Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБКА ВЕБЗАСТОСУНКУ ГЕНЕРАЦІЇ ТВОРЧИХ ІДЕЙ
(тема)

Виконав:
студент 4 курсу, групи ІТІНФ-20-1
Богдан Н.І.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Шафроненко А.Ю.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Богдан Надії Ігорівні
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка вебзастосунку генерації творчих ідей

затверджена наказом університету від 20 травня 2024 року № 003 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 24 травня 2024 р.

3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі, мовна модель GPT-2, мова програмування JavaScript, мова програмування Python.

4. Перелік питань, що потрібно опрацювати в роботі
1. Теоретичний огляд методів генерації тексту та постановка задачі.
2. Визначення технологій розробки та моделювання бази даних.
3. Програмна реалізація генератора та демонстрація результату.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Актуальність проблеми генерації творчих ідей, постановка задачі, запит для генерації.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	08.04.2024	
2	Аналіз завдання, підбір літератури	08.04.24-09.04.24	
3	Аналіз літератури з досліджуваної проблеми	10.04.24-11.04.24	
4	Аналіз технічних засобів	12.04.24-14.04.24	
5	Розробка методу	15.04.24-17.04.24	
6	Програмна реалізація	18.04.24-02.05.24	
7	Оформлення пояснювальної записки	20.05.24-21.05.24	
8	Перевірка на плагіат	27.05.24	
9	Рецензування	28.05.24	
10	Підготовка презентації та доповіді	29.05.24-02.06.24	
11	Занесення роботи в електронний архів	03.06.24	
12	Попередній захист кваліфікаційної роботи	03.06.24	

Дата видачі завдання 8 квітня 2024 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Шафроненко А.Ю.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 63 с., 1 табл., 41 рис., 40 джерел.

КЛІЄНТ-СЕРВЕР, ВЕБЗАСТОСУНОК, VISUAL STUDIO CODE, MYSQL, МОВА ЗАПИТІВ SQL, HTML, CSS, JAVASCRIPT, NODE.JS, PYTHON, МОВНА МОДЕЛЬ GPT-2, ГЕНЕРАТОР ІДЕЙ.

Об’єктом роботи є аналіз творчих ідей використанням мовної моделі GPT-2.

Метою роботи є розробка вебзастосунку генерації творчих ідей з використанням мовної моделі GPT-2, що дозволяє створювати унікальний творчий контент на основі користувацького запиту.

Застосовано логіку генерації даних на основі мови розробки JavaScript. Використано технологію створення текстового контенту на базі мовної моделі GPT-2 посередництвом мови програмування Python. Розроблено алгоритми фільтрації користувацьких запитів. Реалізовано механізми збереження ідей користувачів у базі даних для подальшого використання. Також реалізовано інтерфейс для взаємодії з користувачем, який дозволяє вводити запити і отримувати унікальні творчі ідеї від вебзастосунку.

У результаті роботи здійснена програмна реалізація вебзастосунку генерації творчих ідей.

CLIENT-SERVER, WEBAPPLICATION, VISUAL STUDIO CODE, MYSQL, SQL QUERY LANGUAGE, HTML, CSS, JAVASCRIPT, NODE.JS, PYTHON, GPT-2 LANGUAGE MODEL, IDEA GENERATOR.

The object of the work is the analysis of creative ideas using a language model GPT-2.

The aim of the work is to develop a webapplication for generating creative ideas using the GPT-2 language model, which allows creating unique creative content based on user requests.

The logic of data generation is implemented based on JavaScript development language. The technology for creating text content is based on the GPT-2 language model through Python programming language. Algorithms for filtering user requests have been developed. Mechanisms for saving user ideas in a database for further use have been implemented. An interface for interacting with the user has also been implemented, allowing input of requests and receiving unique creative ideas from the webapplication.

As a result of the work, a software implementation of a webapplication for generating creative ideas has been carried out.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

HTML – HyperText Markup Language (мова розмітки гіпертексту)

CSS – Cascading Style Sheets (каскадні таблиці стилів)

JS – JavaScript (джаваскрипт)

БД – база даних

СУБД – система управління базами даних

GPT-2 – Generative Pre-trained Transformer 2 (генеративний передньо-навчений трансформер 2)

SQL – Structured Query Language (мова структурованих запитів)

WEB – World Wide Web (всесвітня павутина)

JSON – JavaScript Object Notation (нотація об’єктів JavaScript)

HTTP – HyperText Transfer Protocol (протокол передачі гіпертексту)

UI – User Interface (інтерфейс користувача)

IDE – Integrated Development Environment (інтегроване середовище розробки)

API – Application Programming Interface (інтерфейс програмування застосунків)

ШІ – штучний інтелект

JWT – JSON Web Token (токен веб-сервісу JSON)

URL – Uniform Resource Locator (уніфікований локатор ресурсу)

V8 – JavaScript Engine (движок JavaScript)

ВСТУП

Розглядаючи сучасний розвиток технологій, очевидно, що комп'ютеризація все більше охоплює різні сфери людської діяльності. Від біоінженерії до сфери розваг, від освіти до бізнесу, комп'ютерні технології у тій чи іншій мірі впливають на всі аспекти нашого життя. Одним з таких аспектів є творчість, яка розкривається новими можливостями завдяки інноваційним розробкам в галузі інформаційних технологій.

Саме у цьому контексті розглядається можливість використання мовної моделі GPT-2 для генерації творчих ідей у розробці вебзастосунку генератора. GPT-2, яка є однією з передових моделей глибокого навчання, здатна аналізувати величезні масиви текстової інформації та генерувати унікальний контент на основі аналізу даних. Вона відкриває нові горизонти для створення та розробки творчого контенту, завдяки чому користувачі можуть отримати новий інструмент для отримання ідей, які можуть бути застосовані у різних сферах їхньої діяльності.

Вебзастосунок є актуальним у сучасному світі, де інновації займають важливе місце у розвитку сьогоденного суспільства. Він надає можливість користувачам швидко та ефективно генерувати нові ідеї, що сприяє стимулюванню творчого процесу у шукачів натхнення та розвитку нових концепцій та продуктів.

Загальний внесок роботи полягає у створенні вебзастосунку, який поєднує у собі творче поле з технологіями веброзробки. Результатом проєкту є функціональний інструмент, який може бути використаний для стимулювання творчості та генерації нових ідей у різних галузях творчої діяльності.

Актуальність роботи полягає у поєднанні вебтехнологій з мовною моделлю GPT-2. Робота має потенціал стати корисним інструментом для креативних професіоналів, письменників та інших фахівців, які постійно шукають нові ідеї та способи їх створення.

1 ТЕОРЕТИЧНИЙ ОГЛЯД МЕТОДІВ ГЕНЕРАЦІЇ ТЕКСТУ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Огляд текстових генераторів

Текстовий генератор – це програмна система або алгоритм, який автоматично створює новий текст на основі введеного користувачем або заздалегідь визначеного текстового шаблону. Основна мета текстового генератора полягає в тому, щоб створити вміст, який має логічний зв’язок та відповідає вимогам користувача, тобто конкретної задачі. Генерація тексту у цілому має широкий спектр застосувань, включаючи створення контенту для вебсайтів, чат-ботів, автоматичний переклад тексту та багато чого іншого [1].

Щодо методів текстової генерації, одним з найпоширеніших підходів є використання мовних моделей, які базуються на глибокому та машинному навчанні. Вони вивчають структуру та зв’язки в мові на основі великого обсягу текстових даних. Мовні моделі пройшли великий шлях за останні роки завдяки появі трансформерних моделей, таких як GPT, що призвело до значного стрибка у можливостях генерації природної мови. Здатність генерувати зв’язний та високоякісний текст зробила GPT популярним вибором для різноманітних завдань з генерації тексту [2].

