**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

**МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ**
**Кафедра комп'ютерних наук**

# КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ СТВОРЕННЯ НАВЧАЛЬНИХ ТЕКСТОВИХ КВЕСТІВ»**

Виконав:
студент 4 курсу, групи 6.1229
спеціальності 122 комп'ютерні науки
(шифр і назва спеціальності)
освітньої програми комп'ютерні науки
(назва освітньої програми)
М.Е. Башмаков
(ініціали та прізвище)
Керівник старший викладач кафедри комп'ютерних наук, Циммерман Г.А.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент завідувач кафедри програмної інженерії ЗНУ, к.ф.-м.н., доцент Лісняк А.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя 2023

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

Факультет математичний

Кафедра комп'ютерних наук

Рівень вищої освіти бакалавр

Спеціальність 122 комп'ютерні науки
(шифр і назва)

Освітня програма комп'ютерні науки

**ЗАТВЕРДЖУЮ**
Завідувач кафедри
комп'ютерних наук, д.т.н.,
професор

_____ Чопоров С.В.
(підпис)

" ____ " _____ 2023 р.

# З А В Д А Н Н Я

## НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Башмакову Миколі Едуардовичу
(прізвище, ім'я та по-батькові)

1. Тема роботи Розробка програмного забезпечення для створення навчальних текстових квестів

керівник роботи Циммерман Г.А., старший викладач кафедри комп'ютерних наук
(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 26 » січня 2023 року № 106-с

2. Строк подання студентом роботи 06.06.2023

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

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

1. Постановка задачі.
2. Основні теоретичні відомості.
3. Реалізація програмного забезпечення

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

презентація

6. Консультанти розділів роботи

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| | | | |
| | | | |
| | | | |

7. Дата видачі завдання 26.01.2023

## К А Л Е Н Д А Р Н И Й  П Л А Н

| № | Назва етапів кваліфікаційної роботи | Строк виконання етапів роботи | Примітка |
|---|---|---|---|
| 1. | Розробка плану роботи. | 26.03.2023 | |
| | | | |
| 2. | Збір вихідних даних. | 14.04.2023 | |
| | | | |
| 3. | Обробка методичних та теоретичних | 28.04.2023 | |
| | джерел. | | |
| | | | |
| 4. | Розробка першого та другого розділу. | 20.05.2023 | |
| | | | |
| 5. | Розробка третього розділу. | 01.06.2023 | |
| | | | |
| 6. | Оформлення та нормоконтроль | 06.06.2023 | |
| | кваліфікаційної роботи бакалавра. | | |
| | | | |
| 7. | Захист кваліфікаційної роботи. | 21.06.2023 | |

Студент _____ М.Е. Башмаков
(підпис) (ініціали та прізвище)

Керівник роботи _____ Г.А. Циммерман
(підпис) (ініціали та прізвище)

**Нормоконтроль пройдено**

Нормоконтролер _____ О.Г.Спиця
(підпис) (ініціали та прізвище)

# РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка програмного забезпечення для створення навчальних текстових квестів»: 68 с., 21 рис., 0 таблиць, 11 джерел, 2 додатків.

ІНТЕРАКТИВНЕ НАВЧАННЯ, ТЕКСТОВІ КВЕСТИ, PYTHON, NEO4J, FASTAPI, ВЕБ-ДОДАТОК.

Об’єкт дослідження –створення текстових навчальних квестів.

Мета роботи: надати вчителям, студентам, а також широкому колу користувачів інструмент для створення та проходження текстових квестів в навчальних цілях.

Метод дослідження – системний підхід.

У роботі описаний процес розробки програмного забезпечення для створення навчальних текстових квестів. Було обрано Python 3 для реалізації серверної сторони, Neo4j як базу даних та ReactJS для реалізації клієнтської сторони. Вибір кожної з технологій було обґрунтовано.

Особливу увагу в роботі приділено використанню графової бази даних Neo4j, яка дозволила використати ефективні графові структури для моделювання текстових квестів.

Було представлено процес створення текстового квесту в додатку, включаючи створення запитань, варіантів дій, взаємозв'язків, отримання квестів та збереження результату проходження. Детально описано механізм взаємодії з базою даних.

В результаті роботи було створено програмне забезпечення, яке забезпечує можливість створення та проходження навчальних текстових квестів. Програмне забезпечення було успішно протестовано та готове до використання.

# SUMMARY

Master's qualifying paper «Development of the Software for Educational Text-Based Quests Creating»: 68 pages, 21 figures, 0 tables, 11 references, 2 supplements.

INTERACTIVE STUDY, TEXT QUESTS, PYTHON, NEO4J, FASTAPI, WEB-APP.

Object of the study – Creating Text-based Educational Quests.

Aim of the study: To provide teachers, students, and a wide range of users with a tool for creating and experiencing text-based quests for educational purposes.

Method of research – Systematic approach.

The work describes the process of developing software for creating educational text-based quests. Python 3 was chosen for the backend, Neo4j as the database, and ReactJS for the front end. The choice of each technology was justified.

Special attention was given to using the Neo4j graph database, which allowed for efficient graph structures for modeling text-based quests.

The process of creating a text-based quest in the application was presented, including question creation, action choices, interconnections, quest retrieval, and result saving.

The interaction mechanism with the database was described in detail. As a result of the work, software was created that enables the creation and completion of educational text-based quests. The software was successfully tested and is ready for use.

**ЗМІСТ**

# ВСТУП

Освіта – це важлива складова розвитку суспільства. У сучасному світі, особливо з появою інформаційних технологій, навчальний процес стає все більш інтерактивним і доступним. Одним з нових підходів до навчання є використання інтерактивних технологій, зокрема квестів [1]

Квести - це інтерактивні ігри, які розроблені для досягнення певної мети, зазвичай залучення інтересу до вивчення певної теми. Вони можуть бути використані як для навчання так і для розважальних цілей.

Отже, текстовий квест - це форма гри або інтерактивної історії, де учасники спілкуються з грою шляхом введення текстових команд та отримання текстових відповідей від гри. Текстові квести базуються на сценаріях, які описують вигляд та поведінку віртуального світу чи ситуації, в якій знаходяться гравці. Основною особливістю текстових квестів є підкреслення ролі текстового вводу та виводу, що робить їх інтерактивними та стимулює уяву та творчість учасників. Особливість текстового квесту полягає у тому, що весь геймплей та комунікація з грою відбуваються через текст.

Сценарій – це послідовність подій, дій та станів, що відбуваються у текстовому квесті. Він використовується як основа для створення та реалізації інтерактивної історії.

Однак, розробка квестів є часомістким і складним процесом, що вимагає від знань з різних областей, таких як графічний дизайн, розробка серверної та клієнтської частин та взаємодія з базами даних.

Метою цієї дипломної роботи є розробка програмного забезпечення для створення навчальних текстових квестів. Це програмне забезпечення легко використовувати та надавати можливість користувачам створювати власні навчальні квести без знань з програмування та графічного дизайну. Воно доступне та зручне для вчителів та учнів.

У цій роботі вирішуються такі задачі, як аналіз існуючих програм для створення квестів, проектування, розробка та тестування програмного забезпечення.

Цей проект може мати значний внесок у навчальний процес, допомагаючи вчителям та учням створювати цікаві та інтерактивні навчальні квести, які допоможуть збільшити мотивацію до навчання та підвищити рівень засвоєння матеріалу.

Важливість розробки програмного забезпечення для створення навчальних квестів підкреслюється тим, що зараз багато шкіл та вишів шукають способи використання нових технологій для покращення навчання та залучення уваги учнів. Крім того, з огляду на те, що близько 30% навчальних закладів працюють дистанційно[2], можна побачити зростання популярності дистанційного навчання та онлайн-курсів, що створює попит на інструменти для створення інтерактивного навчального контенту, яким є це програмне забезпечення.

# 1 ДОСЛІДЖЕННЯ ВИКОРИСТАННЯ ТЕКСТОВИХ КВЕСТІВ У НАВЧАННІ

У цьому розділі досліджуються праці, присвячені застосуванню інтерактивних технологій у навчанні. Проведений аналіз різноманітних способів та підходів до створення текстових квестів, їхній вплив на освітній процес, а також результат.

Крім того, будуть розглянуті переваги та недоліки використання текстових квестів у навчанні. Проведений аналіз результатів досліджень, які демонструють ефективність використання цих інструментів для підвищення мотивації студентів та поліпшення їхньої академічної успішності.

Також у розділі будуть розглянуті варіанти застосування текстових квестів у різних ситуаціях. Будуть проаналізовані приклади використання текстових квестів для підвищення здібностей до критичного мислення та розвитку креативності у студентів.

## 1.1 Визначення проблеми

В останні роки спостерігається зростаючий інтерес до навчальних ігор та інтерактивних форматів навчання. У науково-методичному збірнику «ОСВІТА УКРАЇНИ В УМОВАХ ВОЄННОГО СТАНУ» зазначено: «*Найбільш актуальними залишаються такі проблеми: організація інноваційного навчального середовища […]; формування мотивації і пізнавального інтересу учнів до навчання; упровадження інтерактивних форм навчання учнів […]; активізація навчально-пізнавальної діяльності учнів, формування індивідуальної траєкторії розвитку учня […]; використання електронних освітніх ігрових ресурсів для підвищення якості освіти»*[3]. Варто зауважити, що ці проблеми є актуальними для шкіл, а також для закладів вищої освіти. Одним із способів

рішення вищеописаних проблем є запровадження використання текстових квестів, які використовуються для навчання та розвитку критичного мислення, логіки, творчості та рішення проблем.

Проте, існуючі інструменти для створення навчальних текстових квестів мають свої обмеження та недоліки, які будуть детально розглянуті в пункті 1.5. Вони можуть бути складними у використанні для вчителів або творців курсів, а також мають обмежені можливості налаштування та індивідуалізації.

Однією з основних проблем організації сучасного супроводу освітнього процесу є відсутність зручного та потужного програмного забезпечення, яке б дозволяло швидко та ефективно створювати навчальні текстові квести з урахуванням особливостей теми, навчальних цілей та потреб учнів. Наявні рішення можуть бути недостатньо гнучкими або не забезпечувати достатньої функціональності для повноцінного створення навчальних квестів.

Таким чином, актуальною проблемою є розробка програмного забезпечення, яке надасть зручний та потужний інструмент для створення навчальних текстових квестів з можливістю налаштування, індивідуалізації та адаптації до потреб користувачів. Таке програмне забезпечення допоможе вчителям, авторам навчальних курсів та іншим зацікавленим особам швидко та ефективно створювати захоплюючі та інтерактивні навчальні текстові квести для різних груп учнів.

## 1.2 Доцільність використання текстових квестів

Текстові квести можна ефективно використовувати в різних ситуаціях, але є певні випадки, коли їх застосування може бути менш доцільним. Розглянемо деякі з них:

Ефективне використання текстових квестів:

- Навчання мови: Текстові квести можуть бути корисними для навчання мови, оскільки вони дозволяють практикувати читання, розуміння тексту, побудову речень та виразів.
- Розвиток літературної компетенції: Текстові квести сприяють розвитку літературної компетенції, адже вони включають аналіз текстів, розуміння літературних прийомів та стилів, інтерпретацію літературних персонажів та подій.
- Розвиток критичного мислення: Текстові квести можуть стимулювати критичне мислення, оскільки гравці повинні аналізувати інформацію, приймати рішення та розв'язувати проблеми.
- Розвиток творчості: Текстові квести можуть сприяти розвитку творчих навичок, оскільки гравці повинні уявляти сценарії, створювати власні рішення та розвивати креативність.

Менш доцільне використання текстових квестів:

- Навчання практичних навичок: У випадках, коли навчання вимагає безпосередньої практичної діяльності, наприклад, фізичних навичок або роботи зі спеціальним обладнанням, текстові квести можуть бути не такими ефективними.
- Випереджаючі технології: У деяких випадках, коли доступні більш передові технології, наприклад, віртуальна реальність або доповнена реальність, використання текстових квестів може бути менш привабливим.

Варто враховувати, що ефективність використання текстових квестів може залежати від контексту навчання та особливостей аудиторії. Потрібно аналізувати цільові цілі та потреби учнів, а також розглядати альтернативні методи, щоб забезпечити найефективніше навчання.

## 1.3 Аналіз методик створення текстових квестів

Існує два основні підходи до створення квестів – лінійний та нелінійний[4]. Обидва мають свої особливості.

Лінійний підхід до створення квестів – це підхід в якому порядок запитань передбачений заздалегідь та не залежить від відповіді проходжуючого квест.

Нелінійний підхід – це підхід в якому порядок запитань невизначений, тобто проходжуючий може відповідати на запитання в буд-якому порядку.

Аналіз різних методик для створення текстових квестів у навчанні надає можливість оцінити їхні переваги та недоліки з точки зору навчального процесу. Вирішення, яка методика є найкращою, залежить від конкретних цілей, контексту та вимог проекту. Нижче наведено аналіз кожної методики:

1) Особливості лінійного підходу:
   :
   - Простота структури: лінійний підхід дозволяє створювати прямолінійні та логічно зв'язані сценарії квесту.
   - Контрольоване навчання: учасники квесту проходять послідовні етапи, що дає можливість контролювати навчальний процес та забезпечує більшу увагу до деталей.
   - Обмеженість вибору: лінійний підхід може обмежувати свободу вибору та креативності учасників квесту, оскільки рішення та події заздалегідь визначені.
   - Менша варіативність: учасники можуть відчувати обмеженість, якщо сюжет квесту прогресує тільки в одному напрямку.
2) Особливості нелінійного підходу:
   - Свобода вибору: учасники мають можливість вибирати різні шляхи та рішення, що сприяє розвитку критичного мислення та самостійності.

- Більша варіативність: нелінійний підхід дозволяє створювати квести з багатьма можливими сценаріями та закінченнями, що підвищує цікавість та залученість учасників.
- Складніше керування: управління нелінійним квестом може бути складніше через багато можливих шляхів та рішень.
- Можливість заблукати: учасники можуть загубитися або неконтрольовано рухатися у різних напрямках, що може вплинути на ефективність навчання.

