ).unwrap_or_default();

        let groups = groups_get()
        .map_err( | e | BasicError::new( e.to_string() ) )?;

        let rt = tokio::runtime::Runtime::new().unwrap();
```

```
    let result = rt.block_on( calendar_events_wrapper( groups.into_iter(), &filter ) )?;

    result.into_iter().for_each( | this | println!( "{this}" ) );

    Ok( () )
  }
  use lt_logic::endpoint::calendar::calendar_list_wrapper;
  use wca::{ Args, Props, wtools::BasicError };

  use crate::global_filter::groups_get;

  pub( crate ) const CALENDAR_LIST_PHRASE : &str = "calendar.list";

  /// Declarate command
  pub( crate ) fn calendar_list_command() -> wca::Command
  {
    wca::Command::former()
    .hint( "Get list of calendars" )
    .long_hint( "This command will get calendars from google api and display them

    ### Example

    ```
    .calendar.list
    ```

    ### Config requirement

    To execute this command, you need to set some fields.
    You can set these fields according to the examples:
    `gcp_cloud.refresh_token   = \"1//0cD5VgXzPyu1QCgYIARAAAAwSNwF-
```

```
L5IrqzKw9ma4z_qtUI543FKrRAceIYi4LnEUGoBrWYe4njK2npTKXF4-
tWwg0VnBdduh6zQ\"`
        `gcp_cloud.client_secret = \"GOCSPX-
inmYUGqCzyjw5Uipu1f8JI5lN9ZS\"`
        `gcp_cloud.client_id = \"238696131479-
r7qk24crlsr5mdnqb55hkim6vgqvusfo.apps.googleusercontent.com\"`" )
        .phrase( CALENDAR_LIST_PHRASE )
        .form( )

      }

      pub( crate ) fn calendar_list( ( _, _ ) : ( Args, Props ) ) -> wca::wtools::Result<
() >
      {

        let groups = groups_get()
        .map_err( | e | BasicError::new( e.to_string() ) )?;

        let rt = tokio::runtime::Runtime::new().unwrap();

        let result = rt.block_on( calendar_list_wrapper( groups.into_iter() ) )?;

        result.into_iter().for_each( | this | println!( "{this}" ) );

        Ok( () )
      }

      use wca::{ Args, Props, wtools::BasicError };
      use lt_logic::endpoint::{curriculum::curriculum_about_wrapper,
table::IntoTable};

      use crate::global_filter::groups_get;
```

```
pub( crate ) const CURRICULUM_ABOUT_PHRASE : &str = "curriculum.about";

/// Declarate command
pub( crate ) fn curriculum_about_command() -> wca::Command
{
  wca::Command::former()
  .hint( "Get curriculum topology" )
  .long_hint( "This command collects information about the curriculum topology from the TOML file.
  It shows how many sections the curriculum has, and how many tasks are in each chapter.

  ### Example

  ```
  .curriculum.about
  ```

  +-----------+-----------+
  | Group | 2023q3_M2 |
  +-----------+-----------+
  | Chapter № | Tasks |
  +-----------+-----------+
  | 0 | 1 |
  +-----------+-----------+
  | 1 | 10 |
  +-----------+-----------+
  | 2 | 7 |
  +-----------+-----------+
  | 3 | 12 |
  +-----------+-----------+
  | 4 | 4 |
  +-----------+-----------+
```

```
  | 5        | 7        |
  +----------+----------+
  | 6        | 4        |
  +----------+----------+
  | Total    | 45       |
  +----------+----------+

  ### Config requirement

  To execute this command, you need to set some fields.
  You can set these fields according to the examples:
  `curriculum.chapters_topology = [[0,1],[0,10],[0,5],[0,13],[1,10]]`
  `curriculum.tasks_topology = [[], [], [], [], [0,0,2,1,1,1,1,0,0,1]]`
  " )
  .phrase( CURRICULUM_ABOUT_PHRASE )
  .form()

 }

 pub( crate ) fn curriculum_about( ( _, _ ) : ( Args, Props ) ) -> wca::wtools::Result< () >
 {
  let groups = groups_get()
  .map_err( | e | BasicError::new( e.to_string() ) )?;

  let table = curriculum_about_wrapper( groups.into_iter() )?;

  table.into_iter().for_each( | this | this.into_table().printstd() );

  Ok( () )
 }

 use wca::{ Args, Props };
```

```
pub( crate ) const EXIT_PHRASE : &str = "exit";

/// Declarate command
pub( crate ) fn exit_command() -> wca::Command
{
  wca::Command::former()
  .hint( "Close programm" )
  .long_hint( "Command that terminates the program

  ### Example

  ```
  .exit
  ```" )
  .phrase( EXIT_PHRASE )
  .form()
}

pub( crate ) fn exit( ( _, _ ) : ( Args, Props ) ) -> wca::wtools::Result< () >
{
  println!( "Close programm" );
  Ok( () )
}
```

Додаток Г. Лістинг модуля взаємодії з Github

```
//! Methods for interaction with Github Forks API.

use serde_derive::Deserialize;
use serde_json::json;

use crate::{errors::GitHubError, credentials::GithubCreds};

/// Owner of forked repository. In our cause this is `rust-lang-ua`
#[derive(Debug, Clone)]
pub struct RepositoryOwner< 'a >( pub &'a str);

/// Name that fork should have.
#[derive(Debug, Clone)]
pub struct RepositoryName< 'a >( pub &'a str);

/// Stucture representing repository
#[derive(Debug, Clone)]
pub struct Repo< 'a >
{
  /// Owner of repository
  pub owner : RepositoryOwner< 'a >,
  /// name of repository
  pub repo : RepositoryName< 'a >
}

/// Check is if repository is template
pub async fn is_repo_template(creds : &GithubCreds, repo : &Repo< '_ > ) -
> Result< bool, GitHubError >
{
  #[ derive(Debug, Deserialize ) ]
  struct Response
  {
```