Наприклад, GPT-2 є передовою мовною моделлю, розробленою OpenAI. Вона базується на архітектурі трансформера і була передньо-навчена на широкому спектрі текстових даних з Інтернету. GPT-2 відома своєю здатністю генерувати високоякісний, контекстуально зв’язаний текст, який майже неможливо відрізнити від контенту, написаного людиною.

Глибокий нейронний мережевий алгоритм вражає своєю здатністю генерувати тексти, які надзвичайно реалістичні та логічно продумані. В основі цієї технології лежить масштабна модель машинного навчання, яка навчилася на великій кількості різноманітних текстів. Ця модель може «відтворювати»

стиль, тон та сенс будь-якого вхідного тексту, що дозволяє їй генерувати зразки, які мимовільно вражають своєю автентичністю.

Така здатність до адаптації дозволяє GPT-2 стати потужним інструментом у багатьох сферах, від створення реалістичних текстів для досліджень та аналізу до розважальних застосувань, таких як генерація контенту для ігор або фільмів. Завдяки цій технології виникає можливість використання мовної моделі для навчання, реклами, створення контенту для соціальних медіа та багатьох інших різноманітних цілей.

GPT-2 генерує синтетичні зразки тексту у відповідь на те, що модель налаштована на довільний вхід. Модель подібна на хамелеона – вона адаптується до стилю та змісту умовного тексту. Це дозволяє генерувати реалістичний контент та логічно продовжувати розмову на обрану тему [3].

1.2 Поняття вебзастосунку

Коли Інтернет був тільки що винайдений, вебсайти мали значно менше функціональності, ніж вебзастосунки. Вони були здатні лише до забезпечення інформацією користувачів через статичний контент. Під час їх використання доводилося встановлювати та запускати програмне забезпечення зі складною функціональністю. У той час як вебзастосунки були створені для того, щоб зменшити відмінність між програмним забезпеченням і статичними сайтами. У них з’явилася функціональність та інтерактивні елементи для користувача, як у програмному забезпеченні, але вони були подані за допомогою веббраузера за URL-адресою [4].

Таким чином, вебзастосунок – це програмне забезпечення, яке працює безпосередньо у веббраузері і надає інтерактивний інтерфейс для користувача в мережі Інтернет. Його не потрібно завантажувати, оскільки він доступний через мережу. Користувачі можуть отримати доступ до вебзастосунку через веббраузер, такий як Google Chrome, Mozilla Firefox або ж Safari тощо.

Основною метою вебзастосунку є створення зручного та ефективного інтерфейсу для користувачів, що дозволяє їм взаємодіяти з вебсервером та отримувати необхідну інформацію. Вони можуть бути використані для різноманітних цілей, від простого відображення інформації до складних операцій, таких як онлайн-торгівля, соціальні мережі, або ж великі системи управління контентом.

Для роботи та повноцінного функціонування вебзастосунка потрібен вебсервер, сервер застосунків та база даних. Вебсервери керують запитами, що надходять від клієнта, тоді як сервер застосунків виконує запитану задачу. База даних зберігає будь-яку необхідну інформацію.

Скрипт на клієнтській стороні відповідає за функціональність UI, наприклад, це можуть бути кнопки або випадаючі списки. Коли, наприклад, користувач натискає на посилання вебзастосунку, веббраузер завантажує скрипт на клієнтській стороні та відображає графічні елементи із текстом для взаємодії з користувачем. Це відбувається, коли він може читати вміст, дивитися відео або заповнювати дані у формі оберненого зв’язку. Дії, такі як натискання кнопки відправки, переходять на сервер у вигляді запиту від клієнта.

Скрипт на серверній стороні відповідає за обробку даних. Сервер вебзастосунків обробляє запити клієнтів та надсилає їм відповідь. Зазвичай запити стосуються отримання додаткових даних або редагування та збереження нових. Наприклад, якщо користувач натискає кнопку «Читати далі», сервер вебзастосунку надсилатиме вміст користувачеві. Якщо користувач натискає кнопку «Відправити», сервер застосунку збереже дані користувача у базі даних. У деяких випадках сервер завершує запит на дані та надсилає повну HTML-сторінку клієнту. Це називається відображенням на стороні сервера.

База даних є важливою складовою будь-якого вебзастосунку, оскільки вона забезпечує зберігання та організацію інформації, яка потрібна для правильного функціонування системи. База даних може бути реалізована у

різних форматах, таких як реляційні бази даних (наприклад, MySQL, PostgreSQL), нереляційні бази даних (наприклад, MongoDB, Cassandra) або гібридні, які можуть комбінувати різні підходи.

Реляційні бази даних є одними з найпоширеніших та використовуються для зберігання структурованої інформації у вигляді таблиць з рядками та стовпцями. Кожна таблиця може мати свій унікальний ідентифікатор (ключ), який дозволяє легко знаходити та оновлювати дані. Реляційні бази даних часто використовуються там, де потрібно виконувати складні операції зв’язку між різними типами даних.

База даних взаємодіє з сервером застосунків шляхом виконання запитів на отримання, оновлення, додавання або видалення даних. Це забезпечує збереження інформації користувачів, їх налаштувань, транзакцій та іншої важливої інформації, необхідної для правильної роботи вебзастосунку.

Вебзастосунки зазвичай мають короткі цикли розробки та невеликі розробницькі команди. Більшість вебпрограм створюється на JavaScript, HTML та CSS. Клієнтське програмування зазвичай використовує ці мови, так як вони і створюють фронтенд програми. Серверне програмування пише скрипти, якими буде користуватися вебзастосунок. Мови, такі як Python, Java або інші, зазвичай використовуються в серверному програмуванні [5-7].

Розвиток, збагачення потрібним функціоналом та керування вебзастосунком є задачею команди адміністраторів, які відповідають за забезпечення контентом. Звісно, вони співпрацюють з розробниками для визначення стратегії та шляхів розвитку вебзастосунку [8].

За допомогою сучасних технологій можна створювати різноманітні програми та інструменти для повноцінних вебзастосунків. Однак це також ставить перед учасниками проєкту високі вимоги до кваліфікації, оскільки робота з багатофункціональними і складними завданнями вимагає глибоких знань. Розробка великих вебзастосунків потребує висококваліфікованих фахівців [9].

Загалом, вебзастосунки стали невід’ємною частиною сучасного Інтернет-світу. Вони забезпечують користувачам широкі можливості взаємодії та доступу до інформації з будь-якого пристрою з підключенням до Інтернету.

1.3 Огляд вебзастосунків, призначених для створення текстового контенту

1.3.1 Приклади творчих генераторів

Сучасний вебпростір пропонує безліч інтернет-ресурсів, які сприяють генерації творчого текстового контенту. Ці ресурси включають у себе різноманітні генератори творчих ідей, які розроблені для надання користувачам унікальних концепцій та ідей у різних сферах діяльності.

Генератори творчих ідей можуть бути розділені на такі категорії:

– генератори персонажів, які спрямовані на створення вигаданих чи реалістичних особистостей для літературних, графічних проєктів або ж задля іншої мети, яку переслідує митець. Вони можуть надавати інформацію про вік, зовнішність, характеристики особистості та інші атрибути персонажа;

– генератори імен персонажів, які допомагають авторам знайти відповідні та оригінальні імена для своїх героїв. Ці інструменти можуть базуватися на різних культурах, епохах або фантастичних світах, забезпечуючи різноманітність вибору;

– генератори сюжетів, що надають короткі описи ідеї сюжету для літературних або кінематографічних проєктів тощо. Ці інструменти можуть включати ключові події, конфлікти, розвиток сюжету та інші деталі;

– генератори локацій, які створюють описи або концепції уявних місць, де можуть відбуватися дії твору користувача. Вони можуть описувати пейзажі, архітектуру, кліматичні умови та інші аспекти локацій.