Зважаючи на вищеперераховані фактори, був обраний гібридний підхід.

Обгрунтування вибору гібридного підходу для створення текстових квестів базується на поєднанні переваг лінійного та нелінійного підходів, що дозволяє створити більш гнучкі та цікаві навчальні сценарії. Ось декілька аргументів, що обґрунтовують вибір гібридного підходу:

1) Структурованость та свобода: Гібридний підхід дозволяє поєднати структурованість лінійного підходу зі свободою вибору. Це створює баланс між контрольованістю та креативністю, що є важливим для навчання.
2) Забезпечення індивідуального підходу: Гібридний підхід дає можливість кожному учаснику квесту вибрати свій власний шлях та рішення, враховуючи їхні інтереси та здібності. Це дозволяє створити персоналізований навчальний досвід і підтримувати мотивацію студентів.
3) Розвиток критичного та проблемного мислення: Гібридний підхід надає можливість створювати складні сценарії, де учасники повинні вирішувати проблеми, здійснювати аналіз і приймати вагомі рішення. Це сприяє розвитку критичного та проблемного мислення у студентів.
4) Варіативність та захопленість: Гібридний підхід дозволяє створювати квести з багатьма можливими шляхами та закінченнями,

що робить їх більш захопливими для учасників. Це дозволяє стимулювати інтерес та залученість до навчання.

5) Гнучкість та адаптивність: Гібридний підхід може бути адаптований до різних навчальних цілей, предметів та груп студентів. Він надає можливість комбінувати різні елементи та підходи, що дозволяє розширити можливості навчання.

Загалом, гібридний підхід є компромісом між лінійним та нелінійним підходами, що надає більше гнучкості, індивідуалізації та захопленості в навчальному процесі. Цей підхід може бути особливо ефективним у створенні навчальних сценаріїв, які стимулюють активну участь та розвиток критичного мислення у студентів.

## 1.4 Огляд існуючих програмних засобів для створення текстових квестів

На сучасному ринку існує кілька програмних засобів, які надають можливість створювати текстові квести. Проаналізувавши їх, можна виділити такі основні рішення:

1. Інтерактивні платформи для створення текстових квестів:
   - Twine: Twine є одним з найпопулярніших інструментів для створення інтерактивних ігор і текстових квестів. Він має простий та зрозумілий інтерфейс, дозволяє створювати гілки сюжету, використовувати графічні елементи та ефекти. Однак, Twine має обмежену функціональність та орієнтований на розповідь історії, тобто використання його в навчанні також обмежена.

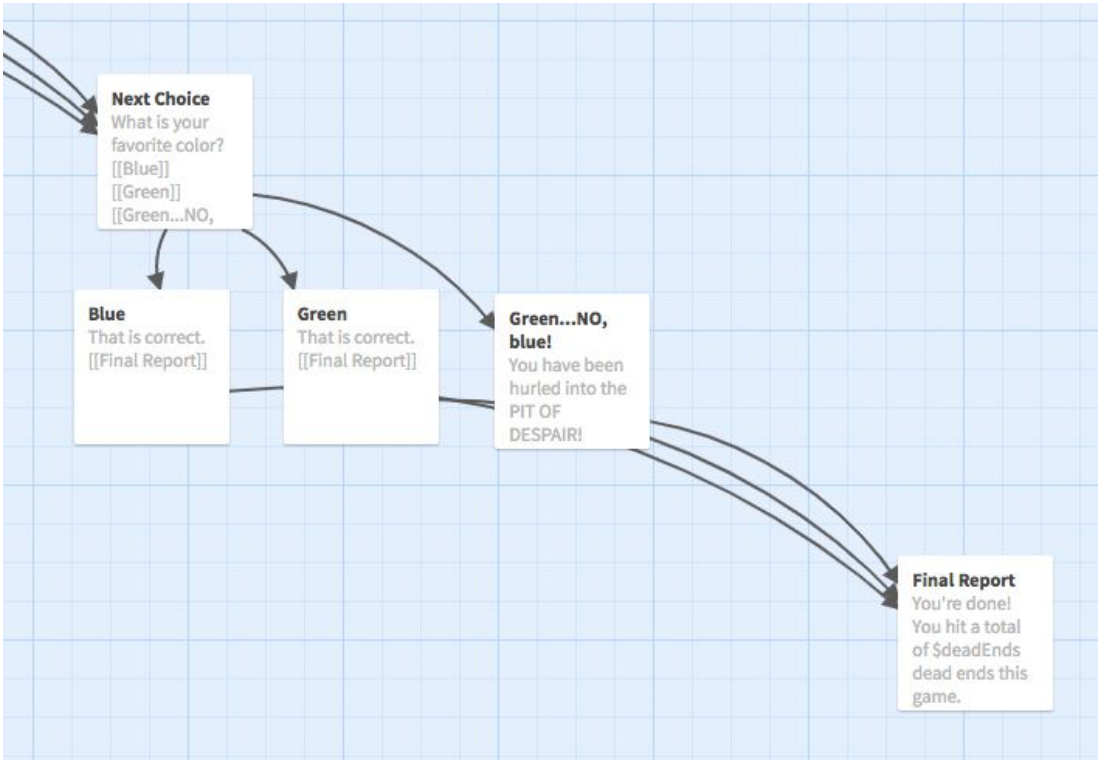


Рисунок 1.1 – Інструмент Twine

– Всеосвіта вебквести: інструмент вебквести від Всеосвіта надає можливість створювати нелінійні інтерактивні текстові квести. Цей інструмент більш підходить для молодшої школи, так як основна взаємодія відбуваються з мультимедійними елементами та квест обмежений однією сценою.

Рисунок 1.2 – Інструмент Всеосвіта вебквести

– Classcraft: Classcraft є платформою, спрямованою на навчання через ігровий процес. Вона дозволяє створювати навчальні квести з лінійними сюжетами, але також надає можливість використовувати графічні елементи, винагороди та систему управління класом. Classcraft спрямований на освітні заклади та вчителів, що дає змогу налаштовувати навчальні завдання для своїх учнів та відстежувати їх прогрес.


Рисунок 1.3 – Інструмент Classcraft

2. Мови програмування та фреймворки:
   – Python та бібліотека Pygame: Python є популярною мовою програмування, а Pygame - бібліотекою для створення ігор. За допомогою Python та Pygame можна реалізувати текстовий квест з власними правилами та функціональністю. Однак, цей підхід вимагає значних програмувальних навичок та більшої затрати часу на розробку.

Кожен з цих програмних засобів має свої переваги та обмеження. Вони можуть бути корисними для певних сценаріїв використання, але не задовольняють всі потреби користувачів, так як вимагають створення квестів

виключно вручну. Враховуючи це, з’являється потреба у створенні нового програмного забезпечення, яке би об'єднало переваги інструментів, забезпечило зручний інтерфейс, гнучкість налаштувань та широкі можливості створення навчальних текстових квестів.

### 1.5 Мета та завдання дослідження

Метою даної дипломної роботи є розробка програмного забезпечення для створення навчальних текстових квестів, яке відповідатиме вимогам та потребам користувачів. Головною метою є забезпечення зручного, ефективного та гнучкого інструменту для створення навчальних квестів, який допоможе вчителям та авторам навчальних курсів реалізувати свої навчальні цілі та створити захоплююче середовище для учнів.

Для досягнення поставленої мети передбачено наступні завдання дослідження:

- Провести огляд існуючих програмних засобів для створення текстових квестів та проаналізувати їхні можливості, обмеження та недоліки.
- Визначити основні вимоги до програмного забезпечення для створення навчальних текстових квестів з урахуванням потреб користувачів, особливостей навчального процесу та вимог до функціональності.
- Розробити алгоритм для ефективного створення текстових квестів, включаючи механізми для генерації сюжету, обробки введення користувача та навчання.
- Реалізувати програмне забезпечення для створення навчальних текстових квестів на основі розробленого алгоритму та вимог, забезпечуючи зручний інтерфейс користувача.
- Провести експериментальне тестування розробленого програмного забезпечення з використанням реальних вчителів та авторів навчальних

курсів для оцінки його ефективності, зручності використання та відповідності вимогам.

– Зробити аналіз отриманих результатів, оцінити досягнення мети та порівняти розроблене програмне забезпечення з існуючими рішеннями на ринку.

Цільовою аудиторією цього інструменту є освітні заклади, вчителі та студенти, які зацікавлені в застосуванні текстових квестів у навчальному процесі. Інструмент розробляється з урахуванням потреб та вимог освітньої галузі, спрямований на полегшення створення та використання навчальних текстових квестів.

Цільова аудиторія включає такі групи:

– Освітні заклади: Інструмент призначений для використання в школах, коледжах, університетах та інших освітніх закладах. Він надає вчителям зручні та ефективні засоби для створення та проведення навчальних квестів у різних предметних областях.
– Вчителі: Інструмент допомагає вчителям створювати цікаві та інтерактивні навчальні квести, які сприяють активному залученню студентів до навчання. Вони можуть налаштовувати сценарії квестів, завдання, питання та відслідковувати прогрес студентів.
– Студенти: Інструмент надає студентам можливість взяти участь у захопливих навчальних квестах, які сприяють їхньому активному залученню та поглибленому засвоєнню навчального матеріалу. Вони можуть розвивати критичне мислення, приймати рішення та вирішувати завдання, що сприяє їхньому навчанню та розвитку.

Цей інструмент може бути корисним для різних навчальних дисциплін, від наукових предметів до гуманітарних наук. Він може бути використаний як в початковій, так і в середній/вищій освіті, забезпечуючи зручний та ефективний спосіб навчання через текстові квести.

В результаті виконання дослідження та розробки програмного забезпечення очікується створення зручного та потужного інструменту для

створення навчальних текстових квестів, який допоможе покращити якість навчання та розвитку учнів, сприятиме зацікавленості та активній участі у навчальному процесі.

# 2 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ СТВОРЕННЯ НАВЧАЛЬНИХ ТЕКСТОВИХ КВЕСТІВ

В даному розділі дипломної роботи досліджується реалізація програмного забезпечення для створення навчальних текстових квестів. Розглядається архітектура програмного забезпечення, його основні функції, а також основні етапи розробки, починаючи з аналізу вимог і закінчуючи тестуванням і впровадженням. В цьому розділі надаються детальні описи кожного етапу розробки, а також надаються вказівки щодо використання розробленого програмного забезпечення для створення навчальних текстових квестів.

Розглянутий механізм реалізації, інструменти, бібліотеки та середовища розробки, які були використані для розробки програмного забезпечення. У процесі розробки використовувалися сучасні технології, які дозволили забезпечити високу якість продукту та забезпечити ефективність його роботи.

Проаналізовано результати роботи програмного забезпечення, а також надано рекомендації щодо подальшого розвитку та вдосконалення програмного забезпечення.

## 2.1 Постановка технічного завдання

### 2.1.1 Мета та область застосування

Метою цього проекту є розробка веб-застосунку, який дозволить користувачам створювати, редагувати та розповсюджувати навчальні текстові квести. Цей застосунок знайде застосування в освітніх установах та для навчання за допомогою гри.

### 2.1.2 Вимоги до функціональності

1) **модуль авторизації:** Забезпечити реєстрацію/авторизацію користувачів.

2) **модуль створення квесту:** Дозволяє користувачам створювати нові квести, додавати сцени, варіанти дій та взаємозв'язки між сценами.
3) **модуль редагування квесту:** Дозволяє користувачам редагувати створені квести, включаючи сцени, варіанти дій та взаємозв'язки.
4) **модуль перегляду квесту:** Дозволяє користувачам переглядати створені квести та переходити від сцени до сцени, вибираючи варіанти дій.

### 2.1.3 Визначення нефункціональних вимог

У процесі розробки програмного забезпечення для створення навчальних текстових квестів, окрім функціональних вимог, необхідно враховувати й нефункціональні вимоги. Нефункціональні вимоги визначають властивості та характеристики системи, які не пов'язані безпосередньо з її функціональністю, але мають важливе значення для успішної реалізації та ефективного використання.

Основні нефункціональні вимоги до програмного забезпечення для створення навчальних текстових квестів включають наступне:

1) **Надійність**: Система повинна працювати стабільно та надійно, не допускаючи критичних помилок або відмов. Користувачі повинні мати можливість безперебійно використовувати програмне забезпечення для створення та редагування навчальних текстових квестів.
2) **Швидкодія**: Система повинна працювати швидко та ефективно, забезпечуючи миттєву відгуки на дії користувача. Час відгуку системи на команди користувача повинен бути мінімальним, щоб забезпечити комфортне та продуктивне використання.
3) **Масштабованість**: Система повинна бути здатною працювати з різними масштабами навантаження та кількості користувачів. При збільшенні обсягу даних або кількості одночасних користувачів, система повинна

продовжувати функціонувати стабільно та швидко, не втрачаючи продуктивності.

4) **Безпека**: Система повинна забезпечувати високий рівень безпеки даних, запобігати несанкціонованому доступу та зберігати конфіденційність інформації. Для цього можуть використовуватись механізми автентифікації, авторизації та шифрування даних.
5) **Зручність використання**: Система повинна мати зрозумілий та інтуїтивно зрозумілий інтерфейс користувача, що дозволяє зручно та ефективно створювати та редагувати навчальні текстові квести. Користувачі мають мати легкий доступ до необхідних функцій та можливість швидко орієнтуватись у системі.
6) **Підтримка**: Система повинна бути підтримуваною та оновлюваною з метою усунення можливих помилок, недоліків або вразливостей. Користувачі повинні мати можливість звертатися до служби підтримки для отримання допомоги та вирішення проблем.

Враховуючи ці нефункціональні вимоги, розроблене програмне забезпечення буде надійним, швидкодійним, масштабованим, безпечним, зручним у використанні та підтримуваним, що дозволить задовольнити потреби користувачів та забезпечити ефективне створення навчальних текстових квестів.