```
  pub is_template : bool
 }

 let client = reqwest::ClientBuilder::new().build()?;

 let response = client
 .get( format!( "{}repos/{}/{}", creds.github_url, repo.owner.0, repo.repo.0 ) )
 .header( "Authorization", format!( "Bearer {}", creds.github_token ) )
 .header( "Accept", "application/vnd.github+json" )
 .header( "X-GitHub-Api-Version", "2022-11-28" )
 .header( "User-Agent", &creds.user_agent )
 .send()
 .await?;

 if !response.status().is_success()
 {
  Err( GitHubError::NotFound( format!( "The {}/{} repository not exist or other.", repo.owner.0, repo.repo.0 ) ) )?
 }

 let response : Response = response.json().await?;

  Ok( response.is_template )
 }

 /// Create fork into organization from `fork_to` parameter
 pub async fn fork_create(creds : &GithubCreds, fork_from : &Repo< '_ >, fork_to : &Repo< '_ > ) -> Result< (), GitHubError >
 {
  if is_exists( creds, &fork_to.owner, &fork_to.repo ).await?
  {
   Err( GitHubError::NotFound( format!( "The {}/{} repository already exist or other.", fork_to.owner.0, fork_to.repo.0 ) ) )?
```

```
    }

    let client = reqwest::ClientBuilder::new().build()?;

    let response = client
    .post( format!( "{}repos/{}/{}/forks", creds.github_url, fork_from.owner.0, fork_from.repo.0 ) )
    .header( "Authorization", format!( "Bearer {}", creds.github_token ) )
    .header( "Accept", "application/vnd.github+json" )
    .header( "X-GitHub-Api-Version", "2022-11-28" )
    .header( "User-Agent", &creds.user_agent )
    .json
    (
      &json!
      (
        {
          "organization" : fork_from.owner.0,
          "name" : fork_to.repo.0,
          "default_branch_only" : true
        }
      )
    )
    .send()
    .await?;

    if response.status().is_server_error()
    {
      Err( GitHubError::CustomError("Github server issue. Try again later.".to_string() ) )?
    }
    if response.status().is_client_error()
    {
      Err( GitHubError::CustomError( format!( "Failed to create fork: {} : {}", fork_to.repo.0, response.text().await? ) ) )?
```

```
    }

    Ok( () )
  }

  /// Change visibilty of repository
  pub async fn repo_visibility_change(creds : &GithubCreds, repo : &Repo< '_
>, is_private : bool ) -> Result<(), GitHubError >
  {
    let visibility = match is_private
    {
      true => "private",
      false => "public",
    };

    let client = reqwest::ClientBuilder::new().build()?;

    client
    .patch( format!( "{}/repos/{}/{}", creds.github_url, repo.owner.0,
repo.repo.0 ) )
    .header( "Authorization", format!( "Bearer {}", creds.github_token ) )
    .header( "Accept", "application/vnd.github+json" )
    .header( "X-GitHub-Api-Version", "2022-11-28" )
    .header( "User-Agent", &creds.user_agent )
    .json( &json!( { "private" : visibility } ) )
    .send()
    .await?;

    Ok( () )
  }

  /// Check if repository exists already
```

```
    pub async fn is_exists(creds : &GithubCreds, owner : &RepositoryOwner<'_>, repo : &RepositoryName<'_> ) ->  Result< bool, GitHubError >
    {
      let client = reqwest::ClientBuilder::new().build()?;

      let response = client
      .get( format!( "{}/repos/{}/{}", creds.github_url, owner.0, repo.0 ) )
      .header( "Authorization", format!( "Bearer {}", creds.github_token ) )
      .header( "Accept", "application/vnd.github+json" )
      .header( "X-GitHub-Api-Version", "2022-11-28" )
      .header( "User-Agent", &creds.user_agent )
      .send()
      .await?;

      Ok( response.status().is_success() )
    }



    //! Credentials struct

    use derive_builder::Builder;

    /// builder method
    #[ allow( clippy::single_call_fn ) ]
    fn default_github() -> String
    {
      "https://api.github.com".to_owned()
    }

    /// builder method
    #[ allow( clippy::single_call_fn ) ]
    fn default_graphql_github_url() -> String
    {
```

```
  "https://api.github.com/graphql".to_owned()
 }

 /// Strcut with credentials required to access github api
 #[derive(Debug, Clone, Builder)]
 pub struct GithubCreds
 {
  /// Url to github rest api
  #[builder(default = "default_github()" )]
  pub github_url : String,
  /// Url to github GraphQL api
  #[builder(default = "default_graphql_github_url()" )]
  pub github_graph_url : String,
  /// Bearer auth token. Example header `Authorization : Bearer <your_token>`
  pub github_token : String,
  /// Github request that you use your GitHub username, or the name of your application, for the User-Agent header value.
  pub user_agent : String,
 }
```

Додаток Д Лістинг модуля взаємодії з Google Calendar

```
use chrono::{SecondsFormat, TimeZone, Utc};
use chrono_tz::{Tz, OffsetComponents};
use google_calendar::{Client, types::EventAttendee, ClientError};
use google_calendar::types::{ConferenceData, CreateConferenceRequest,
EventDateTime, EventReminder, MinAccessRole, Reminders, SendUpdates};
use rand::distributions::{Alphanumeric, DistString};
use rand::thread_rng;
use serde_email::Email;
use crate::errors::CalendarError;
use crate::event::CalendarEvent;
use crate::event::{Conference, EventId};

pub use google_calendar::types::{Event, CalendarListEntry};

#[ derive( Debug ) ]
/// Options of calendar
pub struct CalendarOptions
{
  /// Id of client
  pub client_id : String,
  /// Secret
  pub client_secret : String,
  /// Refresh token for api
  pub refresh_token : String,
  /// Id of used calendar
  pub calendar_id : String,
  /// Override api host
  pub host_override : Option<String>,
}

#[ derive( Debug ) ]
/// Wrapper for google api
```

```
pub struct Calendar
{
  /// Options of calendar
  options : CalendarOptions,
}
impl Calendar
{
  /// Create new
  pub fn new
  (
    options : CalendarOptions
  ) -> Self
  {
    Self { options }
  }