Це не повний список усіх можливих ідейних ресурсів. У цілому, такі генератори є допоміжним інструментом для творчих особистостей, які шукають нові ідеї та натхнення для своїх проєктів. Вони дозволяють швидко та ефективно генерувати концепції, що стимулює творчий процес та сприяє розвитку нових продуктів [10].

Генератори творчих ідей загалом надають користувачеві можливість фільтрувати дані згідно із категорією, як показано на рисунку 1.1 [11].

Рисунок 1.1 – Приклад генератора сюжетів – головна сторінка застосунку

Користувач має обрати категорію або тему, для якої він бажає отримати творчі ідеї. Наприклад, якщо користувач працює над літературним твором, він може вибрати категорію «Сюжети» або «Персонажі» у генераторі творчих ідей. Після вибору категорії користувач може отримати доступ до різних варіантів ідей, які відповідають його вибору.

Такий підхід дозволяє користувачам зосередитися на конкретній темі або напрямку творчості та отримати ідеї, які відповідають їхнім потребам та інтересам. Фільтрація даних також допомагає збільшити ефективність процесу генерації ідей, оскільки користувач може отримати лише ті ідеї, які відповідають обраній категорії або темі. Це сприяє збільшенню

продуктивності та забезпечує користувачам більш точне та цілеспрямоване отримання творчого натхнення.

1.3.2 Загальна характеристика творчих генераторів

Творчі генератори мають різноманітні характеристики та функціональні можливості. Звісно, унікальні особливості залежать від конкретного вебзастосунку та його контенту. На рисунку 1.2 показано розділ за категоріями генерованих даних [12].

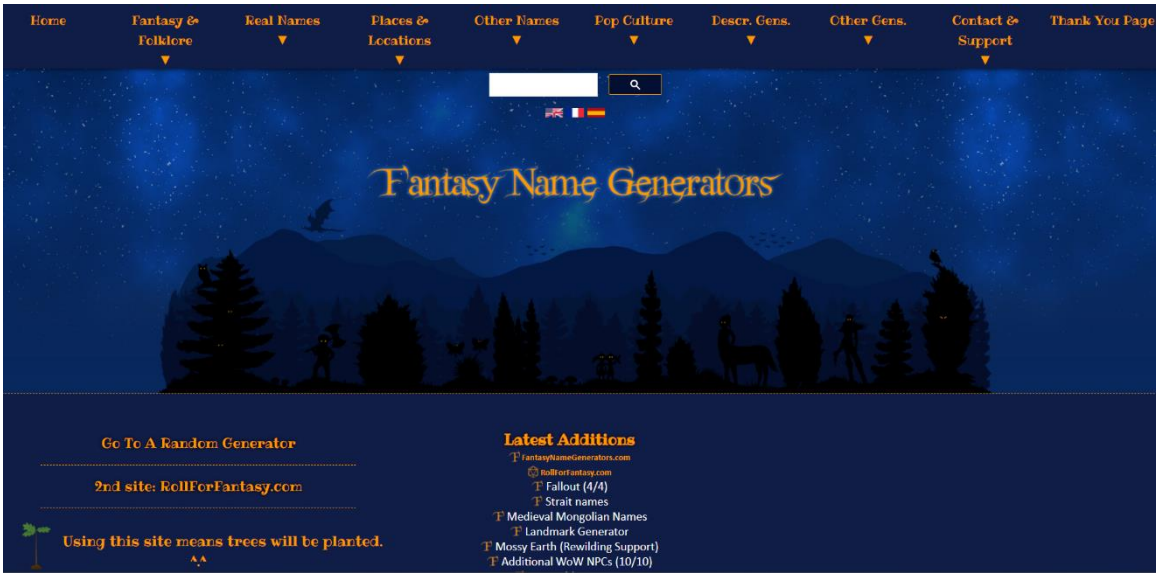

Рисунок 1.2 – Приклад генератора імен – головна сторінка застосунку

Переважна більшість застосунків надає послуги користувачам безкоштовно або ж за реєстрацією на сайті. Однак, чим потужнішим є функціонал, тим більшою є потреба у захисті даних – наприклад, користуватися генерацією можуть лише зареєстровані користувачі – як показано на рисунку 1.3 [13].

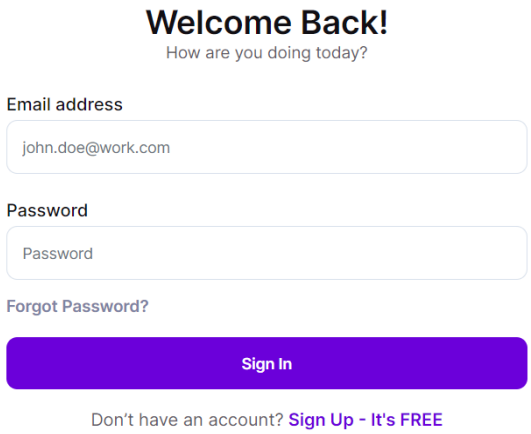

Рисунок 1.3 – Сторінка авторизації у застосунку генератора імен

Деякі вебзастосунки забезпечені засобами штучного інтелекту для більш унікального контенту.

Наприклад, генератори творчих ідей можуть використовувати методи машинного навчання для аналізу великих обсягів даних, що дозволяє їм створювати більш різноманітний та оригінальний контент.

Штучний інтелект в таких вебзастосунках може використовуватися для автоматичної генерації текстового контенту на основі аналізу попередніх користувацьких запитів або шаблонів. Наприклад, система може аналізувати вибори користувача, його інтереси, реакції на попередні ідеї та інші параметри для створення індивідуалізованого контенту, який відповідає його потребам та очікуванням. Це дозволяє підвищити якість та різноманітність згенерованого контенту, забезпечуючи користувачам більш насичений та цікавий досвід. Крім того, використання штучного інтелекту дозволяє вебзастосункам адаптуватися до індивідуальних потреб користувачів та постійно вдосконалюватися за рахунок збору та аналізу даних про їхні взаємодії [10].

Використання новітніх текстових моделей, таких як GPT-4, може включати в себе платні послуги. Це зумовлено великими витратами на розвиток та підтримку таких моделей, а також високою вартістю обчислювальних ресурсів, необхідних для їхньої роботи.

Платні послуги можуть включати в себе доступ до розширених функцій моделі, збільшення лімітів використання, покращену підтримку клієнтів та інші додаткові можливості. Користувачі, які платять за такі послуги, можуть мати доступ до більшого обсягу функцій та отримувати переваги в порівнянні з користувачами безплатних версій.

Популярні вебзастосунки, які використовують моделі штучного інтелекту для генерації текстового контенту, часто пропонують преміальні плани або підписки, що надають доступ до додаткових можливостей за певну плату – як це продемонстровано на рисунку 1.4 [14]. Це дозволяє розробникам забезпечувати стабільний дохід і зберігати високу якість своїх продуктів.