### 2.1.4 Вимоги до використовуваних технологій

1) **Серверна сторона:** Python 3 з використанням фреймворку FastAPI для розробки API.
2) **База даних:** Neo4j для зберігання даних, пов'язаних з квестами та користувачами.
3) **Клієнтська сторона:** ReactJS для створення інтерфейсу користувача.
4) **IDE:** PyCharm для розробки коду.

### 2.1.5 Вимоги до продуктивності та надійності

Програмне забезпечення повинне бути здатне обслуговувати одночасно до 100 користувачів без значного зниження продуктивності. Додаток має забезпечувати коректне функціонування в усіх передбачуваних сценаріях використання, а також захист даних користувачів.

### 2.1.6 Вимоги до тестування

Програмне забезпечення повинне бути повністю протестоване на наявність помилок і багів перед випуском. Повинні бути розроблені та виконані тестові сценарії для всіх основних функцій додатку.

### 2.1.7 Графік виконання проекту

Проект повинен бути завершений до кінця академічного року. Детальний графік роботи буде встановлено після узгодження з науковим керівником.

### 2.1.8 Вимоги до документації

Проект повинен включати повну документацію, включаючи технічне завдання, опис архітектури та дизайну системи, інструкції з використання та обслуговування додатку, а також звіти про тестування.

## 2.2 Обгрунтування вибору технологій, мови програмування та середовища розробки

У процесі розробки програмного забезпечення для створення навчальних текстових квестів було обрано використання бази даних Neo4j, мови програмування Python 3, фреймворку FastAPI для серверної сторони, фреймворку ReactJS для клієнтської сторони та IDE PyCharm. В даному розділі ми розглянемо обґрунтування вибору кожної з цих технологій, а також порівняємо їх з можливими альтернативами.

### 2.2.1 Neo4j

Neo4j є графовою базою даних, яка ефективно представляє та створює взаємозв'язки між даними. Вона була спеціально розроблена для графів, що робить її ідеальною для обробки взаємозв'язаної інформації, що є характерною для текстових квестів. Додатково, Neo4j пропонує виразну запитову мову Cypher, що спеціально призначена для графових операцій і значно спрощує обробку взаємозв'язаних даних[5]. Її основні переваги це:

- **Створена для графів:** На відміну від реляційних баз даних, що використовують таблиці, Neo4j була створена для графів. Це означає, що вона ефективно обробляє взаємозв'язані дані, які відповідають вимогам вашого проекту з розробки навчальних текстових квестів.
- **Швидкість:** Neo4j є вкрай швидкою при запитах до складних взаємозв'язків. Вона була розроблена для того, щоб швидко обробляти запити до графів, навіть якщо їх розмір зростає.
- **Виразний запитовий мова Cypher:** Запитова мова Cypher Neo4j дуже виразна і спеціально призначена для графових операцій. Вона значно спрощує створення і модифікацію взаємозв'язаних даних.

– **Масштабованість та гнучкість:** Neo4j добре масштабується і може легко розширюватися, що важливо для більших додатків.

Альтернативи Neo4j включають інші графові бази даних, такі як OrientDB і Amazon Neptune, але Neo4j часто вибирають через його відносну простоту використання, високу продуктивність та активну спільноту розробників.

#### 2.2.1.1 Neo4j у порівнянні з іншими типами баз даних.

У контексті створення навчальних текстових квестів, бази даних Neo4j, реляційні та документні бази даних відрізняються в ряду ключових аспектів.

**Neo4j vs реляційні бази даних**

Реляційні бази даних, такі як MySQL або PostgreSQL, використовують структуровані таблиці для зберігання даних. Це часто вимагає більшого обсягу роботи при проектуванні схеми бази даних, особливо для взаємозв'язаних даних, які є характерними для текстових квестів. Запити, які включають велику кількість взаємозв'язків, можуть бути складними і непродуктивними.

Натомість, Neo4j була створена спеціально для графів, що робить її ідеальною для обробки взаємозв'язаної інформації. Вона пропонує виразну запитову мову Cypher, що спрощує створення та обробку взаємозв'язків.

**Neo4j vs документні бази даних**

Документні бази даних, такі як MongoDB, зберігають дані в форматі, схожому на JSON, де кожен документ може містити різні набори полів. Це надає гнучкість при зберіганні нерегулярних або динамічних даних. Проте, для складних взаємозв'язків між документами, потрібно використовувати додаткові операції об'єднання, що може впливати на продуктивність.

У випадку з Neo4j, вона ефективно обробляє взаємозв'язані дані з мінімумом додаткових операцій, що робить її більш відповідною для задач, де важливі взаємозв'язки, таких як створення текстов

### 2.2.2 Python 3

Python 3 був обраний за його простий, чистий синтаксис, велику екосистемою бібліотек та фреймворків.

Python є однією з найбільш популярних мов програмування у світі, завдяки своїм простоті, зручності та широкій базі сторонніх бібліотек. Він дозволяє швидко розробляти високоякісне програмне забезпечення з мінімальними затратами на розробку та підтримку.

Python є мовою високого рівня, що дозволяє програмістам концентруватися на розв'язанні завдань, а не на роботі з низькорівневими деталями. Він має читабельний та лаконічний синтаксис, який полегшує розуміння коду та підтримку[6].

Python має велику кількість сторонніх бібліотек та фреймворків, що робить його відмінним вибором для розробки програмного забезпечення різного роду. Зокрема, для розробки веб-додатків на Python широко використовуються фреймворки, такі як Django, FastAPI та Flask.

Ці фактори сприяють швидкому створенню та розгортанню додатків, що робить Python 3 особливо привабливим для нашого проекту.

Можливі альтернативи, такі як Java, JavaScript (Node.js) або C#, можуть вимагати більш складного синтаксису, меншої підтримки баз даних або меншої кількості ресурсів для навчання.

### 2.2.3 Neomodel

В рамках розробки нашого додатку, ми вирішили використати OGM (Object-Graph Mapping) бібліотеку neomodel для Python. Neomodel є потужним та гнучким інструментом для роботи з Neo4j базою даних в Python, який працює на принципах OGM [7].

OGM (Object-Graph Mapping) - це методологія програмування, яка включає в себе механізми для конвертації даних між сумісними графовими базами даних та об'єктно-орієнтованими програмами. Вона дозволяє розробникам працювати з графовою базою даних, використовуючи об'єктно-орієнтоване представлення даних. Такий підхід полегшує розробку та підтримку програмного забезпечення, особливо в тих випадках, коли використовуються графові бази даних [8].

Neomodel надає розробникам інтуїтивно зрозумілий API для взаємодії з Neo4j. Вона дозволяє визначати моделі даних в Python, які відображають структуру даних в Neo4j. Крім того, neomodel підтримує складні запити до графової бази даних та обробку результатів, що значно полегшує розробку нашого додатку.

### 2.2.4 Фреймворк FastAPI

FastAPI є сучасним, високопродуктивним веб-фреймворком, створеним для простоти використання та швидкого розробки API. Його основні переваги включають швидкість виконання, дотримання стандартів OpenAPI та JSON Schema, простоту використання та відмінну підтримку асинхронних операцій [9].

Альтернативи FastAPI, такі як Django або Flask, можуть бути або надмірно складними для нашого проекту, або недостатньо потужними для виконання вимог, пов'язаних з обробкою асинхронних запитів та автоматичної документації API.

### 2.2.5 Фреймворк ReactJS

ReactJS, розроблений Facebook, є одним з найпопулярніших JavaScript фреймворків для розробки інтерфейсів користувача. Він був обраний за його

здатність створювати динамічні, відгукові веб-інтерфейси, велику спільноту розробників та розширений набір ресурсів для навчання [10].

Vue.js та Angular можуть служити альтернативами ReactJS, але вони не мають такої ж широкої підтримки спільноти або набору ресурсів для навчання.

### 2.2.6 IDE PyCharm

PyCharm від JetBrains є потужним розробницьким середовищем, що було створене спеціально для Python. Його автозаповнення, інтегровані інструменти для налагодження та тестування, а також перевірка синтаксису сприяють ефективному написанню коду Python.

Можливі альтернативи, такі як Visual Studio Code або Jupyter Notebook, можуть не мати такої повної підтримки специфічних функцій Python або інтегрованих інструментів для налагодження та тестування, які пропонує PyCharm. [11]

## 2.3 Важливі моменти під час реалізації

Під час розробки програмного забезпечення для створення навчальних текстових квестів, треба також врахувати наступні моменти:

1) розробка графової моделі даних: Незважаючи на гнучкість Neo4j, розробка відповідної графової моделі даних може бути викликом. Оскільки модель даних має відображати взаємозв'язки між елементами текстових квестів, потрібно глибоко зрозуміти структуру квестів та відповідно підібрати структуру графа.
2) безпека додатку: під час зберігання та обробки конфіденційної інформації про користувачів, треба подбати про забезпечення конфіденційності даних користувачів, а також використання

відповідних методів аутентифікації та авторизації. Недостатній захист може призвести до серйозних проблем з безпекою користувачів та даних.

3) Тестування та налагодження: Під час розробки програмного забезпечення, можуть виникнути проблеми з виявленням та виправленням помилок. Забезпечення ефективного тестування та налагодження є важливою частиною процесу розробки.
4) Дизайн інтерфейсу користувача: Створення інтуїтивно зрозумілого інтерфейсу користувача може бути викликом, особливо в умовах обмеженого часу та ресурсів, так як це творчий неперервний процес.

Ці складнощі вимагають глибокого розуміння технологій, які використовуються, та ефективного планування та управління проектом. Однак, з правильним плануванням та уважністю до деталей, ці труднощі можна успішно подолати та розробити ефективну та надійну

## 2.4 Архітектура програмного забезпечення

Для розробки програмного забезпечення для створення текстових квестів була створена така архітектурна модель:

1) Клієнтський інтерфейс: Цей компонент відповідає за взаємодію з користувачем. Він містить графічний інтерфейс (веб-додаток) і забезпечує можливість вводу команд і отримання результатів від гри. Клієнтський інтерфейс може реалізований з використанням технологій, таких як HTML, CSS і JavaScript та фреймворку ReactJS.
2) База даних: Цей компонент використовується для зберігання та управління даними гри, такими як інформація про користувачів, стан гри, завдання та різні налаштування. Для зберігання даних використовується графова база даних Neo4j.

3) Серверна логіка: Цей компонент забезпечує обробку запитів від клієнтського інтерфейсу та взаємодію з базою даних. Він включає наступні модулі:
   – модуль авторизації та аутентифікації користувачів
   – модуль створення, отримання, зміни та видалення даних про предмети, квести та запитання
   – модуль взаємодії з квестом
   – модуль обробки запитів до бази даних та забезпечення безпеки даних, реалізований за допомогою OGM бібліотеки neomodel

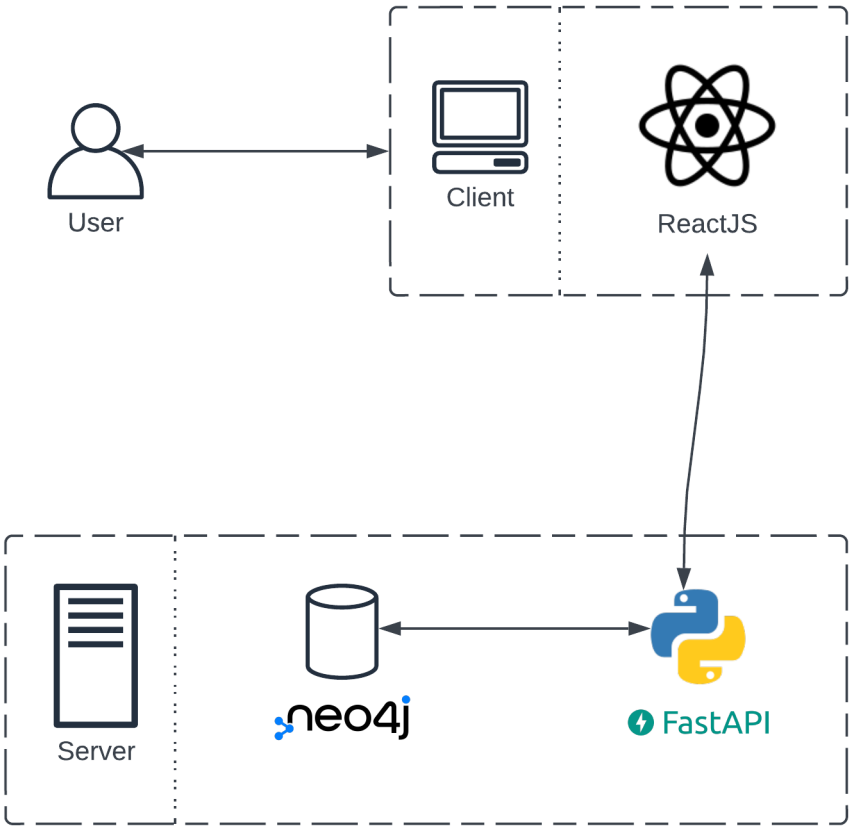


Рисунок 2.1 – Діаграма архітектури програмного забезпечення

## 2.5 Структура текстового квесту

Текстовий квест складається з наступних елементів:

- **Питання:** Кожна сцена представляє собою певну ситуацію в квесті. У неї є опис, який читає гравець, та декілька можливих варіантів дій.
- **Варіанти дій:** Це вибори, які гравець може зробити в кожній сцені. Кожен варіант дії пов'язаний з одним або кількома питаннями, до яких він призводить.
- **Взаємозв'язки:** Взаємозв'язки визначають, як вибори гравця впливають на перехід між питаннями.

Структура текстового квесту може бути представлена як граф, де сцени - це вузли, а взаємозв'язки між сценами через варіанти дій - це ребра.