  /// Get list of all events
  ///
  /// # Arguments
  ///
  /// * `filter` - free form text for event search by name
  /// * `show_deleted`
  pub async fn list_of_calendar_events( &self, filter : &str, show_deleted :
bool )
  -> Result< Vec< Event >, ClientError >
  {
    let google_calendar = self.client_create().await?;

    let events_api = google_calendar.events();


    let list = events_api.list_all
    (
      &self.options.calendar_id,
```

```
            "", // Optional. Use this if you want to search for an event by its iCalendar ID. Use event_get instead
            0, // Return all attenders
            google_calendar::types::OrderBy::default(),
            &[], // Search by private property. Ignore
            filter,
            &[], // Search by private property. Ignore
            show_deleted, // Show only exists events
            false, // Whether to include hidden invitations in the result.
            false, // Return event with `recurring` field instead of list of event
            "", // Upper bound (exclusive) for an event's start time to filter by.
            &chrono::Utc::now().to_rfc3339_opts(SecondsFormat::Secs, true), // ignore events in past
            "", // Use time_zone of calendar
            "", // Get events regardless their updates
          ).await?;

          Ok( list.body )
        }

        /// Get list of all events
        pub async fn all_calendar_events( &self, filter : &str, show_deleted : bool )
        -> Result< Vec< CalendarEvent >, CalendarError >
        {

          let google_calendar = self.client_create().await
          .map_err( CalendarError::ClientError )?;

          let events = google_calendar.events();

          let list = events.list_all
          (
            &self.options.calendar_id,
```

```
      "", // Optional. Use this if you want to search for an event by its iCalendar ID. Use event_get instead
      0, // Return all attenders
      google_calendar::types::OrderBy::default(),
      &[], // Search by private property. Ignore
      filter,
      &[], // Search by private property. Ignore
      show_deleted, // Show only exists events
      false, // Whether to include hidden invitations in the result.
      false, // Return event with `recurring` field instead of list of event
      "", // Upper bound (exclusive) for an event's start time to filter by.
      &chrono::Utc::now().to_rfc3339_opts(SecondsFormat::Secs, true), // ignore events in past
      "", // Use time_zone of calendar
      "", // Get events regardless their updates
    ).await.map_err( CalendarError::ClientError )?;


    let mut mapped_api_events = Vec::< CalendarEvent >::new();
    for event in list.body
    {
      mapped_api_events.push
      (
        event.try_into()?
      );
    }

    Ok( mapped_api_events )
  }

  /// Remove attender from event
  pub async fn attenders_remove_from_calendar
  (
    &self,
```

```
  entities: impl Iterator< Item = &Email >,
  events : impl Iterator< Item = &mut Event >,
) -> Result< (), ClientError >
{
  let google_calendar = self.client_create().await?;
  let event_api = google_calendar.events();

  let entities = entities.collect::< Vec< _ > >();

  for event in events
  {
    for entity in entities.clone()
    {
      for ( num, attender) in event.attendees.iter().enumerate()
      {
        if attender.email == entity.as_str()
        {
          event.attendees.remove(num);
          break;
        }
      }

      event_api.patch
      (
        &self.options.calendar_id,
        &event.id,
        0,
        0,
        false,
        SendUpdates::default(),
        false,
        event
      ).await?;
    }
```

```
  }

  Ok( () )
}

/// Add attender to event
pub async fn attenders_add_to_calendar
(
  &self,
  entities: impl Iterator< Item = &Email >,
  events : impl Iterator< Item = &mut Event > + Clone
) -> Result< (), ClientError >
{
  let google_calendar = self.client_create().await?;
  let event_api = google_calendar.events();

  for entity in entities
  {
    for event in events.clone()
    {
      event.attendees.push
      (
        EventAttendee
        {
          additional_guests: 0,
          comment: String::new(),
          display_name: String::new(),
          email: entity.to_string(),
          id: String::new(),
          optional: false,
          organizer: false,
          resource: false,
          response_status: "needsAction".to_string(),
          self_: false,
```

```
      }
    );

    event_api.patch
    (
      &self.options.calendar_id,
      &event.id,
      0,
      0,
      false,
      SendUpdates::default(),
      false,
      event
    ).await?;
   }
  }

  Ok( () )
}

/// helper method for creating client
async fn client_create( &self ) -> Result< Client, ClientError >
{
  let mut client = Client::new
  (
    self.options.client_id.clone(),
    self.options.client_secret.clone(),
    String::new(),
    String::new(),
    self.options.refresh_token.clone()
  );

  if let Some( host ) = self.options.host_override.clone()
  {
```

```
    client.with_host_override( host );
   }

   if cfg!( not( feature="calendar_token" ) )
   {
    client.refresh_access_token().await?;
   }

   Ok( client )
  }

  /// Create events in calendar
  pub async fn events_create( &self, calendar_events : impl Iterator< Item = &CalendarEvent > )
   -> Result< (), ClientError >
  {
   let google_calendar = self.client_create().await?;

   let events = google_calendar.events();

   for event in calendar_events
   {

    let google_event = event_create( event );

    events.insert
    (
     self.options.calendar_id.as_str(),
     1, //Version 0 assumes no conference data support and ignores conference data in the event's body.
     100,
     true,
     SendUpdates::default(),
     true,
```

```
            &google_event,
          ).await?;
        }

        Ok( () )
      }

      /// Update events in calendar
      pub async fn events_update( &self, calendar_events : impl Iterator< Item = &CalendarEvent >, send_notif : bool )
      -> Result< (), ClientError >
      {
        let google_calendar = self.client_create().await?;

        let events = google_calendar.events();

        for event in calendar_events
        {
          let google_event = event_create( event, );

          events.patch
          (
            self.options.calendar_id.as_str(),
            google_event.id.as_str(),
            1,
            100,
            send_notif,
            SendUpdates::default(),
            true,
            &google_event,
          ).await?;
        }

        Ok( () )
```

```
  }

  /// Delete events in calendar
  pub async fn events_delete( &self, calendar_events : impl Iterator< Item =
&EventId > )
   -> Result< (), ClientError >
  {
   let google_calendar = self.client_create().await?;

   let events = google_calendar.events();

   for event in calendar_events
   {
    events.delete
    (
     self.options.calendar_id.as_str(),
     &event.0,
     false,
     SendUpdates::None,
    ).await?;
   }