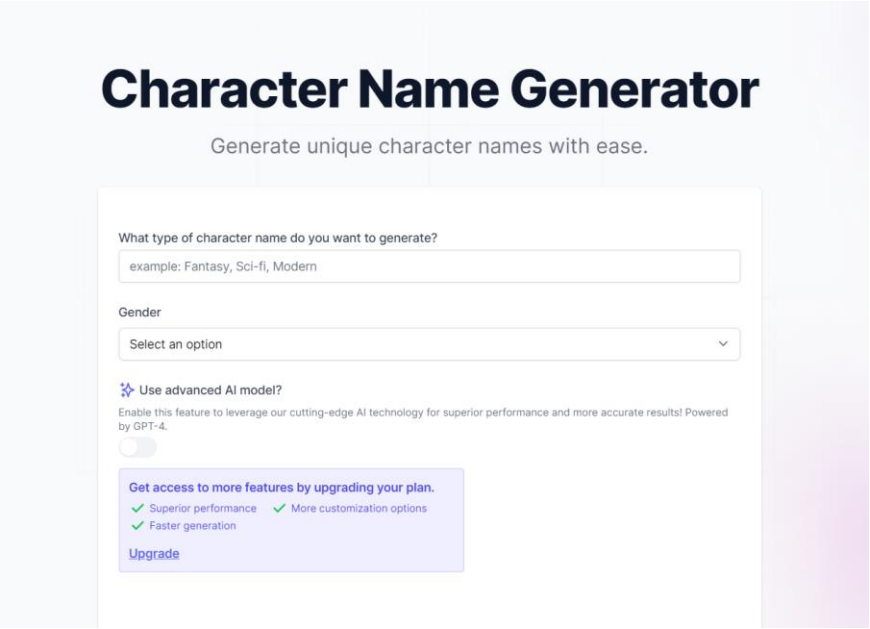

Рисунок 1.4 – Приклад генератора імен, який пропонує можливості мовної моделі GPT-4

1.4 Виявлені недоліки існуючих програмних засобів генерації

Шукачі творчого натхнення в сучасному цифровому світі мають доступ до безлічі вебресурсів, що надають різноманітний творчий контент. Від генераторів імен персонажів до ідей для сюжету чи описів локацій, Інтернет буквально переповнений ресурсами, які пропонують засоби для стимулювання творчості.

Однак, серед цієї різноманітності важко знайти повноцінний вебзастосунок, який би об’єднував усі необхідні категорії генерації та дозволяв створювати єдиний сюжет без переходу між різними програмними інструментами та міг надати логічний зв’язок між згенерованими даними.

Розрізнений функціонал стає однією з проблем існуючих генераторів ідей. Наприклад, для створення повноцінного сюжету користувачу може знадобитися використання кількох різних вебзастосунків: один для генерації імен персонажів, інший для створення сюжету, а ще інший для опису локацій. Це ускладнює процес творення, роблячи його більш витратним на час та зусилля, а також створює проблему у поєднанні даних.

Крім того, багато вебресурсів, що пропонують творчий контент, працюють на платній основі. Хоча деякі з них можуть надати обмежену безкоштовну версію, повний функціонал, часто, доступний тільки за певну плату. Це може стати перешкодою для творчих людей з обмеженим бюджетом, які шукають доступні та зручні інструменти для стимулювання своєї уяви.

Отже, існує потреба в створенні комплексного вебзастосунку, який об’єднає різноманітні категорії генерації та дозволить користувачам легко створювати єдині сюжети та історії без необхідності переходити між різними інструментами та платити за доступ до повного функціоналу.

1.5 Постановка задачі

Таким чином, розробка вебзастосунку генерації творчих ідей є актуальним завданням для вирішення поставленої проблеми. Тому ставиться завдання програмної реалізації генератора із застосуванням мов програмування JavaScript та Python, а також технологій мовної моделі GPT-2.

Об’єктом роботи є аналіз творчих ідей використанням мовної моделі GPT-2.

Метою роботи є розробка вебзастосунку генерації творчих ідей з використанням мовної моделі GPT-2, що дозволяє створювати унікальний творчий контент на основі користувацького запиту.

Для досягнення мети необхідно вирішити такі завдання:

– ознайомитися з існуючими програмними засобами генерації творчих ідей, виявити їх недоліки та переваги;

– розробити концепцію створення вебзастосунку;

– опанувати необхідні технології розробки;

– створити логіку генерації творчого контенту;

– розробити клієнтську частину застосунку згідно із вимогами до користувацького інтерфейсу;

– створити серверну частину, використовуючи технології генерації;

– реалізувати звя’зок застосунку із БД;

– розгорнути робочу програму застосунку.

2 ВИЗНАЧЕННЯ ТЕХНОЛОГІЙ РОЗРОБКИ ТА МОДЕЛЮВАННЯ БАЗИ ДАНИХ

2.1 Огляд технічного стеку для розробки вебзастосунку

2.1.1 Технології клієнтської частини

Частина вебсайту, з якою користувач безпосередньо взаємодіє, називається фронтендом. Також він відомий як клієнтська частина програми.

Клієнтська частина є надзвичайно важливою, так як саме через неї користувач взаємодіє із програмою.

Фронтенд включає все, що бачать користувачі: кольори та стилі тексту, зображення, графіки та таблиці, кнопки, кольори, меню навігації тощо. HTML, CSS та JavaScript – це мови, які використовуються для розробки фронтенду. Адаптивність та продуктивність – це дві основні цілі клієнтської частини застосунку.

Під час розробки фронтенду особливу увагу приділяють технологіям, що використовуються для створення цієї частини програми.

Фронтенд складається з програмних технологій, що забезпечують інтерактивність вебсторінок. Взаємодія користувача із кнопками, навігація по сторінках вебсайту – усе це результат роботи технологій клієнтської частини застосунку.

Наприклад, технологія HTML визначає склад елементів на вебсторінці, їхню безпосередню структуру, CSS відповідає за зовнішній вигляд, стиль та розташування на сторінці, а JavaScript насамперед забезпечує динамічну поведінку та взаємодію з користувачем.

Таким чином, HTML, CSS та JavaScript є складовими технологіями клієнтської частини вебзастосунків (рис. 2.1) [15].

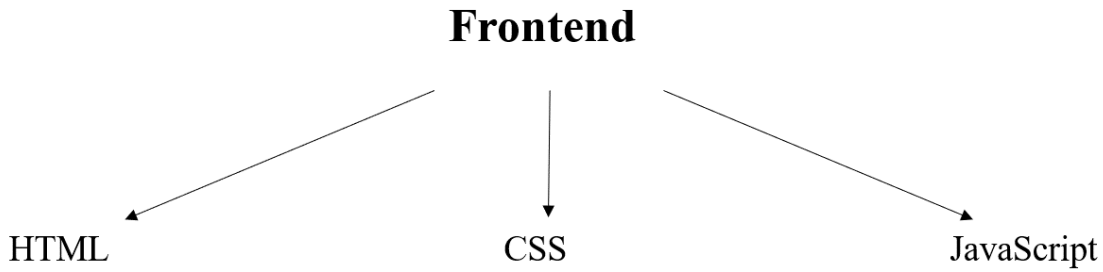

Рисунок 2.1 – Програмні технології фронтенду

JavaScript – це найпотужніша та універсальна мова програмування, що використовується у веброзробці. Вона є легкою, крос-платформною, однопоточною та інтерпретованою. Це загальноприйнята мова програмування для створення динамічних та інтерактивних елементів у вебзастосунках. Вона легка у вивченні, що спрощує її застосування. JavaScript використовується для додавання інтерактивності та функціональності до вебсторінок. Вона дозволяє створювати динамічні елементи, обробляти події користувача та взаємодіяти з сервером без перезавантаження сторінки. Використання JavaScript у проєкті обумовлене її широким функціональним спектром, що спрощує розробку та підтримку клієнтської частини програми [16].

Не менш важливою складовою фронтенд частини є HTML. HTML – це мова вебу, що використовується мільярдами вебсайтів для створення сторінок, які ми бачимо щодня. HTML визначає елементи та їх розташування на сторінці, що робить її необхідним елементом для будь-якого вебзастосунку. Використання мови розмітки у проєкті обумовлене її ключовою роллю у веброзробці та простотою в оволодінні [17].