## 2.6 Механізм побудови квестів

Механізм побудови квестів в нашому додатку включає наступні кроки:

- **Створення питань:** Автор квесту додає нові питання, вводячи текстовий опис ситуації для кожного питання.
- **Створення варіантів дій:** Для кожного питання автор додає один або декілька варіантів дій. Кожен варіант дії має текстовий опис, який представляє дію, яку гравець може обрати.
- **Створення взаємозв'язків:** Автор з'єднує варіанти дій з питаннями, до яких вони ведуть. Це створює взаємозв'язки між питаннями та формує граф

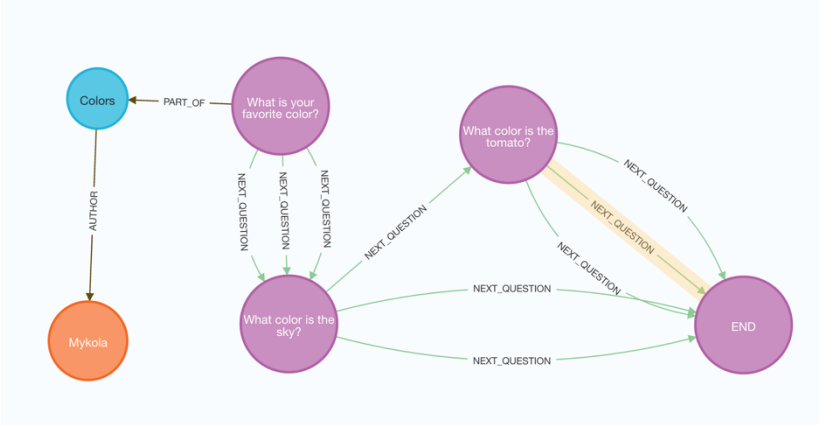


Рисунок 2.2 – Приклад структури текстового квесту

квесту.

Цей процес побудови квесту забезпечує велику гнучкість і дозволяє створювати складні та різноманітні квести.

### 2.6.1 Взаємодія з базою даних

Для взаємодії з базою даних були створені моделі вузлів та зв'язків за допомогою OGM neomodel. Процес створення моделі включає в себе визначення назв та типів властивостей конкретного об'єкта. Так, наприклад, на рисунках 2.2, 2.3 та 2.4 можна побачити модель квесту (Quest), запитання(Question) та ребра, яке з'єднує питання. Повний список моделей можна подивитись в додатку Б

```python
class Quest(StructuredNode):
    uid = UniqueIdProperty()
    name = StringProperty()
    description = StringProperty()
    image_url = StringProperty()

    subject = RelationshipTo("quests.db.models.subject.Subject", "BELONGS_TO")
    author = RelationshipTo("quests.db.models.user.User", "AUTHOR")
    questions = RelationshipFrom("quests.db.models.question.Question", "PART_OF")
```

Рисунок 2.3 – Модель Quest

```python
class Question(StructuredNode):
    ANSWER_TYPES = {"number": "number", "option": "option"}

    uid = UniqueIdProperty()
    name = StringProperty()
    question = StringProperty()
    answer = StringProperty()
    points = IntegerProperty()
    answer_type = StringProperty(ANSWER_TYPES)
    options = ArrayProperty(StringProperty())

    next_questions = RelationshipTo("Question", "NEXT_QUESTION", model=QuestionConnector)
    from_questions = RelationshipFrom("Question", "NEXT_QUESTION", model=QuestionConnector)
```

Рисунок 2.4 – Модель Question

```
class QuestionConnector(StructuredRel):
    points = IntegerProperty(default=0)
    answer = StringProperty()
```

Рисунок 2.5 – Модель QuestionConnector

Виконання запитів до бази даних  відбувається двома основними шляхами – це використовуючи створені моделі для простих запитів, таких як отримання вузла по ідентифікатору; та з використанням мови запитів Cypher для більш складних запитів. Усі запити до бази даних були додані в додатку А.

```
s = Subject.nodes.get(uid=subject_id)
```

Рисунок 2.6 – Приклад запиту з використанням моделі

```
questions, meta = neomodel.db.cypher_query(
    "MATCH (s:Subject {uid: $s_uid})<-[:BELONGS_TO]-(q:Quest {uid: $q_uid})<-[:PART_OF]-(que:Question) RETURN que",
    params={"s_uid": subject_id, "q_uid": quest_id},
    resolve_objects=True,
)
```

Рисунок 2.7 – Приклад запиту з використанням мови Cypher

Створення та збереження квесту відбувається шляхом збереження всіх необхідних об’єктів, таких як *запитання* (вузли) та *відповіді* (ребра) за допомогою OGM neomodel, як показано на рисунку 2.8.

```
q = Question(name=name, question=question, answer_type=answer_type, answer=answer, options=options).save()
for prev_question_data in prev_questions:
    try:
        prev_question = Question.nodes.get(uid=prev_question_data.id)
    except Question.DoesNotExist as e:
        raise CrudError(f"Question {prev_question_data.id} not found") from e
    q.from_questions.connect(prev_question, {"answer": prev_question_data.leading_answer, "points": 0})
if not prev_questions:
    q.quest.connect(quest)
```

Рисунок 2.8 – Процес збереження запитання квесту

### 2.6.2 Отримання квестів

Для отримання переліку квестів використовується запит до бази даних

Для отримання повної інформації про, наприклад, всі квести, створені користувачем з ім’ям «Mykola», достатньо зробити наступний запит до Neo4j:

```
MATCH path = (m:User {name: 'Mykola'})<-[:AUTHOR*]-(quest:Quest)<-[:PART_OF]-(:Question)-[*]->(:Question) RETURN path
```

Розглянемо цей запит: ми запитуємо в бази даних всі зв’язки між вузлами типу «Question», де вузол типу «User» зв’язаний ребром типу «AUTHOR» з вузлом типу «Quest», який в свою чергу з’єднаний ребром типу «PART_OF» з вузлом типу «Question», який відповідає першому запитанню квесту.

### 2.6.3 Збереження результату проходження

Після завершення квесту, результати проходження зберігаються в базі даних. Збереження відбувається шляхом з’єднання вузла користувача з вузлом питанням, на яке була дана відповідь. Після завершення квесту також з’єднується вузол користувача з вузлом квесту для індикації його проходження. Таким чином, зберігається інформація про вибрані користувачем варіанти дій, а також загальна кількість балів по проходженні квесту. Ці дані можуть бути використані для аналізу зворотного зв'язку користувачів, що допомагає авторам квестів поліпшувати їхні розробки та створювати більш цікавий контент.

```
def submit_answer(user_id: str, subject_id: str, quest_id: str, question_id: str, *, answer):
...

    next_question_rel = neomodel.db.cypher_query(
        "MATCH (q:Question {uid: $q_uid})-[r:NEXT_QUESTION {answer: $answer}]->() RETURN r",
        params={"q_uid": question_id, "answer": answer},
        resolve_objects=True,
    )
    if len(next_question_rel[0]) > 0:
        points = next_question_rel[0][0][0].points
    elif question.answer == answer and question.points:
        points = question.points
    else:
        points = None
```

Рисунок 2.9 – Збереження відповіді користувача

Крім того, користувачі мають змогу переглядати свої попередні результати проходження квестів, що дозволяє їм відслідковувати свій прогрес або переглянути свої вибори в попередніх квестах.

## 2.7 Проектування інтерфейсу користувача

Проектування ефективного та зручного інтерфейсу користувача є одним з ключових етапів розробки програмного забезпечення для створення навчальних текстових квестів. Візуальна привабливість та зрозумілість інтерфейсу мають вирішальний вплив на задоволення користувачів та їхню продуктивність під час роботи з програмою.

Основні принципи, якими керувались під час проектування інтерфейсу, включають:

1) **Простота та зрозумілість**: Інтерфейс повинен бути логічно побудованим, зрозумілим та надати користувачам можливість легко орієнтуватись у системі. Це досягається шляхом розміщення елементів інтерфейсу в логічних групах, використання зрозумілих символів та підписів, а також чіткого подання інформації.
2) **Консистентність**: Елементи інтерфейсу повинні мати однаковий стиль та вигляд, забезпечуючи єдиний дизайн. Це створює зручність для користувача та допомагає йому швидко орієнтуватись в програмі. Дотримання стандартів та рекомендацій щодо дизайну інтерфейсу сприяє зручності використання програмного забезпечення.
3) **Візуальна привабливість**: Інтерфейс повинен мати привабливий та естетичний дизайн, що спонукає користувачів до використання програми. Відповідно до цього розроблено дизайн, який включає гармонійні кольорові схеми, зручне розміщення елементів та естетичні деталі.

На скріншотах нижче можна побачити декілька спроектованих сторінок інтерфейсу користувача для програмного забезпечення для створення навчальних текстових квестів:

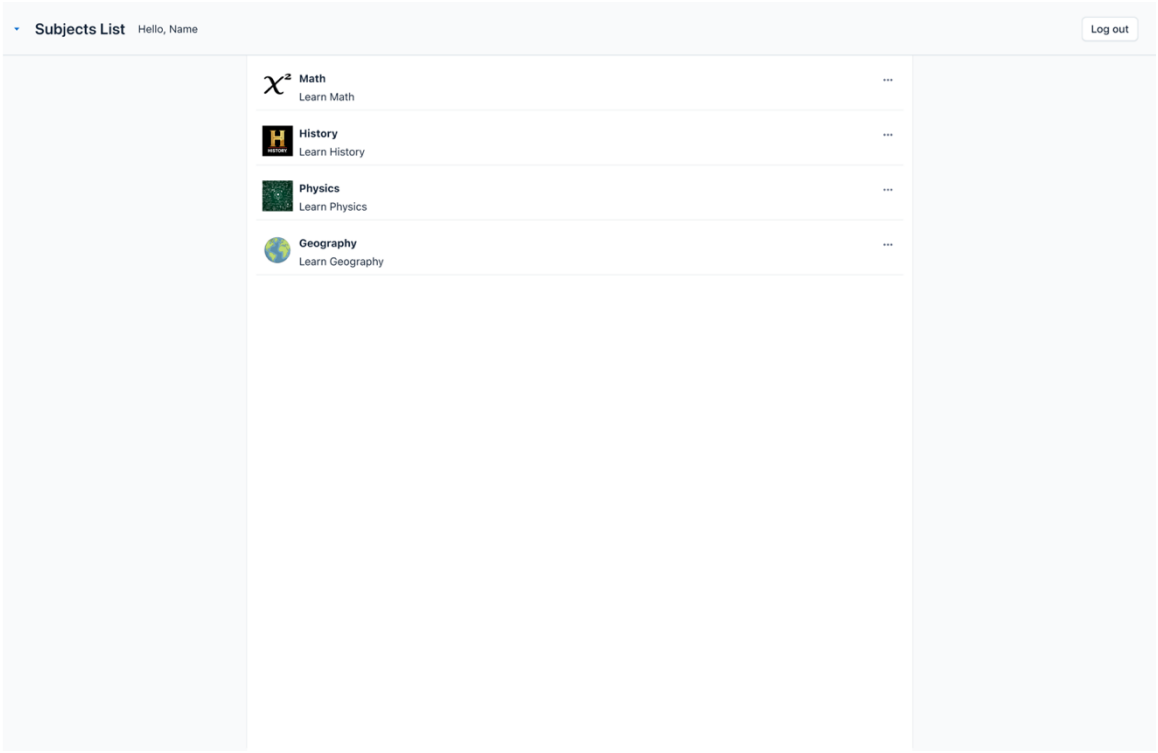


Рисунок 2.10 – Дизайн сторінки списку предметів

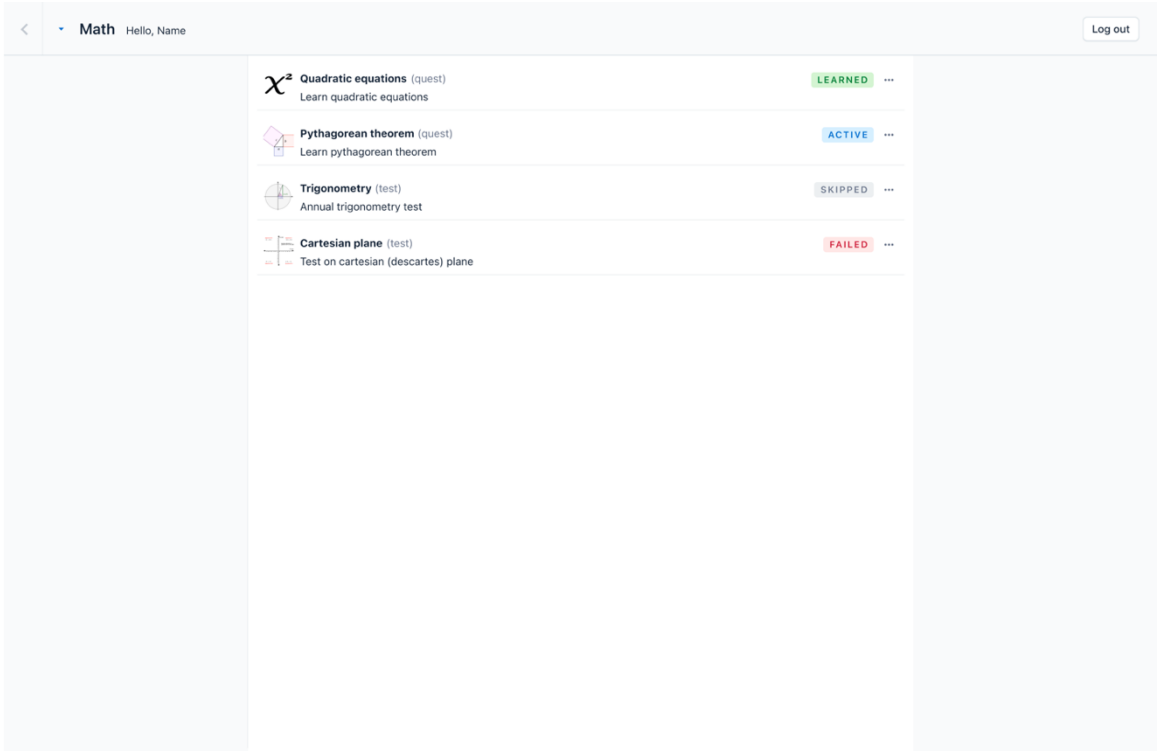


Рисунок 2.11 – Дизайн сторінки списку квестів

## 2.8 Реалізація функціональних модулів

У процесі реалізації програмного забезпечення для створення навчальних текстових квестів, були розроблені різні функціональні модулі для виконання конкретних завдань. Деякі з деталей реалізації деяких модулів описані нижче:

1) **Механізм авторизації**:
   - Реалізація функцій аутентифікації та авторизації.
2) **Механізм збереження та читання квесту в базі даних Neo4j**:
   - Створення моделі даних для збереження квестів у базі даних Neo4j, що включає визначення вузлів (наприклад, квести, категорії, питання) та зв'язків між ними.
   - Розробка функцій для збереження нових квестів, оновлення вже існуючих квестів та отримання інформації про квести з бази даних Neo4j.
   - Виконання запитів до бази даних Neo4j, використовуючи мову запитів Cypher, для отримання даних про квести з урахуванням їхніх залежностей та структури.
3) **Реалізація візуалізації квестів на стороні ReactJS**:
   - Розробка компонентів ReactJS для відображення інтерфейсу користувача, включаючи віджети, сторінки, кнопки та інші елементи, необхідні для візуалізації квестів.
   - Підключення до API програмного забезпечення для отримання даних про квести та їхнє відображення у відповідних компонентах ReactJS.
   - Реалізація логіки навігації між різними сторінками квестів та взаємодії користувача з квестами, наприклад, відповіді на питання та збереження прогресу.