   Ok( () )
  }

  /// Get specific event from calendar
  pub async fn event_get( &self, id : &EventId, time_zone : &Tz )
   -> Result< Event, ClientError >
  {
   let google_calendar = self.client_create().await?;

   let events = google_calendar.events();

   let event = events.get
```

```
      (
        &self.options.calendar_id,
        id.0.as_str(),
        0,
        time_zone.to_string().as_str(),
      ).await?;

      Ok( event.body )
    }

    /// Get list of all user calendars calendars
    pub async fn calendar_list( &self ) -> Result< Vec< CalendarListEntry >, CalendarError >
    {
      let google_calendar = self.client_create().await
      .map_err( CalendarError::ClientError )?;
      let user_calendar = google_calendar.calendar_list();

      let calendars = user_calendar.list_all
      (
        MinAccessRole::Reader,
        false,
        false
      ).await.map_err( CalendarError::ClientError )?;

      let body = calendars.body;
      Ok( body )
    }

    /// Create new user calendar
    pub async fn calendar_create( &self, summary : &str ) -> Result< google_calendar::types::Calendar, CalendarError >
    {
      let google_calendar = self.client_create().await
```

```
    .map_err( CalendarError::ClientError )?;
    let user_calendar = google_calendar.calendars();

    let entity = google_calendar::types::Calendar
    {
      conference_properties: None,
      description: "".to_string(),
      etag: "".to_string(),
      id: "".to_string(),
      kind: "".to_string(),
      location: "".to_string(),
      summary:  summary.to_string(),
      time_zone: "".to_string(),
    };

    let result = user_calendar.insert( &entity )
    .await.map_err( CalendarError::ClientError )?;

    let body = result.body;

    Ok( body )
  }
}

/// Create rand string
fn rand_str() -> String
{
  let mut rand = thread_rng();

  Alphanumeric.sample_string( &mut rand, 10 )
}

/// Create event
pub fn event_create( event : &CalendarEvent ) -> Event
```

```
        {
          let conference = match event.conference
          {
            Conference::Create => Some
            (
              ConferenceData
              {
                conference_id: String::new(),
                conference_solution: None,
                create_request: Some
                (
                  CreateConferenceRequest
                  {
                    conference_solution_key: None,
                    request_id: rand_str(),
                    status: None,
                  }
                ),
                entry_points: vec![],
                notes: String::new(),
                parameters: None,
                signature: String::new(),
              }
            ),
            Conference::None => None,
          };

          let offset = event.time_zone.offset_from_utc_date( &Utc::now().date_naive() );
          let dur = offset.base_utc_offset();

          let event = Event
          {
            attendees: event.emails.iter().map( | this | EventAttendee
```

```
{
  additional_guests: 0,
  comment: String::new(),
  display_name: String::new(),
  email: this.to_string(),
  id: String::new(),
  optional: false,
  organizer: false,
  resource: false,
  response_status: "needsAction".to_string(),
  self_: false,
} ).collect::< Vec< _ > >(),
conference_data: conference,
description: event.description.clone(),
end: Some
(
  EventDateTime
  {
    date: None,
    date_time: Some( event.end_datetime - dur ),
    time_zone: event.time_zone.to_string(),
  }
),
event_type: "default".to_string(),
id: event.id.0.clone(),
kind: "calendar#event".to_string(), // only one possible value
location: event.location.clone(),
recurrence: event.recurrence.clone(),
recurring_event_id: String::new(),
reminders: Some(
  Reminders
  {
    overrides: event.reminders.iter().map
    (
```