На рисунку 2.2 продемонстровано структуризацію елементів сторінки мовою розмітки.

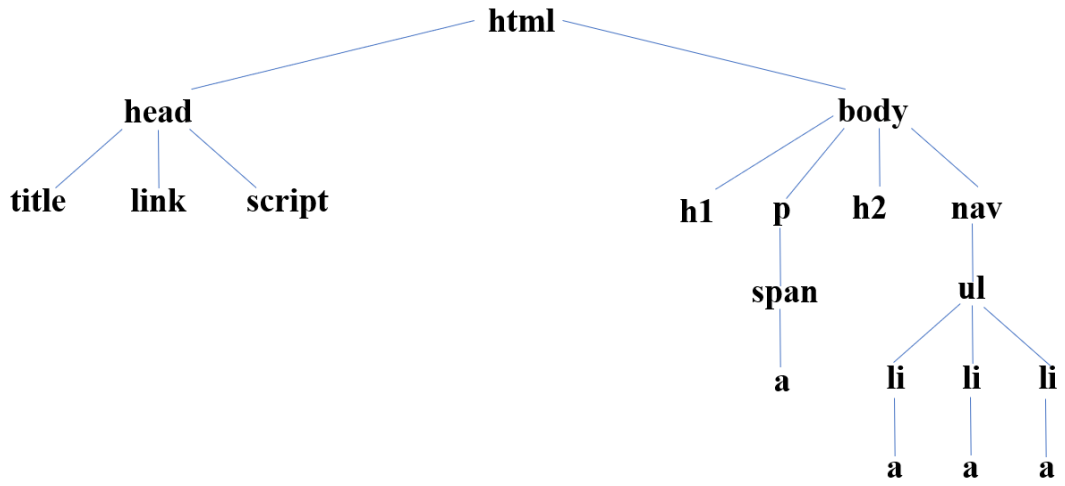

Рисунок 2.2 – Структура елементів документа HTML

Як вже було зазначено вище, створення привабливої сторінки буде неможливим без опанування CSS. Отже, для стилізації та форматування вебсторінок буде використовуватися CSS. CSS визначає вигляд документа, написаного мовою розмітки HTML. Тобто перетворює базову структуру HTML у візуально привабливий вебсайт, надаючи елементам дизайн та стилі.

CSS може змінювати шрифт, колір, розмір та інтервали контенту, розбивати його на кілька стовпців або додавати анімацію та інші функції. Його використання обумовлене можливістю забезпеченням стилів усіх вебсторінок застосунку [18].

2.1.2 Технології серверної частини

Бекенд – це серверна частина вебзастосунку. Вона забезпечує зберігання та організацію даних, а також відслідковує, чи все на клієнтській стороні вебсайту працює належним чином. Іншими словами, це частина застосунку, яку клієнт не бачить та з якою безпосередньо не взаємодіє. Програмні компоненти, розроблені програмістами-бекендерами, доступні користувачам через клієнтську програму [19].

В контексті розробки проєкту було обрано мови програмування, які в змозі забезпечити реалізацію поставлених питань, а саме – Node.js та Python.

Node.js – це відкрите серверне середовище, що використовує JavaScript. Застосунок на Node.js працює в межах одного процесу, без створення нового потоку для кожного запиту. Node.js включає асинхронні примітиви вводу-виводу як частину своєї стандартної бібліотеки, що запобігає блокуванню JavaScript-коду, та, в цілому, бібліотеки в Node.js розробляються з використанням неблокуючих парадигм. Це забезпечує, що блокуюча поведінка є винятком, а не правилом.

Node.js відомо завдяки використанню JavaScript на всьому стеку технологій, асинхронній моделі програмування для обробки кількох запитів одночасно, швидкому виконанню завдяки движку V8, великій та активній підтримці спільноти, масштабованості для реальних застосунків, сумісності з різними платформами та його ролі в забезпеченні повностекової розробки. Усі ці функції роблять Node.js дуже швидким та популярним у сфері розробки.

Ключові особливості Node.js:

– JavaScript «everywhere»: Node.js дозволяє розробникам використовувати JavaScript на всьому стеку: від фронтенду до бекенду. Це спрощує розробку та опанування необхідними технологіями;

– асинхронна модель програмування: Node.js використовує подійно-орієнтовану, неблокуючу (асинхронну) модель введення-виведення. Це дозволяє обробляти кілька запитів одночасно без блокування виконання інших завдань. В результаті додатки Node.js є досить високореагуючими та ефективними;

– швидке виконання: Node.js використовує движок V8, розроблений Google, який компілює та виконує JavaScript з великою швидкістю. Ця перевага в продуктивності робить його відмінним вибором для створення вебзастосунків;

– масштабованість: Node.js є легким та масштабованим, що робить його відмінним вибором для розробників;

– сумісність з різними платформами: Node.js працює на Windows, Linux, Unix, macOS та інших платформах. Ця гнучкість дозволяє розробникам писати код один раз і розгортати його де завгодно [20].

Доречним є застосування в розробці мови програмування Python – досить відомої у сфері машинного навчання та технологій штучного інтелекту.

Python – це високорівнева, загальнопризначена та дуже популярна мова програмування. Python використовують у веброзробці, у галузі машинного навчання, а також в усіх новітніх технологіях сфери програмного забезпечення. Мова має чітку та зрозумілу синтаксичну структуру, що спрощує процес вивчення та створення інтелектуальних моделей без складних кодових конструкцій.

Найважливішою перевагою використання Python є її різноманітність бібліотек та фреймворків, спеціально розроблених для штучного інтелекту та машинного навчання.

Генеративні ШІ, розроблені за допомогою Python – це творчі моделі, які здатні створювати новий контент, який, як правило, охоплює зображення, текст, аудіо або ж інші різноманітні формати даних. Ця галузь ШІ призначена для створення нового контенту на основі вивчених патернів та структур [21].

Архітектури генерації тексту відносяться до спеціалізованих моделей або структур нейронних мереж, створених для відтворення нового текстового контенту.

Одна з моделей генерації тексту – GPT, що і буде використана у проєкті [22]. Саме цієї галузі застосування буде торкатися вебзастосунок, використовуючи модель генерації тексту GPT-2.

На рисунку 2.3 проілюстрований приклад інтеграції мовної моделі у програму на Python [23].

```
# pip install gpt2-client
from gpt2_client import GPT2Client

gpt2 = GPT2Client('117M')
gpt2.load_model()

text = gpt2.generate(interactive=True, n_samples=4, return_text=True)
```

Рисунок 2.3 – Приклад використання моделі GPT-2 на базі мови програмування Python

2.2 Розробка бізнес-правил вебзастосунку

У межах проєкту визначено ролі користувачів вебзастосунку: незареєстрований користувач, зареєстрований користувач та адміністратор.

Табличне представлення користувацьких ролей подане у таблиці 2.1.

Таблиця 2.1 – Перелік ролей користувачів застосунку

Ідентифікатор	**Дійова особа**
АСТ-1	Незареєстрований користувач
АСТ-2	Зареєстрований користувач
АСТ-3	Адміністратор

Незареєстрований користувач може відвідати головну сторінку застосунку та ознайомитися з призначенням та умовами використання. Також перейти до сторінки реєстрації та створити свій обліковий запис (рис. 2.4).