Використання бібліотеки ReactJS дозволяє забезпечити надійність, ефективність та зручність роботи з програмним забезпеченням для створення навчальних текстових квестів.

### 2.8 Тестування та налагодження

В процесі розробки програмного забезпечення для створення навчальних текстових квестів, було проведено ручне тестування залученням друзів та родичів. Цей підхід до тестування дозволяє отримати реальні відгуки та оцінку від потенційних користувачів програми.

Підхід до ручного тестування включав наступні етапи:

1) **Планування тестування**: Був розроблений план тестування, в якому були визначені цілі, функціональність та сценарії тестування. Враховувалися основні функції програмного забезпечення, а також потенційні ситуації, які можуть виникнути під час використання програми.
2) **Підготовка тестового середовища**: Було підготовлено тестове середовище, в якому друзі та родичі могли проводити тестування програми. Це включало встановлення програмного забезпечення, налаштування доступу та надання необхідних вказівок щодо використання.
3) **Виконання тестових сценаріїв**: Тестувальники (друзі та родичі) виконували зазначені тестові сценарії, використовуючи програму для створення навчальних текстових квестів. Вони перевіряли різні функціональність, взаємодію з інтерфейсом, точність результатів та загальний досвід використання програми.
4) **Збір та аналіз результатів**: Було проведено збір та аналіз результатів тестування. Друзі та родичі надавали зворотній зв'язок, вказуючи на

виявлені помилки, проблеми у взаємодії, а також надавали загальні враження та пропозиції щодо поліпшення програмного забезпечення.

Результати ручного тестування були наступними:

- Виявлено і виправлено деякі помилки коду, що допомогло покращити стабільність та функціональність програмного забезпечення.
- Отримано важливий зворотний зв'язок щодо зручності використання програми, інтуїтивності інтерфейсу та загального враження від користування.
- Запропоновано деякі ідеї та побажання щодо додаткових функцій та поліпшень, які можуть бути впроваджені у майбутніх версіях програмного забезпечення.

Ручне тестування дало цінну інформацію щодо функціональності та користувацького досвіду програмного забезпечення. Отримані результати були використані для подальшого вдосконалення та налагодження програми перед її випуском.

## 2.9 Огляд кінцевого результату

У цьому пункті розглядається досягнутий результат з точки зору користувача. Отже, процес взаємодії починається зі створення користувача в системі. Для цього треба надати ім’я користувача та електронну адресу

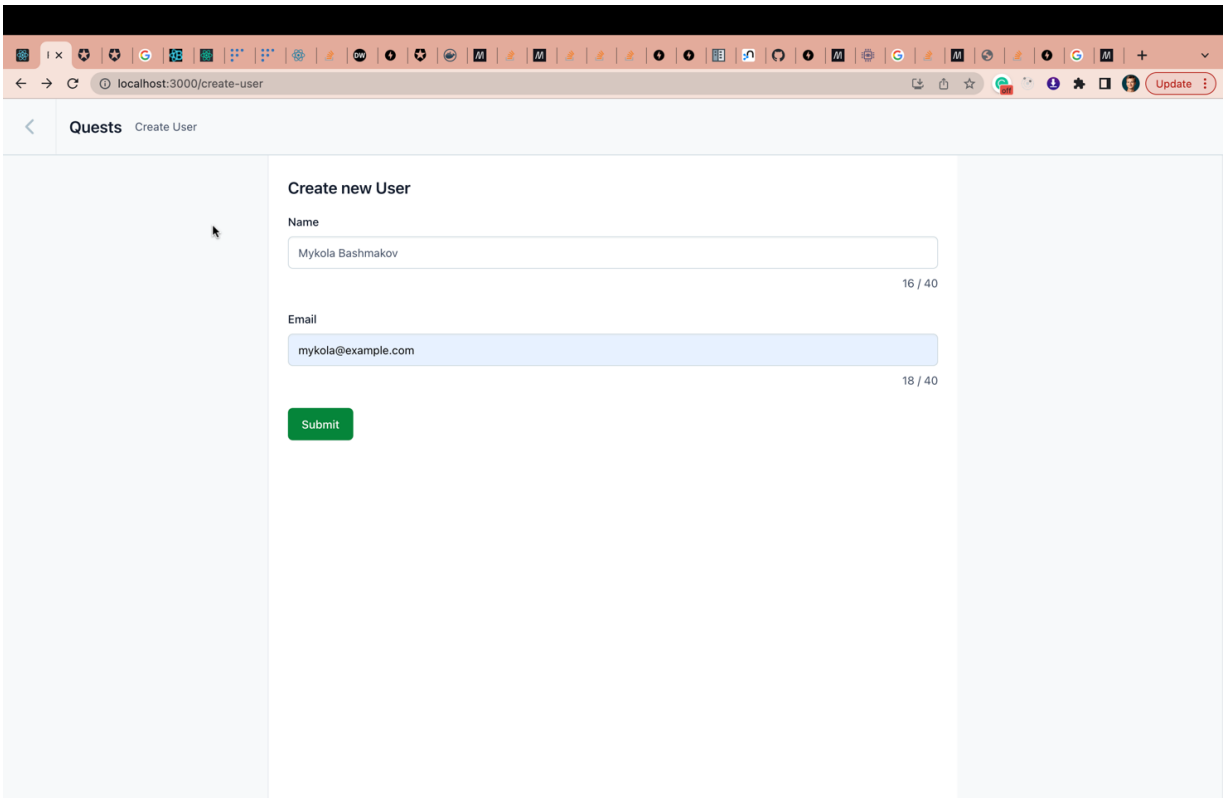


Рисунок 2.12 – Форма створення користувача

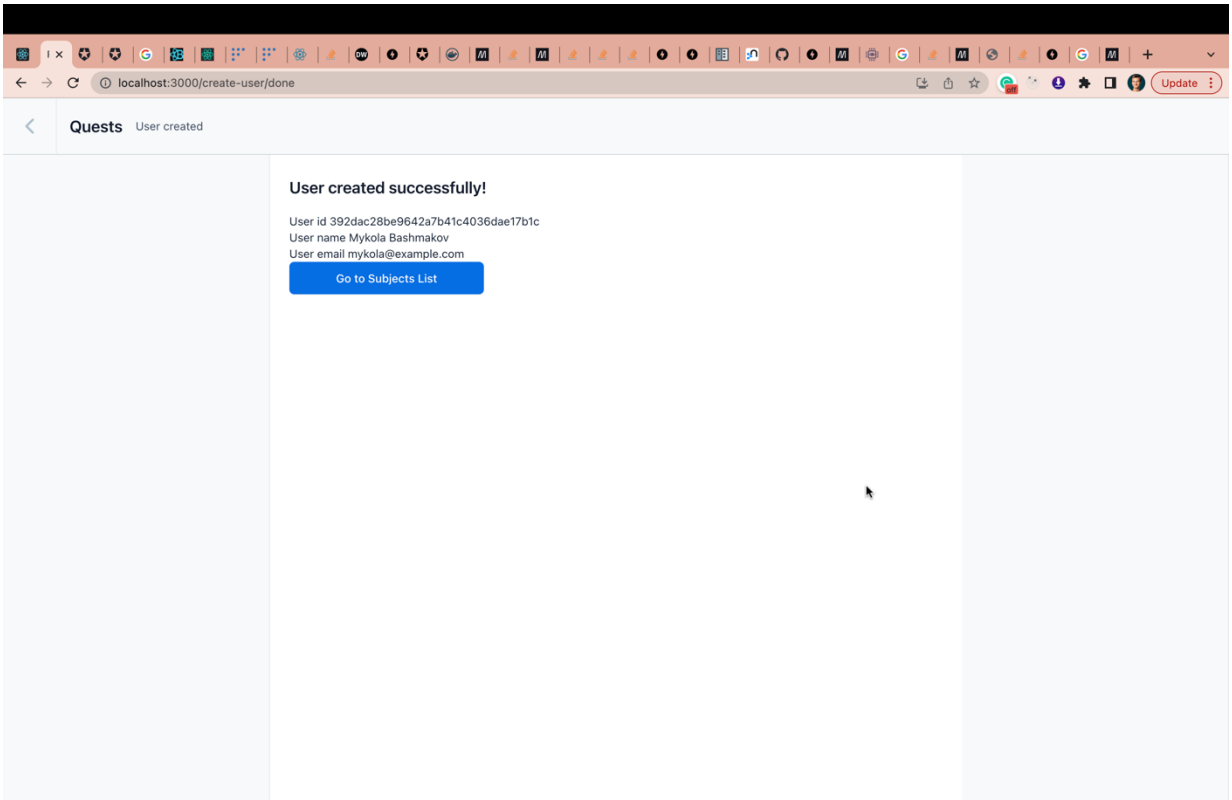


Рисунок 2.13 – Відповідь від сервера на створення користувача

Після відправки форми, користувач може побачити дані свого акаунту

Наступним кроком буде створення предмету та квесту. В користувача запитується назва предмету, або квесту та його короткий опис.

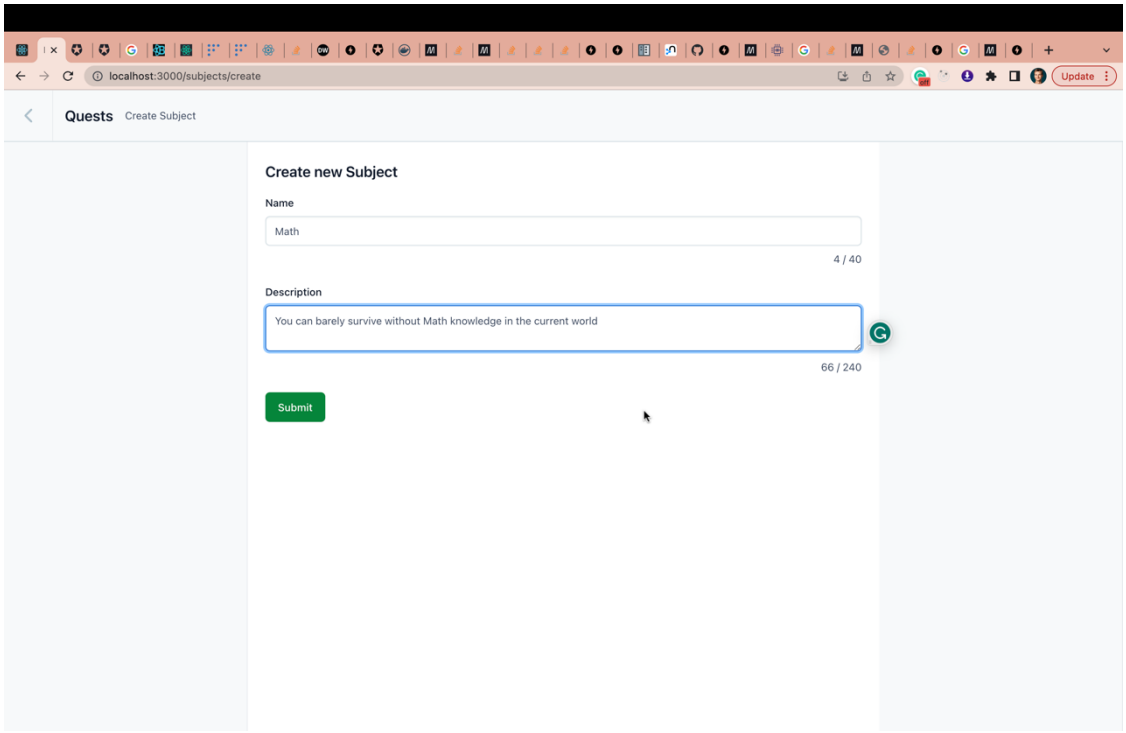


Рисунок 2.14 – Форма створення предмету

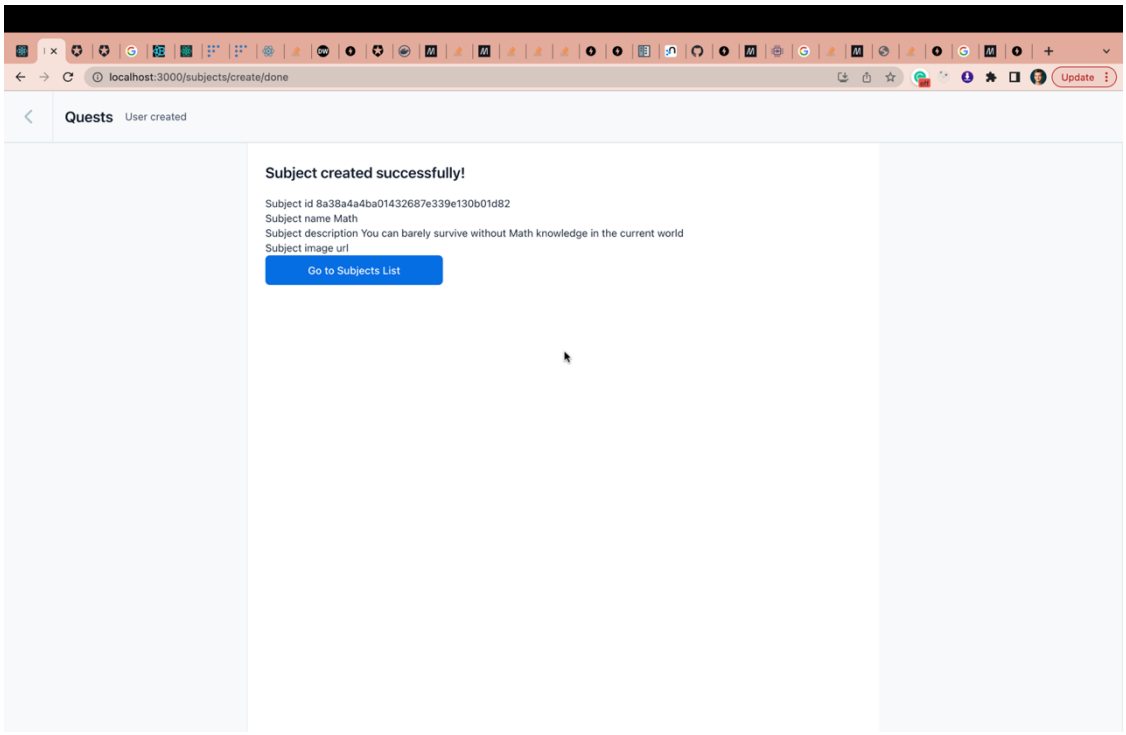


Рисунок 2.15 – Відповідь від сервера про успішне створення предмету

Після цього можна створювати питання до квесту, вказуючи ідентифікатори предмету, квесту, та, опціонально, попереднього питання. Також треба вказати очікуваний тип відповіді, надати правильну відповідь, якщо така є та кількість балів, які можна отримати за неї.