```
                | this | EventReminder { method: this.1.as_ref().to_string(), minutes: this.0 }
              ).collect::< Vec< _> >(),
              use_default: false
            }
          ),
          start: Some
          (
            EventDateTime
            {
              date: None,
              date_time: Some( event.start_datetime - dur ),
              time_zone: event.time_zone.to_string(),
            }
          ),
          status: event.status.as_ref().to_string(),
          summary: event.summary.clone(), transparency: event.transparency.clone().unwrap_or_default().as_ref().to_string(),
          visibility: event.visibility.clone().unwrap_or_default().as_ref().to_string(),
          ..Default::default()
        };
        event
      }
```

Додаток Ж

ІЛЮСТРАТИВНА ЧАСТИНА

РОЗРОБКА ПРОГРАМНИХ ЗАСОБІВ ДЛЯ УДОСКОНАЛЕННЯ СИСТЕМ КЕРУВАННЯ НАВЧАННЯМ

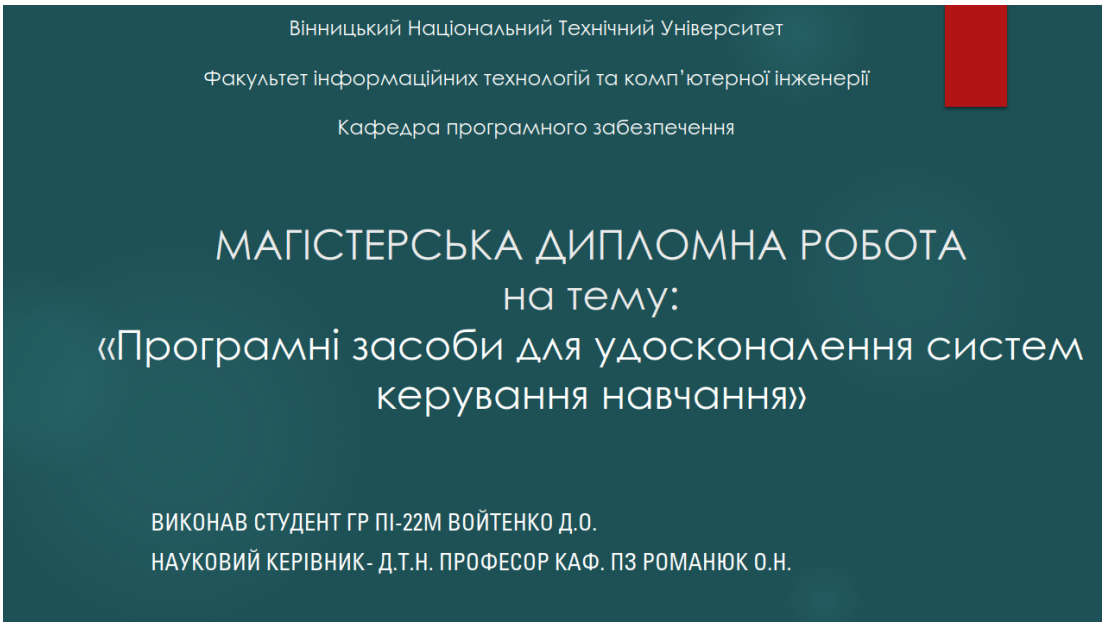

Рисунок В.1 – Слайд презентації № 1

Рисунок В.2 – Слайд презентації № 2

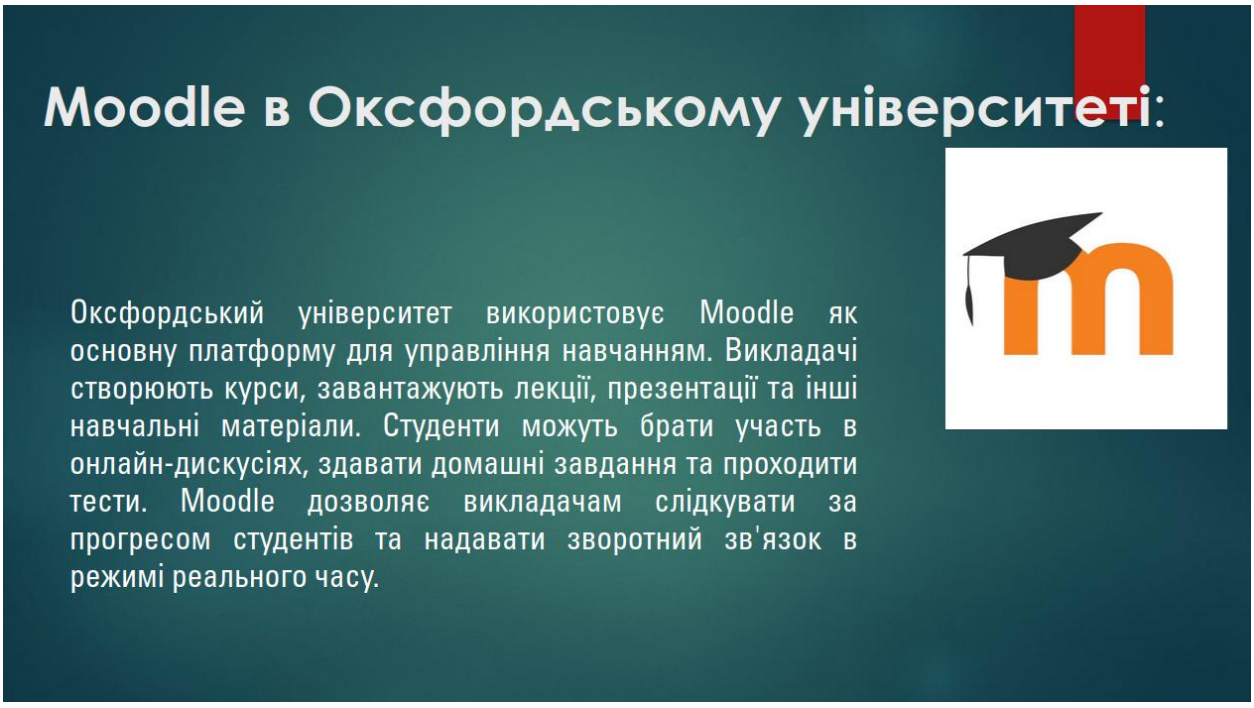

Рисунок В.3 – Слайд презентації № 3

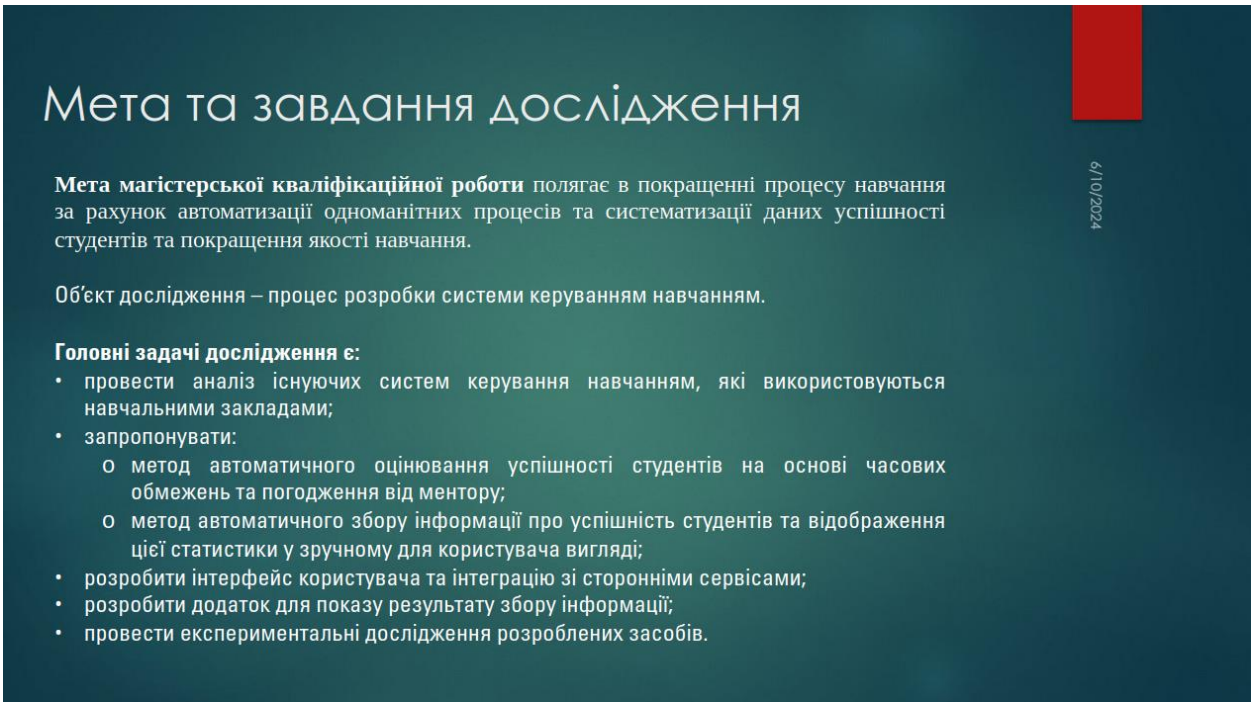

Рисунок В.4 – Слайд презентації № 4

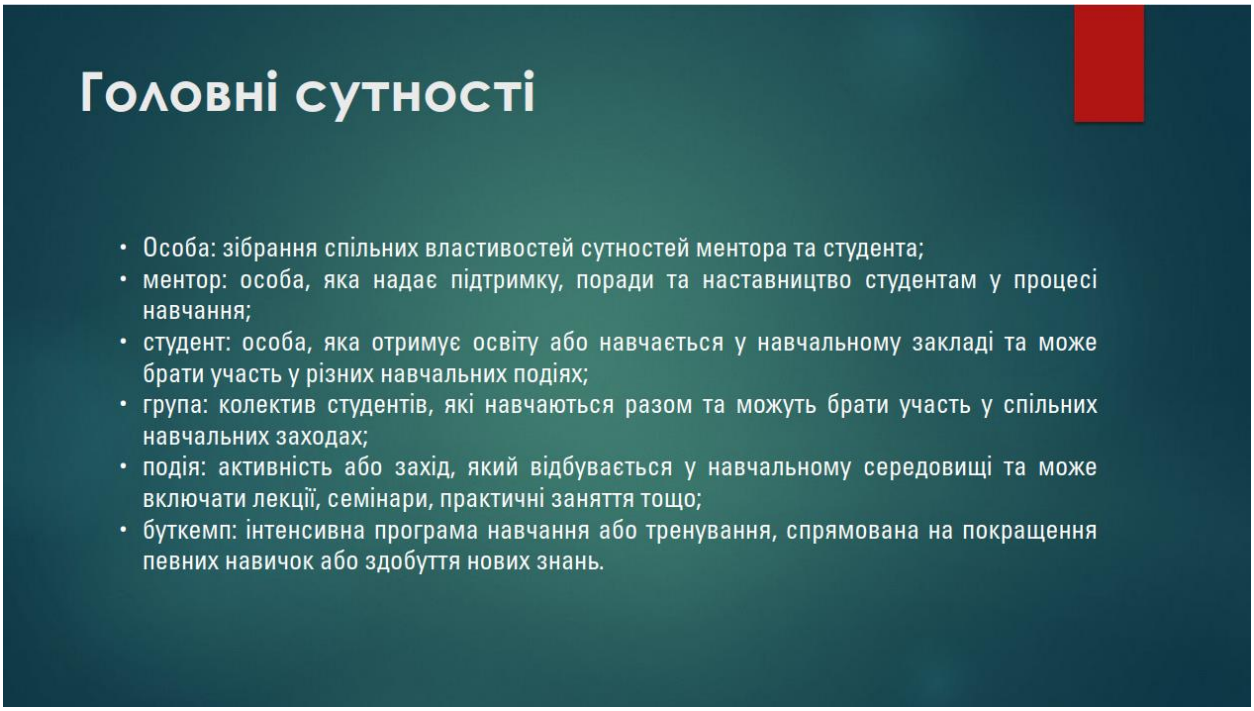

Рисунок В.5 – Слайд презентації № 5

Рисунок В.6 – Слайд презентації № 6