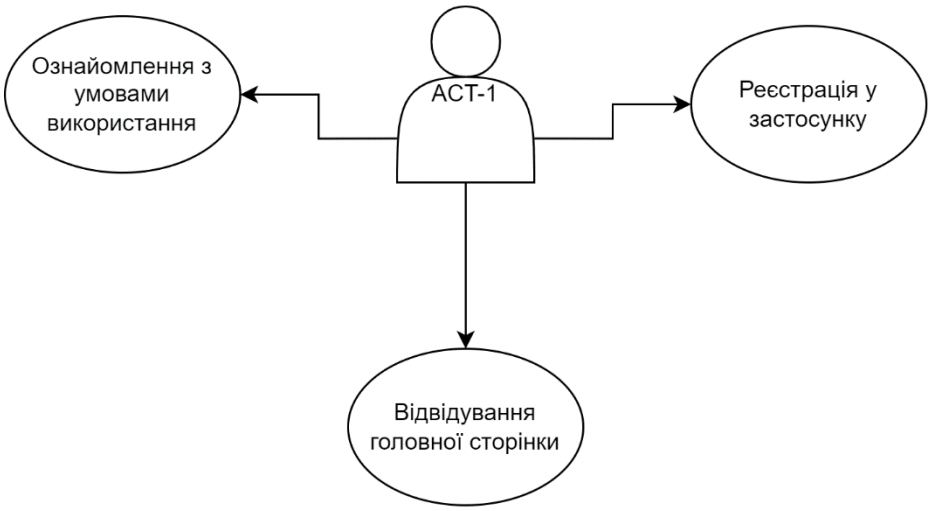

Рисунок 2.4 – Варіанти використання для ролі АСТ-1

Зареєстрований користувач має можливість увійти до свого кабінету. Потрапивши на власну сторінку, він може відвідувати сторінки генерації творчого контенту та збережених даних. Тобто, якщо користувачеві сподобається контент, який запропонував застосунок, він може зберегти його на свою сторінку зберігань та робити перегляд. Також, за потребою, може видаляти збережені ідеї (рис. 2.5).

Рисунок 2.5 – Варіанти використання для ролі АСТ-2

Адміністратор вебзастосунку має увійти до свого акаунту. Після входу він потрапляє на сторінку керування – може переглядати список усіх користувачів та їх збережений контент. Має змогу видаляти як користувачів, так і їх збережені ідеї (рис. 2.6).

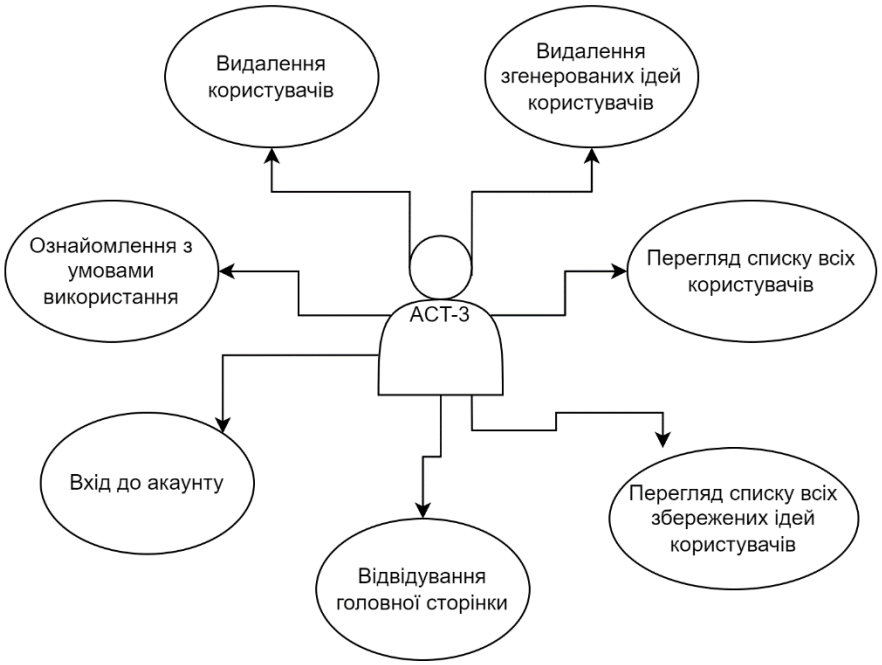

Рисунок 2.6 – Варіанти використання для ролі АСТ-3

2.3 Огляд та моделювання бази даних

2.3.1 MySQL як система управління БД

Для розробки бази даних вебзастосунку було обрано MySQL – систему керування реляційними БД для зберігання всієї інформації, необхідної для ефективного функціонування застосунку [24].

Вибір саме реляційної бази даних обумовлений забезпеченням точністю даних, цілісністю та безпекою збереження інформації, гнучкістю та зручністю у використанні.

Оскільки MySQL є широко використовуваною системою управління базами даних, вона підтримується різними мовами програмування, такими як Python, та JavaScript, тому її використання є доречним для створення проєкту.

2.3.2 Визначення необхідних відношень для БД

Вебзастосунок передбачає зберігання користувачів у базу даних. Тому стає необхідним створення таблиці «users», у якій будуть зберігатися дані зареєстрованих користувачів.

Для реєстрації у застосунку та авторизації знадобляться електронна адреса та пароль, отже, слід реалізувати відношення з відповідними полями.

Таблиця «users» буде містити наступні поля:

– id: унікальний ідентифікатор кожного користувача, який автоматично збільшується (AUTO_INCREMENT) при додаванні нового користувача. Використовується як первинний ключ (PRIMARY KEY), що гарантує унікальність записів;

– email: поле призначене для зберігання електронної адреси користувача. Його тип даних VARCHAR(255) вказує на те, що воно може містити рядок символів довжиною до 255 знаків. Поле обов’язкове для заповнення (NOT NULL), щоб кожен користувач мав унікальну адресу електронної пошти;

– password: поле призначене для зберігання пароля користувача. Так само, як і з полем «email», тип даних VARCHAR(255) вказує на те, що воно може містити рядок символів довжиною до 255 знаків. Пароль також є обов’язковим для заповнення (NOT NULL), щоб забезпечити безпеку даних користувачів.

Отже, на основі визначених полів буде створено таблицю «users», яка буде включати поля id, email та password.

Скрипт відношення представлений у лістингу 2.1, із реалізацією усіх необхідних полів.

Лістинг 2.1 Реалізація таблиці «users»:

```
CREATE TABLE IF NOT EXISTS users (
    id INT AUTO_INCREMENT PRIMARY KEY,
   email VARCHAR(255) NOT NULL,
    password VARCHAR(255) NOT NULL
);
```

Відомо, що зареєстровані користувачі матимуть змогу зберігати контент та переглядати його. Тому стає питання реалізації таблиці для зберігання згенерованих ідей.

Звісно, вона повинна включати зовнішній ключ, так як ідея прив’язана до користувача.

Буде створено відношення «savedIdeas», яке містить наступні поля:

– idIdea: унікальний ідентифікатор кожної збереженої ідеї, який автоматично збільшується (AUTO_INCREMENT) при додаванні нової ідеї. Це поле використовується як первинний ключ (PRIMARY KEY), що забезпечує унікальність записів;

– userId: зовнішній ключ (FOREIGN KEY), який посилається на поле «id» таблиці «users». Він вказує на користувача, який зберігає дану ідею. Кожна збережена ідея пов’язана з конкретним користувачем за допомогою цього поля;

– idea: поле призначене для зберігання текстового опису згенерованої ідеї. Тип даних TEXT використовується для зберігання великих обсягів тексту, що дозволяє користувачам зберігати ідеї будь-якої довжини. Також використовується обмеження CHECK, яке перевіряє, чи не перевищує довжина тексту 5000 символів.

Таким чином, буде створено таблицю «savedIdeas», яка буде мати поля idIdea, userId та idea.

Скрипт відношення представлений у лістингу 2.2.