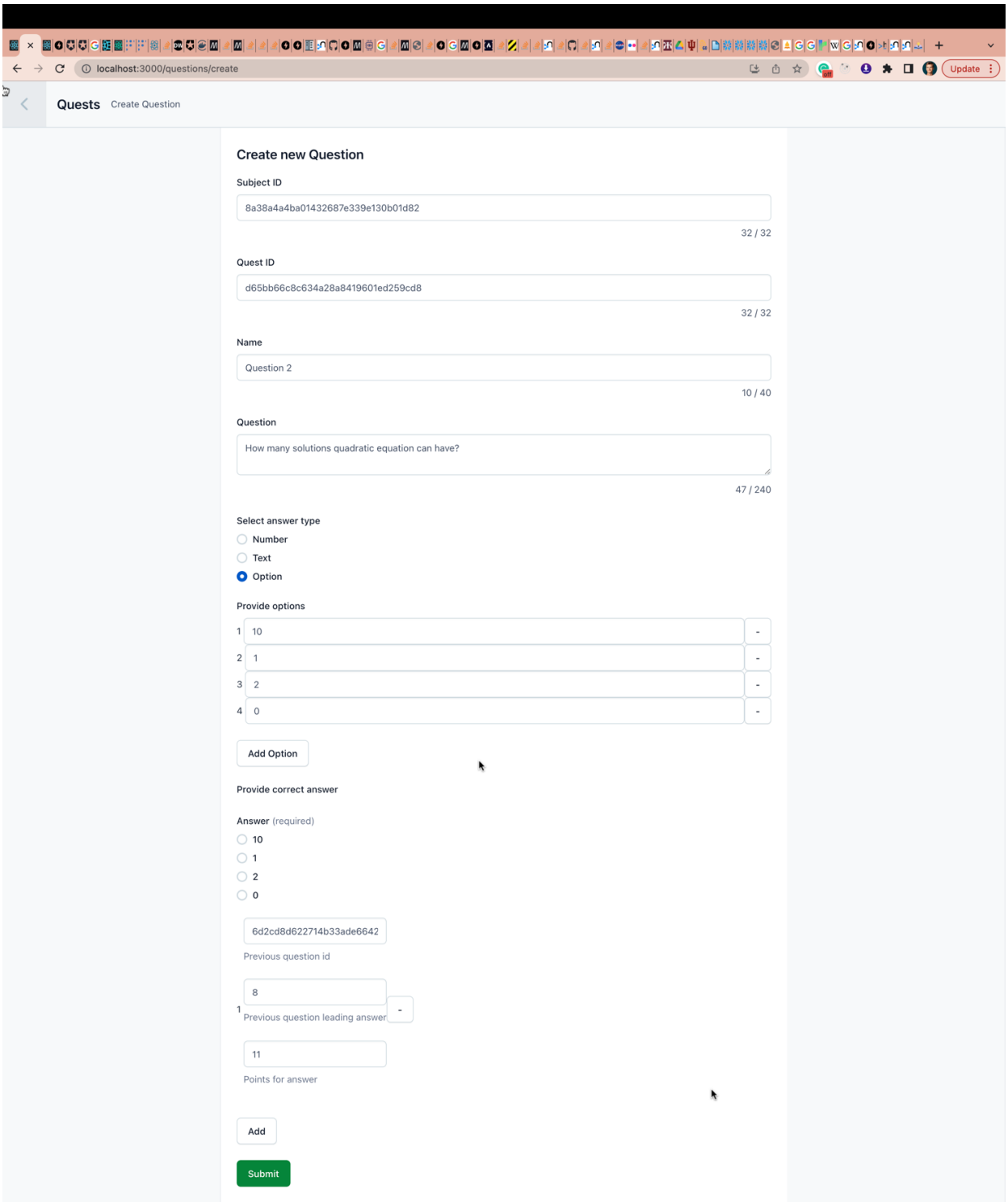


Рисунок 2.16 – Форма створення питання до квесту

Після створення квесту, користувач може переглянути список квестів, обрати бажаний до проходження та почати його проходити

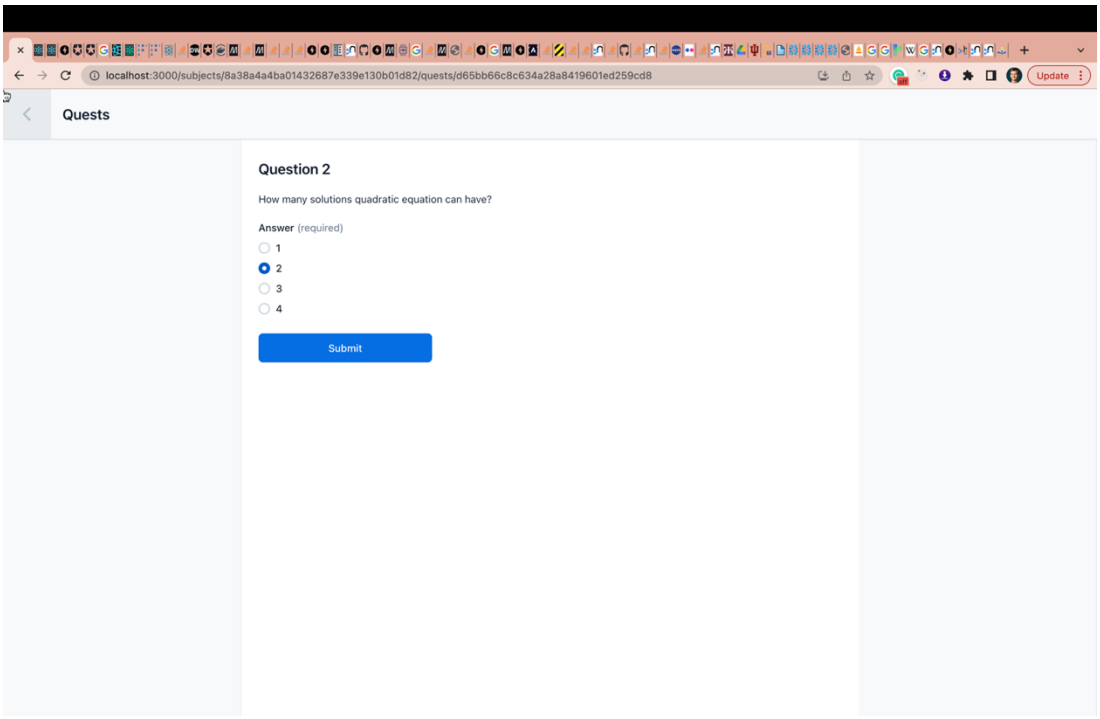


Рисунок 2.17 – Сторінка відповіді на запитання квесту

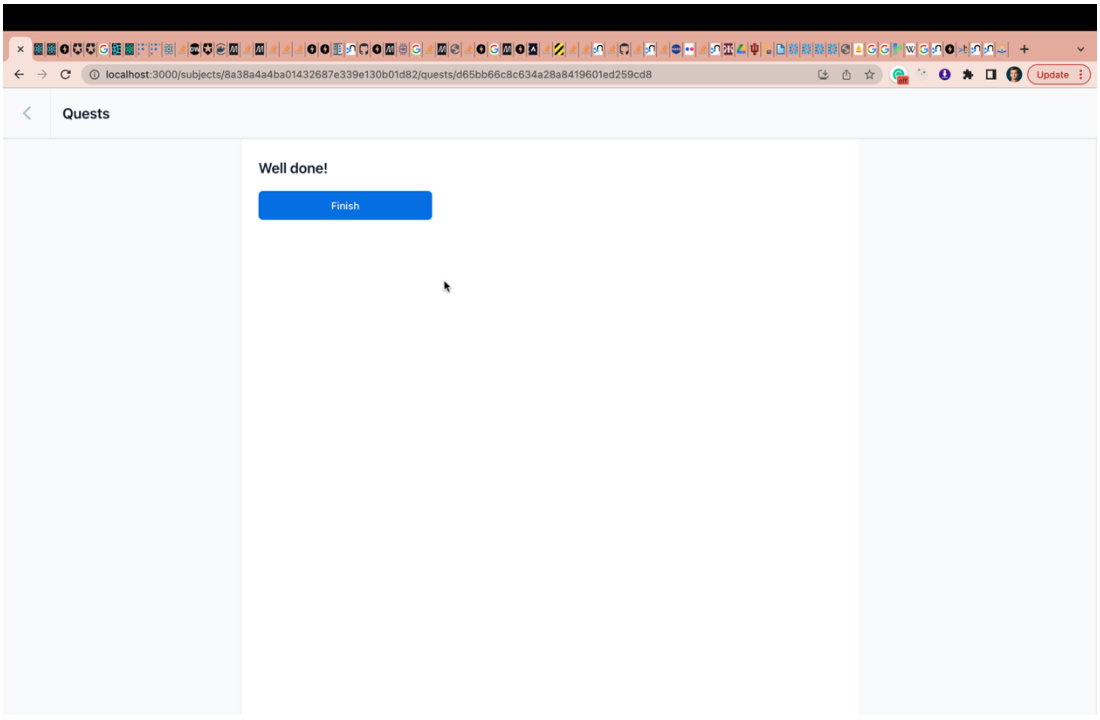


Рисунок 2.18 – Повідомлення про завершення квесту

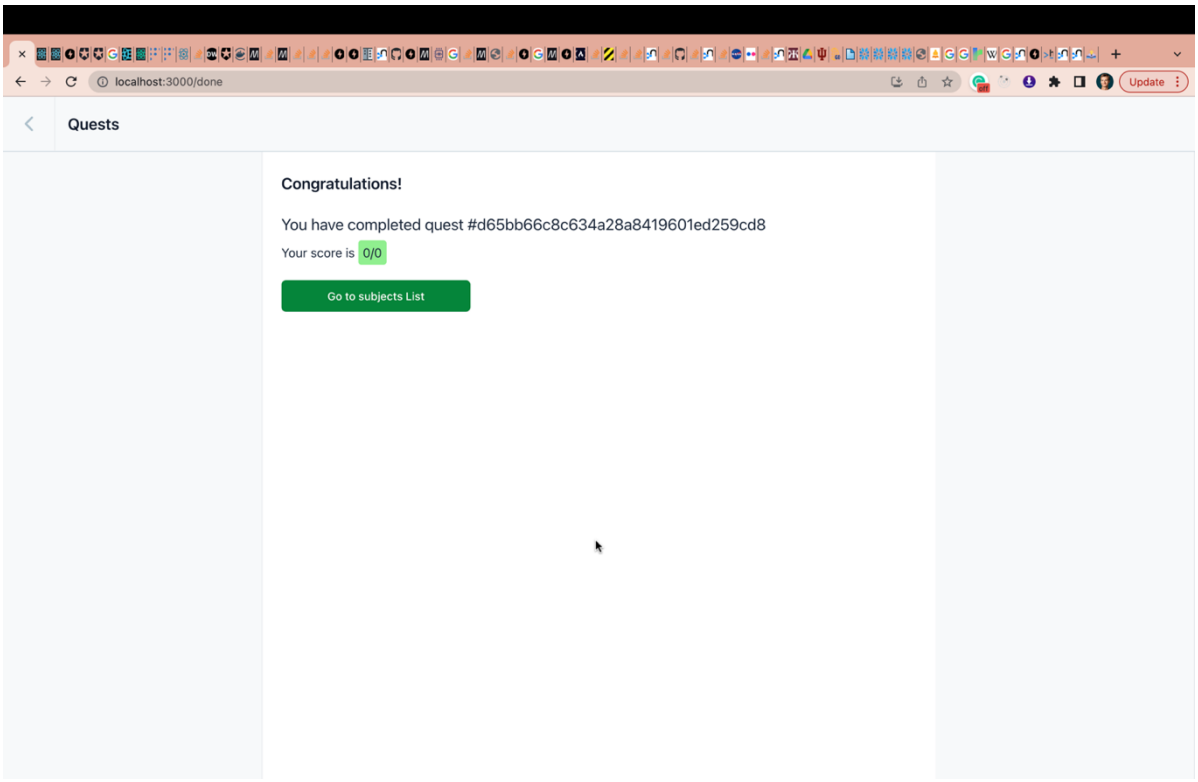


Рисунок 2.18 – Відповідь від сервера про закінчення квесту та його результат

Саме таким чином відбувається проходження квесту. Можна додавати будь-яку кількість запитань та варіантів відповідей. Тоді й проходження квесту буде довшим, ніж на наведеному прикладі.