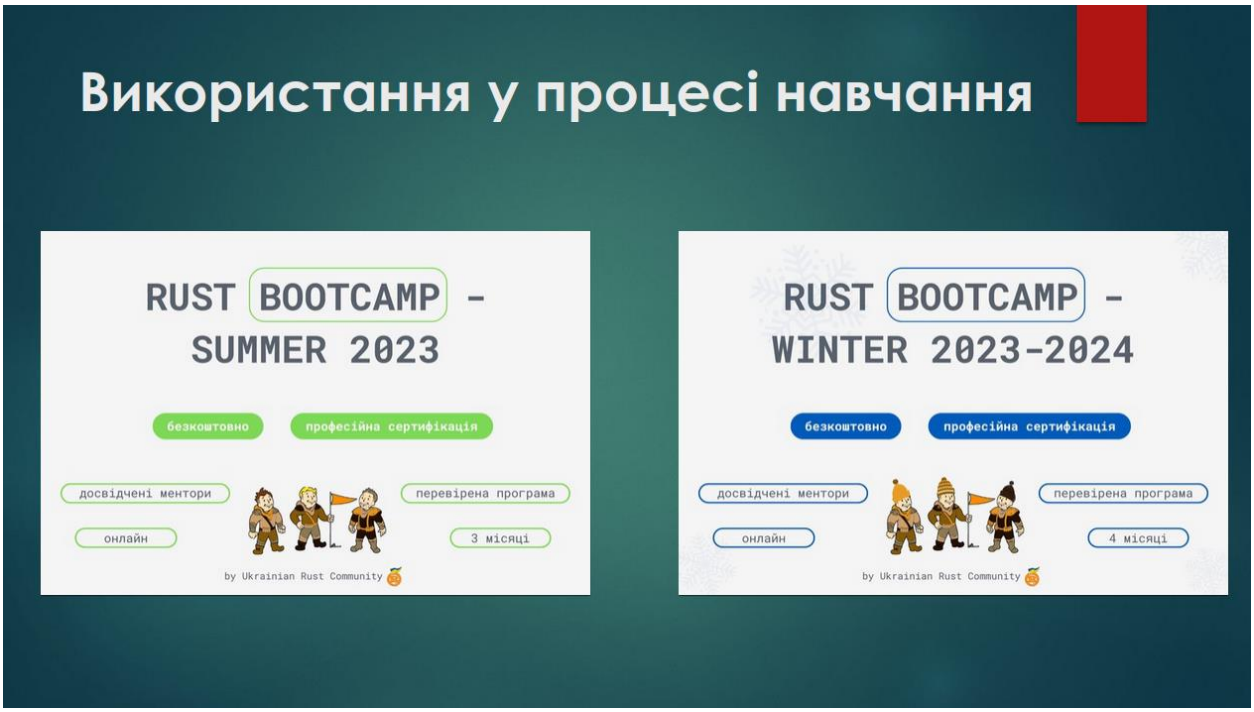

Рисунок В.7 – Слайд презентації № 7

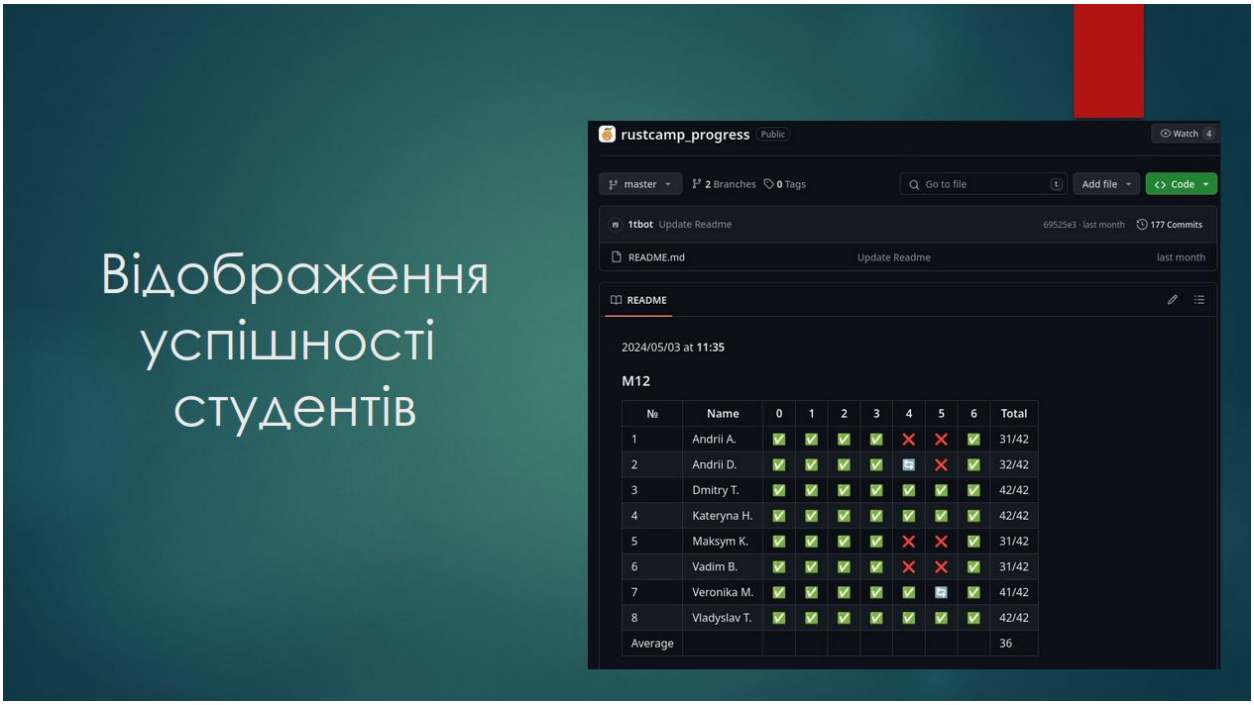

Рисунок В.8 – Слайд презентації № 8

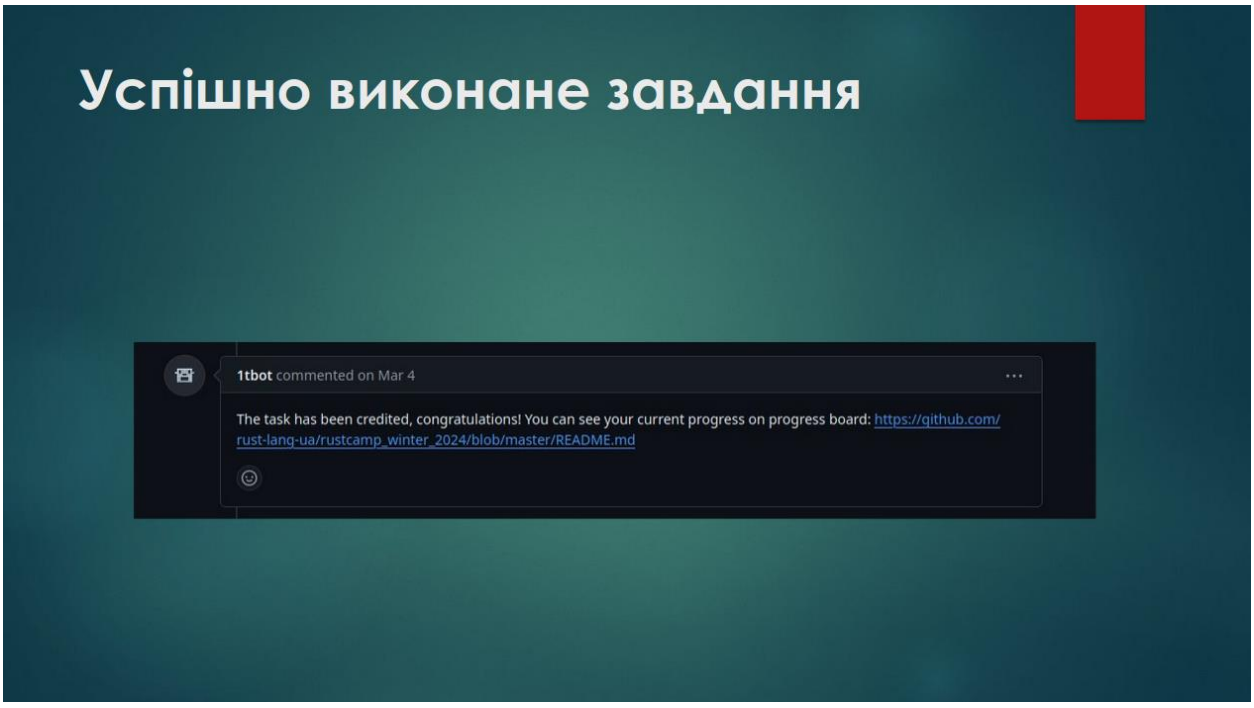

Рисунок В.9 – Слайд презентації № 9

Рисунок В.10 – Слайд презентації № 10

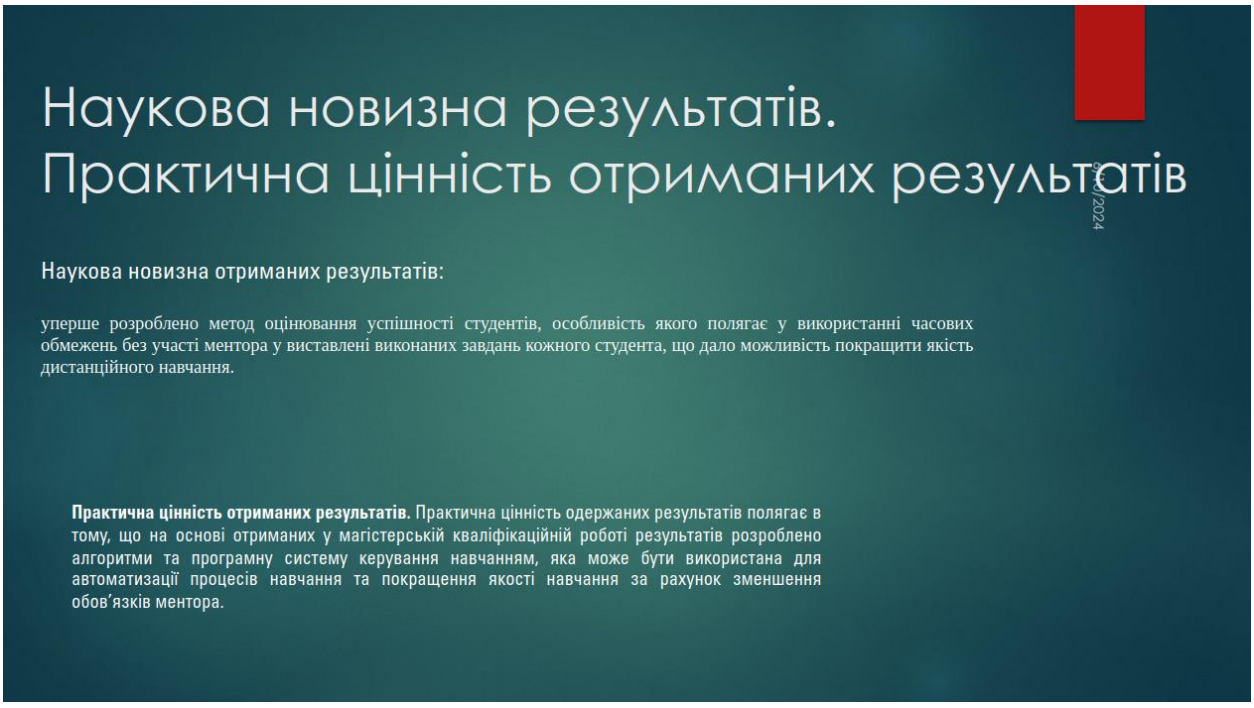

Рисунок В.11 – Слайд презентації № 11

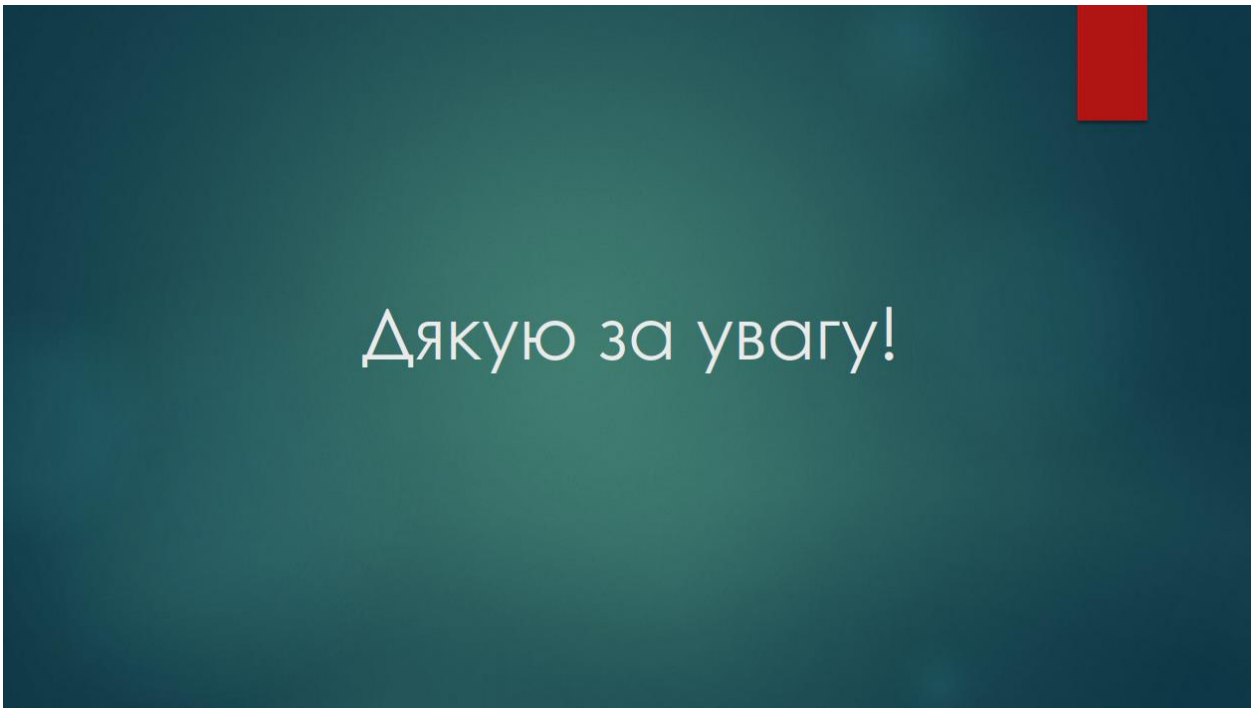

Рисунок В.12 – Слайд презентації № 12