Лістинг 2.2 Реалізація таблиці «savedIdeas»:

```
CREATE TABLE IF NOT EXISTS savedIdeas (
    idIdea INT AUTO_INCREMENT PRIMARY KEY,
    userId INT,
    idea TEXT CHECK (CHAR_LENGTH(idea) <= 5000),
    FOREIGN KEY (userId) REFERENCES users(id)
);
```

2.3.3 Моделювання БД

На підставі створених відношень буде створена БД вебзастосунку «generator_database» (рис. 2.7).

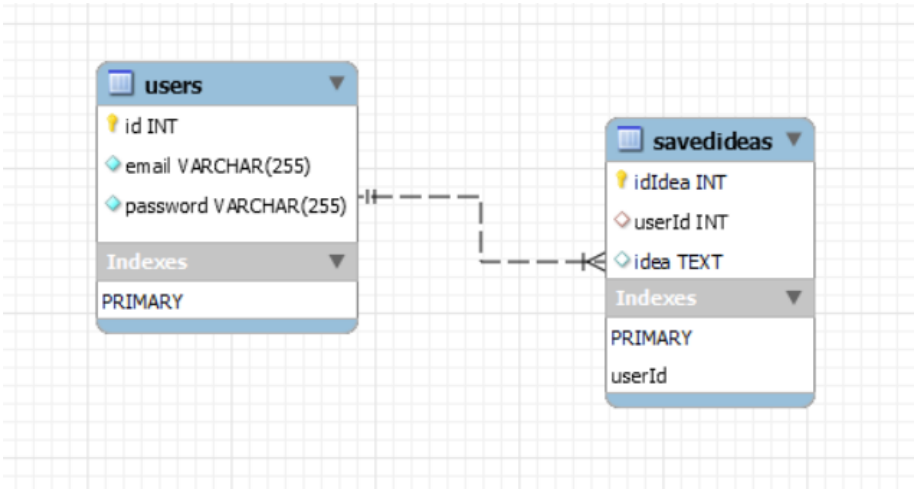

Рисунок 2.7 – Згенерована у MySQL Workbench модель БД

2.4 Обґрунтування вибору IDE

Важливим фактором є вибір середовища розробки програмного продукту, так як це визначає продуктивність та зручність у процесі написання, а також впливає на якість та швидкість створення вебзастосунку. Одним з найефективніших і популярних середовищ розробки для проєктів, що базуються на сучасних мовах програмування, є Visual Studio Code, або ж просто VS Code.

VS Code – гнучкий текстовий редактор, призначений для розробників. Він має обмежений набір функцій, однак він надає програмістам набір широких можливостей через розширення (рис. 2.8) [25].

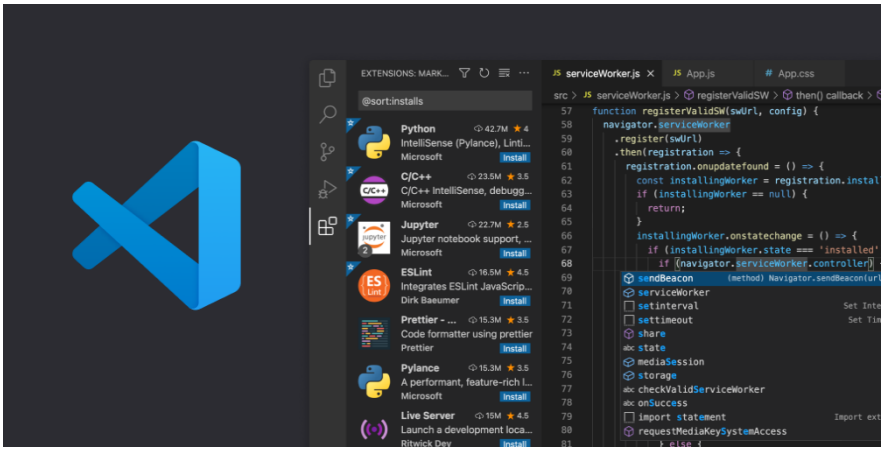

Рисунок 2.8 – Розширення у IDE VS Code

VS Code здобув популярність завдяки своєму привабливому інтерфейсу та вражаючим можливостям. Інструмент має мінімалістичний дизайн UI, що сприяє безперешкодному використанню для програмування та швидкому освоєнню.

Його переваги полягають в адаптивності, підтримці різних мов програмування через розширення. Тобто VS Code цілком підходить для проєкту, так як має вбудовану підтримку необхідних мов програмування [26].

2.5 Заходи щодо безпеки персональних даних у вебзастосунку

Проєкт передбачає реєстрацію користувачів. Як наслідок, слід забезпечити безпеку персональної інформації.

Для забезпечення безпеки персональної інформації у вебзастосунку було обрано метод автентифікації на основі токенів. Один з популярних механізмів використання токенів – це JSON Web Token. Використання токенів JWT дозволяє забезпечити безпеку та надійність передачі даних між клієнтом та сервером, а також ефективно контролювати доступ до ресурсів вебзастосунку [27].

JSON Web Token – це запропонований Інтернет-стандарт для створення даних з опційним підписом і/або опційним шифруванням, який містить JSON-об’єкт заяв, що підтверджує певну кількість претензій. Токени підписуються за допомогою приватного секретного ключа або ж публічного ключа.

Токени JWT генеруються сервером, підписуються секретним ключем і передаються клієнту. Клієнт використовує цей токен для підтвердження своєї автентичності під час подальшої взаємодії з вебзастосунком. Наприклад, сервер може згенерувати токен з претензією «увійшов як адміністратор» і передати його клієнту. Клієнт може використовувати цей токен, щоб довести, що він увійшов саме як адміністратор. Токени можуть підписуватися приватним ключем однієї сторони (зазвичай сервера), щоб будь-яка сторона пізніше могла перевірити, чи є токен законним. Якщо інша сторона, за допомогою відповідного та надійного засобу, володіє відповідним публічним ключем, вони також можуть перевірити законність токена. Токени призначені для компактності та безпеки , особливо в контексті одноразового входу в браузері. Твердження JWT зазвичай можуть використовуватися для передачі ідентифікації автентифікованих користувачів між постачальником ідентичності та постачальником послуг, або будь-якого іншого типу претензій [28].

JWT діє подібно до традиційної реалізації, проте він має деякі переваги – він самодостатній, всі дані, необхідні для автентифікації, можна зберігати в самому токені.

JWT складається з трьох основних частин: заголовка (header) (рис. 2.9), навантаження (payload) (рис. 2.10) та підпису (signature) (рис. 2.11) [29].

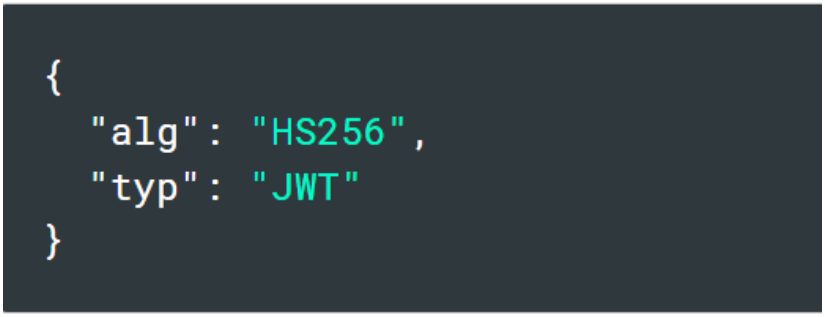

Рисунок 2.9 – Приклад заголовка

Рисунок 2.10 – Приклад навантаження

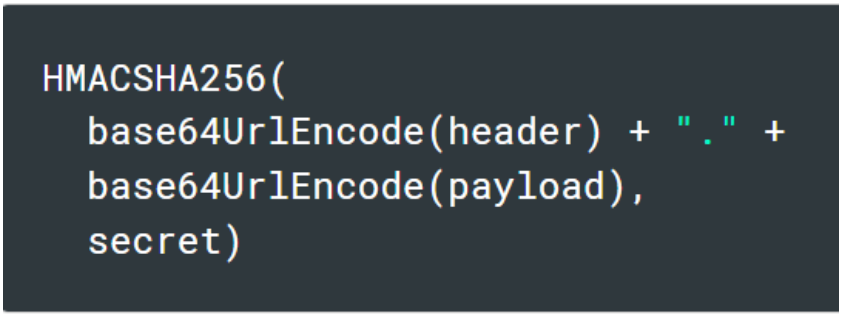

Рисунок 2.11 – Приклад підпису

Заголовок та навантаження формуються окремо, а потім на їхній основі обчислюється підпис. Зазвичай заголовок складається із двох полів: типу токена (у даному випадку JWT) та алгоритму хешування підпису.