## 2.10 Висновки

У цьому розділі було описано процес реалізації програмного забезпечення для створення навчальних текстових квестів. Були розглянуті деталі реалізації механізму збереження та читання квесту в базі даних Neo4j та візуалізації квестів на стороні ReactJS.

Щодо механізму збереження та читання квесту в базі даних Neo4j, було успішно реалізовано функціонал, що дозволяє зберігати квести у графовій базі даних. Використання бібліотеки Neo4j дозволяє ефективно зберігати та отримувати дані про квести, забезпечуючи швидкий доступ до них. Цей механізм

дозволяє легко маніпулювати квестами та здійснювати різні операції з ними, такі як створення, редагування та видалення.

Реалізація візуалізації квестів на стороні ReactJS була успішно здійснена за допомогою відповідних бібліотек та компонентів. Використання ReactJS дозволяє побудувати динамічний та інтерактивний інтерфейс користувача, який забезпечує зручне та зрозуміле взаємодію з квестами. Користувачі можуть переглядати та взаємодіяти з квестами, відповідати на питання та продовжувати квест.

Загальною висновок, реалізація програмного забезпечення для створення навчальних текстових квестів вдалася успішно. Використання бази даних Neo4j для зберігання квестів та ReactJS для візуалізації створює потужний інструмент для створення та використання навчальних квестів. Реалізоване програмне забезпечення дозволяє зручно та ефективно створювати, редагувати та використовувати квести, покращуючи навчальний процес та залучаючи студентів до активного навчання.

# 3 ПІДСУМКИ

У цій дипломній роботі було проведено дослідження та розроблено програмне забезпечення для створення навчальних текстових квестів. Результати дослідження показали, що використання текстових квестів у навчанні має значний потенціал для поліпшення мотивації студентів та підвищення їхньої академічної успішності.

Під час реалізації програмного забезпечення були використані сучасні технології та інструменти, такі як база даних Neo4j для зберігання квестів та ReactJS для візуалізації. Ці технології дозволили забезпечити зручну та ефективну роботу з програмним забезпеченням, а також забезпечити безпеку та швидкість обробки даних.

В процесі дослідження було розглянуто різні аспекти використання текстових квестів у навчанні. Були проаналізовані різні методи та підходи до створення квестів, їх вплив на навчальний процес та результати досліджень, що показали ефективність використання цих інструментів. Також були розглянуті переваги та недоліки використання текстових квестів у навчанні та їхній вплив на здібності студентів.

Загалом, розроблене програмне забезпечення відкриває нові можливості для створення та використання навчальних текстових квестів. Воно дозволяє використовувати інтерактивний та захоплюючий формат квестів для підвищення мотивації студентів та активного навчання. Також воно допомагає вчителям та навчальним закладам ефективно організовувати навчальний процес та забезпечувати індивідуальний підхід до кожного студента.

В подальшому розроблене програмне забезпечення може бути розширене та вдосконалене. Можливі напрями подальшого розвитку включають розширення функціоналу, вдосконалення інтерфейсу користувача, покращення системи збереження та обробки даних, а також впровадження додаткових функцій та інструментів для підвищення навчального ефекту.

У цілому, дослідження та розробка програмного забезпечення для створення навчальних текстових квестів є актуальною та перспективною галуззю, яка може принести значний внесок у покращення якості навчання та розвитку студентів. Використання інтерактивних та захоплюючих форматів навчання сприяє активній партіципації студентів у процесі навчання та їхньому успіху.

### 3.1 Висновки щодо досягнутих результатів

Під час розробки та дослідження програмного забезпечення для створення навчальних текстових квестів було досягнуто кілька важливих результатів. Основні висновки з цих результатів такі:

1. Було успішно реалізовано програмне забезпечення, яке дозволяє створювати, редагувати та використовувати навчальні текстові квести. Воно надає користувачам зручний інтерфейс та функціонал для створення захоплюючих та інтерактивних квестів.
2. Використання текстових квестів у навчанні показало значні переваги. Вони сприяють активному навчанню, розвитку критичного мислення та підвищенню мотивації студентів. Результати досліджень свідчать про позитивний вплив використання текстових квестів на академічну успішність студентів.
3. Застосування бази даних Neo4j для збереження квестів дозволяє ефективно маніпулювати даними та здійснювати різні операції з квестами. Швидкий доступ до даних та гнучкість бази даних Neo4j сприяють зручній роботі з квестами та забезпечують швидку відповідь системи.
4. Використання бібліотеки ReactJS для візуалізації квестів забезпечує інтерактивний та привабливий інтерфейс користувача. ReactJS дозволяє побудувати динамічні та зручні компоненти, що полегшують взаємодію користувача з квестами.

Загалом, розроблене програмне забезпечення та проведені дослідження підтверджують важливість та перспективи використання текстових квестів у навчанні. Вони можуть вплинути на якість та ефективність навчання, сприяючи активній партіципації студентів та їхньому успіху. Результати роботи вказують на потенціал використання програмного забезпечення для створення текстових квестів у різних навчальних дисциплінах та контекстах.

### 3.2 Внесок у сферу дослідження

Результати цієї дипломної роботи мають важливий внесок у сферу дослідження використання текстових квестів у навчанні. Нижче наведено основні внески, які ця робота принесла:

1. Теоретичний внесок: Дослідження, проведені в рамках цієї роботи, дозволили розширити наші знання про використання текстових квестів як інструменту навчання. Були проаналізовані різні методи та підходи до створення квестів, їх вплив на навчальний процес та ефективність використання в навчанні. Ці дослідження допомагають зрозуміти, як краще використовувати текстові квести для покращення навчального процесу та стимулювання активного навчання.
2. Практичний внесок: Розроблене програмне забезпечення є практичним інструментом для створення та використання навчальних текстових квестів. Воно надає можливість вчителям та тренерам створювати цікаві та захоплюючі квести для своїх студентів. Крім того, програмне забезпечення надає користувачам зручний інтерфейс та функціонал для легкого створення, редагування та використання квестів.
3. Педагогічний внесок: Дослідження, проведені в рамках цієї роботи, дозволяють зрозуміти, як використовувати текстові квести для покращення навчального процесу та підвищення мотивації студентів. Результати досліджень показують, що використання квестів може сприяти

активному навчанню, розвитку критичного мислення та покращенню академічної успішності студентів. Ці знання можуть бути використані в практичній роботі вчителів та навчальних закладів для покращення навчання та досягнення кращих результатів учнів.

4. Технологічний внесок: Реалізоване програмне забезпечення використовує сучасні технології та інструменти, такі як база даних Neo4j для збереження квестів та ReactJS для візуалізації. Це дозволяє використовувати новітні технології у навчанні та створювати ефективні та інтерактивні навчальні інструменти.

Отже, ця дипломна робота внесла вагомий внесок у сферу дослідження використання текстових квестів у навчанні. Вона розширила наші знання, надала практичний інструмент для створення квестів та проливає світло на важливість та перспективи використання цих інструментів у навчальному процесі.

### 3.3 Рекомендації щодо подальших досліджень

Ця дипломна робота відкриває широкі можливості для подальших досліджень у сфері використання текстових квестів у навчанні. Нижче наведено деякі рекомендації щодо подальших напрямків досліджень:

1. Розширення функціоналу програмного забезпечення: Можливою розвиток програмного забезпечення є розширення його функціоналу. Наприклад, можна розглянути додавання нових функцій, таких як можливість використання мультимедійних елементів, що збагатять навчальний досвід, інтеграції з системами електронного забезпечення навчання, такими як Google Classroom, moodle.
2. Вдосконалення інтерфейсу користувача: Подальші дослідження можуть бути спрямовані на поліпшення інтерфейсу користувача. Дослідники можуть досліджувати різні методики та дизайн-принципи, які допоможуть

зробити інтерфейс більш інтуїтивно зрозумілим та привабливим для користувачів.

3. Дослідження ефективності використання: Для більш глибокого розуміння впливу використання текстових квестів у навчанні, можна провести дослідження, що оцінюють ефективність використання цих квестів на різних групах студентів. Важливо дослідити, які чинники та особливості квестів сприяють кращим результатам та більшій мотивації студентів.
4. Варіативність підходів до створення квестів: Дослідження можуть досліджувати різні підходи та методики створення текстових квестів. Наприклад, можна порівняти ефективність квестів, створених за допомогою різних методів, або дослідити вплив різних елементів квестів на навчальний процес.
5. Застосування в інших навчальних контекстах: Дослідження можуть розширити свій обсяг, розглядаючи використання текстових квестів у різних навчальних дисциплінах та контекстах. Можна дослідити, як квести впливають на навчання в різних галузях знань, таких як математика, література, історія та інші.
6. Розробка методик навчання на основі текстових квестів: Подальші дослідження можуть бути спрямовані на розробку методик навчання, які використовують текстові квести як основний навчальний інструмент. Дослідження можуть досліджувати оптимальні підходи до використання квестів, створюючи наочні та підвищуючи рівень засвоєння матеріалу.

Ці рекомендації щодо подальших досліджень можуть допомогти розширити наші знання та зрозуміння використання текстових квестів у навчанні та покращити їхній вплив на студентів.

# ВИСНОВКИ

Розділ "Висновки" містить підсумок проведеного дослідження з розробки програмного забезпечення для створення навчальних текстових квестів. У цьому розділі будуть наведені ключові висновки, отримані в результаті дослідження та реалізації програмного забезпечення. Загалом, висновки будуть розглядати ефективність та користь використання текстових квестів у навчанні, а також внесок дипломної роботи у сферу дослідження та можливі напрямки подальших досліджень.

Ефективність використання текстових квестів у навчанні

В ході дослідження та реалізації програмного забезпечення для створення навчальних текстових квестів було проведено серію експериментів та аналіз результатів. Згідно отриманих даних, використання текстових квестів у навчальному процесі позитивно впливає на мотивацію студентів та їхню академічну успішність. Квести створюють цікаву та інтерактивну навчальну атмосферу, сприяють розвитку критичного мислення та креативності студентів. Вони також допомагають у підвищенні залученості студентів до навчального матеріалу та покращенні їхнього розуміння теми. За допомогою створеного додатку можна не тільки ефективно створювати текстові квести, але й також використовувати їх в навчальному процесі. Текстові квести можуть використовуватись у наступних цілях, але не обмежуються цим переліком: донесення матеріалу до учня, оцінки рівня знань учня та розважання. Способи використання залежать від конкретних потреб навчаючого.

Внесок у сферу дослідження

Дипломна робота вносить свій внесок у сферу дослідження використання текстових квестів у навчанні. Результати дослідження та розроблене програмне забезпечення надають нові підходи та інструменти для створення та використання текстових квестів у навчальному процесі. Дослідження розкриває потенціал цього типу навчання і показує, як він може покращити якість та

ефективність навчання. Розроблене програмне забезпечення є цінним ресурсом для вчителів та студентів, які бажають впроваджувати інтерактивні методи навчання та покращити навчальний процес.

Рекомендації щодо подальших досліджень

На основі проведеного дослідження і реалізації програмного забезпечення можна сформулювати декілька рекомендацій щодо подальших досліджень у цій області. По-перше, рекомендується провести додаткові експерименти та аналіз результатів залучення текстових квестів у навчання різних дисциплін та на різних рівнях освіти. Це допоможе детальніше вивчити вплив квестів на різні групи студентів та визначити оптимальні умови їх використання.

Крім того, рекомендується дослідити можливості розширення функціональності програмного забезпечення. Наприклад, додати можливість створення багаторівневих квестів зі складнішими галузями, реалізувати механізм оцінювання результатів студентів після проходження квесту, розвинути інструменти для створення більш складних сценаріїв квестів.

Також, рекомендується провести порівняльний аналіз з іншими типами навчання та інтерактивними методиками, щоб визначити практичну значимість текстових квестів та їхню перевагу у навчальному середовищі.

Результати цих подальших досліджень можуть допомогти покращити ефективність навчального процесу та збільшити розуміння впливу текстових квестів на академічні показники студентів.

Загалом, дипломна робота привертає увагу до значущості використання текстових квестів у навчанні та надає підґрунтя для подальших досліджень у цій області. Результати дослідження вказують на потенціал текстових квестів у поліпшенні навчального процесу та сприяють розвитку інноваційних підходів до навчання.

# ПЕРЕЛІК ПОСИЛАНЬ

1. Farber, M. Using Quests for Project-Based Learning. Edutopia. URL: https://www.edutopia.org/article/using-quests-project-based-learning/ (дата звернення: 13.04.2023).

2. МОН України, «Overview of the current state of education and science in Ukraine in terms of russian aggression (as of January 2023)». URL: https://drive.google.com/file/d/19UxynvPVEXMVfwoUJcuxNirP1UWiXgSX/view (дата звернення: 20.05.2023).

3. МОН України, «ОСВІТА УКРАЇНИ В УМОВАХ ВОЄННОГО СТАНУ Інноваційна та проєктна діяльність», Київ 2022, ISBN 978-966-997-111-1, с. 112

4. Квести як спосіб ефективного урізноманітнення освітнього процесу. URL: https://vseosvita.ua/news/kvesty-iak-sposib-efektyvnoho-uriznomanitnennia-osvitnoho-protsesu-43687.html (дата звернення: 24.05.2023).

5. Overview of the Neo4j Graph Data Platform. URL: https://neo4j.com/developer-blog/overview-of-the-neo4j-graph-data-platform/ (дата звернення: 26.05.2023)

6. Python 3 – Overview. URL: https://www.tutorialspoint.com/python3/python_overview.htm (дата звернення: 26.05.2023)

7. neomodel - Python OGM for Neo4j. URL: https://neo4j.com/labs/neomodel/ (дата звернення: 24.05.2023)

8. What is an OGM? URL: https://neo4j.com/docs/ogm-manual/current/introduction/#introduction:ogm (дата звернення: 24.05.2023)

9. FastAPI. URL: https://fastapi.tiangolo.com/lo/ (дата звернення: 26.05.2023)

10. Overview of React.js. URL: https://www.patterns.dev/posts/reactjs (дата звернення: 01.06.2023)

11. Simran Kaur Arora, What is PyCharm? Features, Advantages & Disadvantages. URL: https://hackr.io/blog/what-is-pycharm (дата звернення: 06.04.2023)

# ДОДАТОК А

**Лістинги коду модулів створення, перегляду та зміни предметів, квестів та запитань**

## А.1 Модуль створення, перегляду та зміни предметів

```
from typing import Optional

import neomodel

from quests.db.models import Subject


def create_subject(name: str, description: str = None):
    s = Subject(name=name, description=description).save()
    return s.dict


def update_subject(subject_id: str, name: Optional[str],
description: Optional[str], image_url: Optional[str]):
    s = Subject.nodes.get(uid=subject_id)
    if name:
        s.name = name
    if description:
        s.description = description
    if image_url:
        s.image_url = image_url
    s.save()

    return s.dict


def get_subjects(skip: int, limit: int):
    subjects, meta = neomodel.db.cypher_query(
        "MATCH (s:Subject) RETURN s SKIP $skip LIMIT $limit",
        params={"skip": skip, "limit": limit},
        resolve_objects=True,
    )

    if len(subjects) == 0:
        return []
    return [x[0].dict for x in subjects]
```

## А.2 Модуль створення, перегляду та зміни квестів

```
from typing import Optional

import neomodel

from quests.db.models import Quest, Subject


class CrudError(Exception):
    pass


def create_quest(subject_id: str, name: str, description: str =
None):
    try:
        s = Subject.nodes.get(uid=subject_id)
    except Subject.DoesNotExist as e:
        raise CrudError(f"Subject {subject_id} not found") from e

    q = Quest(name=name, description=description).save()
    q.subject.connect(s)
    return q.dict


def update_subject(subject_id: str, quest_id: str, name:
Optional[str], description: Optional[str], image_url:
Optional[str]):
    try:
        s = Quest.nodes.get(uid=subject_id)
    except Quest.DoesNotExist as e:
        raise CrudError(f"Quest {quest_id} not found") from e

    q = Subject.nodes.get(uid=subject_id)
    if name:
        q.name = name
    if description:
        q.description = description
    if image_url:
        q.image_url = image_url
    q.save()

    return q.dict


def get_subject_quests(subject_id: str):
    quests, meta = neomodel.db.cypher_query(
        "MATCH (s:Subject {uid: $s_uid})<-[:BELONGS_TO]-(q:Quest)
RETURN q",
        params={"s_uid": subject_id},
        resolve_objects=True,
```

```
    )
    if len(quests) == 0:
        return []

    return [x[0].dict for x in quests]


def get_quest(subject_id: str, quest_id: str):
    quests, meta = neomodel.db.cypher_query(
        "MATCH (s:Subject {uid: $s_uid})<-[:BELONGS_TO]-(q:Quest
{uid: $q_uid}) RETURN q",
        params={"s_uid": subject_id, "q_uid": quest_id},
        resolve_objects=True,
    )
    if len(quests) == 0:
        return None
    return quests[0][0].dict


def get_quest_first_question(subject_id: str, quest_id: str):
    try:
        Subject.nodes.get(uid=subject_id)
    except Subject.DoesNotExist as e:
        raise CrudError(f"Subject {subject_id} not found") from e

    try:
        Quest.nodes.get(uid=quest_id)
    except Quest.DoesNotExist as e:
        raise CrudError(f"Quest {quest_id} not found") from e

    questions, meta = neomodel.db.cypher_query(
        "MATCH (s:Subject {uid: $s_uid})<-[:BELONGS_TO]-(q:Quest
{uid: $q_uid})<-[:PART_OF]-(que:Question) RETURN que",
        params={"s_uid": subject_id, "q_uid": quest_id},
        resolve_objects=True,
    )
    if len(questions) == 0:
        return None

    return questions[0][0].dict
```

### А.3 Модуль створення, перегляду, зміни запитань та збереження відповідей на запитання

```
from typing import Optional

import neomodel
```

```
from quests.db.models import Quest, Subject, Question, User
from .schemas import PreviousQuestion


class CrudError(Exception):
    pass


def create_question(
    subject_id: str,
    quest_id: str,
    name: str,
    question: str = None,
    answer: str = None,
    answer_type: str = "number",
    options: Optional[list[str]] = None,
    *,
    prev_questions: Optional[list[PreviousQuestion]] = None,
):
    if not prev_questions:
        prev_questions = []

    try:
        Subject.nodes.get(uid=subject_id)
    except Subject.DoesNotExist as e:
        raise CrudError(f"Subject {subject_id} not found") from e

    try:
        quest = Quest.nodes.get(uid=quest_id)
    except Quest.DoesNotExist as e:
        raise CrudError(f"Quest {quest_id} not found") from e

    q = Question(name=name, question=question,
answer_type=answer_type, answer=answer, options=options).save()
    for prev_question_data in prev_questions:
        try:
            prev_question =
Question.nodes.get(uid=prev_question_data.id)
        except Question.DoesNotExist as e:
            raise CrudError(f"Question {prev_question_data.id} not
found") from e
        q.from_questions.connect(prev_question, {"answer":
prev_question_data.leading_answer, "points": 0})
    if not prev_questions:
        q.quest.connect(quest)
    return q.dict


def get_quest_questions(subject_id: str, quest_id: str):
    questions, meta = neomodel.db.cypher_query(
        "MATCH (s:Subject {uid: $s_uid})<-[:BELONGS_TO]-(q:Quest
{uid: $q_uid})-[:PART_OF|NEXT_QUESTION*]-(que:Question) RETURN
que",
```

```
        params={"s_uid": subject_id, "q_uid": quest_id},
        resolve_objects=True,
    )
    if len(questions) == 0:
        return None

    return [x[0].dict for x in questions]


def get_question(subject_id: str, quest_id: str, question_id:
str):
    questions, meta = neomodel.db.cypher_query(
        "MATCH (s:Subject {uid: $s_uid})<-[:BELONGS_TO]-(q:Quest
{uid: $q_uid})<-[*]-(que:Question {uid: $que_uid}) RETURN que",
        params={"s_uid": subject_id, "q_uid": quest_id, "que_uid":
question_id},
        resolve_objects=True,
    )
    if len(questions) == 0:
        return None

    return questions[0][0].dict


def get_next_question(subject_id: str, quest_id: str, question_id:
str, answer: str):
    try:
        Subject.nodes.get(uid=subject_id)
    except Subject.DoesNotExist as e:
        raise CrudError(f"Subject {subject_id} not found") from e

    try:
        Quest.nodes.get(uid=quest_id)
    except Quest.DoesNotExist as e:
        raise CrudError(f"Quest {quest_id} not found") from e

    questions, meta = neomodel.db.cypher_query(
        "MATCH (s:Subject {uid: $s_uid})<-[:BELONGS_TO]-"
        "(q:Quest {uid: $q_uid})-[:PART_OF|NEXT_QUESTION*]-"
        "(q1:Question {uid: $prev_q_uid})-[:NEXT_QUESTION {answer:
$answer}]->"
        "(que:Question) "
        "RETURN que",
        params={"s_uid": subject_id, "q_uid": quest_id,
"prev_q_uid": question_id, "answer": answer},
        resolve_objects=True,
    )
    if len(questions) == 0:
        return None

    return questions[0][0].dict
```

```
def submit_answer(user_id: str, subject_id: str, quest_id: str,
question_id: str, *, answer):
    try:
        user = User.nodes.get(uid=user_id)
    except User.DoesNotExist as e:
        raise CrudError(f"User {user_id} does not exist") from e

    try:
        Subject.nodes.get(uid=subject_id)
    except Subject.DoesNotExist as e:
        raise CrudError(f"Subject {subject_id} does not exist")
from e

    try:
        quest = Quest.nodes.get(uid=quest_id)
    except Quest.DoesNotExist as e:
        raise CrudError(f"Quest {quest_id} does not exist") from e

    try:
        question = Question.nodes.get(uid=question_id)
    except Question.DoesNotExist as e:
        raise CrudError(f"Question {question_id} does not exist")
from e

    next_question_rel = neomodel.db.cypher_query(
        "MATCH (q:Question {uid: $q_uid})-[r:NEXT_QUESTION
{answer: $answer}]->() RETURN r",
        params={"q_uid": question_id, "answer": answer},
        resolve_objects=True,
    )
    if len(next_question_rel[0]) > 0:
        points = next_question_rel[0][0][0].points
    elif question.answer == answer and question.points:
        points = question.points
    else:
        points = None

    user.answered_questions.connect(question, {"earned_points":
points or 0, "answer": answer})


def get_total_points(subject_id: str, quest_id: str):
    points = neomodel.db.cypher_query(
        "MATCH (:User)-[r:ANSWERED]->(:Question)-
[:NEXT_QUESTION|PART_OF*]-(:Quest {uid: $q_uid})-[:BELONGS_TO]-
>(:Subject {uid: $s_uid}) RETURN sum(r.earned_points)",
        params={"q_uid": quest_id, "s_uid": subject_id},
    )
    return points[0][0][0]


def get_max_points(subject_id: str, quest_id: str):
    try:
```

```
        points = neomodel.db.cypher_query(
        "MATCH (:Subject {uid: $s_uid})<-[:BELONGS_TO]-(:Quest
{uid: $q_uid})<-[:PART_OF]-(q:Question)-[r:NEXT_QUESTION]-
>(:Question) "
        "WITH MAX(r.points) as rpoints, q.points as qpoints "
        "WITH SUM(CASE WHEN rpoints>=qpoints THEN rpoints ELSE
qpoints END) as s "
        "MATCH (:Subject {uid: $s_uid})<-[:BELONGS_TO]-(:Quest
{uid: $q_uid})<-[:PART_OF]-(:Question)-[r:NEXT_QUESTION*]-
>(q:Question) "
        "RETURN q.points + s",
        params={"q_uid": quest_id, "s_uid": subject_id},
    )[0][0][0]
    except IndexError:
        points = 0
    return points
```

# ДОДАТОК Б

## Лістинги коду модулю взаємодії з базою даних Neo4j

### Б.1 Лістинг коду моделі користувача

```
from neomodel import (
    StructuredNode,
    StringProperty,
    RelationshipTo,
    RelationshipFrom,
    UniqueIdProperty,
    StructuredRel,
    IntegerProperty,
    DateTimeProperty,
    EmailProperty,
)


class UserAnsweredQuestion(StructuredRel):
    earned_points = IntegerProperty(default=0)
    answer = StringProperty()


class UserCompletedQuest(StructuredRel):
    total_points = IntegerProperty(default=0)
    completed_timestamp = DateTimeProperty(default_now=True)


class User(StructuredNode):
    uid = UniqueIdProperty()
    name = StringProperty()
    email = EmailProperty()

    authored_quests =
RelationshipFrom("quests.db.models.quest.Quest", "AUTHOR")
    answered_questions =
RelationshipTo("quests.db.models.question.Question", "ANSWERED",
model=UserAnsweredQuestion)
    completed_quests =
RelationshipTo("quests.db.models.quest.Quest", "COMPLETED",
model=UserCompletedQuest)

    @property
    def dict(self):
        return {
            "id": self.uid,
            "name": self.name,
```

```
            "email": self.email,
        }
```

### Б.2 Лістинг коду моделі предмету

```
import json

from neomodel import StructuredNode, StringProperty,
RelationshipFrom, UniqueIdProperty


class Subject(StructuredNode):
    uid = UniqueIdProperty()
    name = StringProperty()
    description = StringProperty()
    image_url = StringProperty()

    quests = RelationshipFrom("quests.db.models.quest.Quest",
"BELONGS_TO")

    @property
    def dict(self):
        return {
            "id": self.uid,
            "name": self.name,
            "description": self.description,
            "imageUrl": self.image_url,
        }

    @property
    def json(self):
        return json.dumps(self.dict)
```

### Б.3 Лістинг коду моделі квесту

```
from neomodel import StructuredNode, StringProperty,
RelationshipTo, RelationshipFrom, UniqueIdProperty


class Quest(StructuredNode):
    uid = UniqueIdProperty()
    name = StringProperty()
    description = StringProperty()
    image_url = StringProperty()
```

```
    subject = RelationshipTo("quests.db.models.subject.Subject",
"BELONGS_TO")
    author = RelationshipTo("quests.db.models.user.User",
"AUTHOR")
    questions =
RelationshipFrom("quests.db.models.question.Question", "PART_OF")

    @property
    def dict(self) -> dict:
        return {
            "id": self.uid,
            "name": self.name,
            "description": self.description,
            "imageUrl": self.image_url,
        }
```

### Б.4 Лістинг коду моделі запитання

```
from neomodel import (
    StructuredNode,
    StringProperty,
    RelationshipTo,
    RelationshipFrom,
    Relationship,
    UniqueIdProperty,
    StructuredRel,
    IntegerProperty,
    ArrayProperty,
)


class QuestionConnector(StructuredRel):
    points = IntegerProperty(default=0)
    answer = StringProperty()


class Question(StructuredNode):
    ANSWER_TYPES = {"number": "number", "option": "option"}

    uid = UniqueIdProperty()
    name = StringProperty()
    question = StringProperty()
    answer = StringProperty()
    points = IntegerProperty()
    answer_type = StringProperty(ANSWER_TYPES)
    options = ArrayProperty(StringProperty())

    next_questions = RelationshipTo("Question", "NEXT_QUESTION",
model=QuestionConnector)
```

```
    from_questions = RelationshipFrom("Question", "NEXT_QUESTION",
model=QuestionConnector)
    quest = Relationship("quests.db.models.quest.Quest",
"PART_OF")

    @property
    def dict(self):
        return {
            "id": self.uid,
            "name": self.name,
            "question": self.question,
            "answer": self.answer,
            "answerType": self.answer_type,
            "options": self.options,
        }
```