Навантаження – це будь-які дані, які ви хочете передати у токені. Проте стандарт передбачає кілька зарезервованих полів:

- iss (issuer) – видавець токена;
- sub (subject) – «тема», призначення токена;
- aud (audience) – аудиторія, одержувачі токена;
- exp (expire time) – термін дії токена;
- nbf (not before) – термін, до якого токен не дійсний;
- iat (issued at) – час створення токена;
- jti (JWT id) – ідентифікатор токена.

Усі ці поля не є обов’язковими, але їх використання не за призначенням може призвести до колізій.

Підпис обчислюється на основі заголовка та навантаження. Таким чином, якщо хтось намагатиметься змінити дані у токені, він не зможе змінити підпис, не знаючи приватний ключ. При автентифікації приватним ключем може виступати пароль користувача (або хеш від пароля) [30].

Після першого входу у систему, сервер генерує JWT і повертає його клієнту. При кожному наступному запиті, клієнт повинен передавати JWT відповідним способом, установленим API (наприклад, через заголовок або як параметр запиту). Сервер розкодовує заголовок та навантаження і перевіряє зарезервовані поля. Якщо все вірно, за вказаним у заголовку алгоритмом формується підпис. І якщо отриманий підпис збігається з переданим, користувача авторизовано.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ГЕНЕРАТОРА ТА ДЕМОНСТРАЦІЯ РЕЗУЛЬТАТУ

3.1 Програмна реалізація

У рамках кваліфікаційної роботи був розроблений вебзастосунок генерації творчих ідей.

Програмна реалізація заснована на мовах розробки JavaScript та Python. Фільтрація та генерація даних для персонажів та вигаданих локацій створена на JavaScript, у той час як сюжет на основі цього контенту на Python з використанням бібліотеки Transformers для роботи моделі GPT-2, а також бібліотеки PyTorch, заснованої на Torch для реалізації нейронних мереж.

Розглянемо програмний код генерації персонажа на прикладі генерації фентезі імені. Його подано у лістингу 3.1.

Лістинг 3.1 Реалізація функції генерації фентезі імені персонажа:

```
function generateNameFromTemplate(startsWith, gender) {
  const vowels = 'aeiou';
  const consonants = 'bcdfghjklmnpqrstvwxyz';
  const bannedWords = ['...']; // List hidden to show code due to censorship
  function generateVowel() {
    return vowels.charAt(Math.floor(Math.random() * vowels.length));
  }
  function generateConsonant() {
    return consonants.charAt(Math.floor(Math.random() * consonants.length));
  }
  function hasThreeInRow(name) {
    const regex = /[aeiou]{3,}|[^aeiou]{3,}/i;
    return regex.test(name);
```

```
}
function containsBannedWords(name) {
 const lowerCaseName = name.toLowerCase();
 return bannedWords.some(word => lowerCaseName.includes(word));
}
function generateNameTemplate(template, startsWithLetter) {
 let name = startsWithLetter;
 let remainingLength = getRandomInt(3, 6) - name.length;
 let lastCharType = template[0];
 for (let i = 1; i < template.length && remainingLength > 0; i++) {
  let charToAdd = '';
  if (template[i].toLowerCase() === 'v') {
   charToAdd = generateVowel();
  } else if (template[i].toLowerCase() === 'c') {
   charToAdd = generateConsonant();
  } else {
   charToAdd = Math.random() < 0.5 ? generateVowel() : generateConsonant();
  }
  if (hasThreeInRow(name + charToAdd)) {
   if (template[i - 1].toLowerCase() === 'v') {
    charToAdd = generateConsonant();
   } else {
    charToAdd = generateVowel();
   }
  }
  name += charToAdd;
  remainingLength--;
 }
 return name;
```

```
  }
  function regenerateName(template, startsWithLetter) {
    let name = generateNameTemplate(template, startsWithLetter);
    while (containsBannedWords(name)) {
      name = generateNameTemplate(template, startsWithLetter);
    }
    return name;
  }
  const fantasyTemplates = {
    male: {
      3: ['cvc'],
      4: ['cvcv', 'cvcvc', 'vcvc', 'vcvcc', 'vccv', 'cvcc', 'cvvc'],
// the same to previous names for 5 and 6
    },
    female: {
      3: ['vcv', 'cvv'],
      4: ['vccv', 'cvcv', 'vcvc', 'vcvv'],
// the same to male names for 5 and 6
    },
    universal: {
      // the same to male and female names
    }
  };
let fantasyTemplatesForGender;
if (gender.toLowerCase() === 'male') {
  fantasyTemplatesForGender = fantasyTemplates.male;
} else if (gender.toLowerCase() === 'female') {
  fantasyTemplatesForGender = fantasyTemplates.female;
} else {
  fantasyTemplatesForGender = fantasyTemplates.universal;
```

```
    }
    const nameLength = getRandomInt(3, 6);
    const randomTemplate = getRandomElement(fantasyTemplatesForGender[nameLength]);
     let startsWithLetter = startsWith;
     if (startsWith.toLowerCase() === 'random') {
      startsWithLetter = Math.random() < 0.5 ? generateVowel() : generateConsonant();
     }
      const name = regenerateName(randomTemplate, startsWithLetter);
     return name;
    }
```

Функція generateNameFromTemplate(startsWith, gender) приймає запит користувача про першу літеру імені у змінній startsWith та стать для імені – записану у змінну gender.

Для генерації вигаданого ім'я спочатку було визначено набори голосних голосних (vowels), приголосних (consonants) літер та списку заборонених слів (bannedWords). Це обов'язково з точки зору цензури, так як слід фільтрувати випадкові небажані збігання при комбінації голосних та приголосних літер.

Функції generateVowel() та generateConsonant() генерують випадкові голосні та приголосні літери відповідно для подальшого застосуванні у шаблоні імен. Функція hasThreeInRow(name) перевіряє, чи містить ім'я три голосні чи приголосні літери поспіль. Вона потрібна з точки зору заходів щодо милозвучності генерованого імені. containsBannedWords(name) перевіряє, чи має ім'я заборонені слова.

Реалізація генерації вигаданого імені заснована на використанні шаблонів імен. Функція generateNameTemplate(template, startsWithLetter) створює ім'я на основі заданого шаблону та початкової літери імені. Виходить так, що ім'я формується шляхом вибору випадкових букв відповідно до

шаблону, при цьому перевіряється наявність трьох голосних або приголосних поспіль. Якщо таке сталося, то літери замінюються, щоб уникнути цієї ситуації [31].

Шаблони поділені на категорії за обраною користувачем статтю. Тобто в залежності від вибраної статі та випадкової довжини імені (від 3 до 6 символів), вибирається відповідний набір шаблонів [32].

Потім випадково вибирається один із шаблонів для генерації імені. Якщо початкова літера не задана, вона генерується випадковим чином.

Отримане ім’я проходить певну перевірку. Якщо воно містить заборонені слова, то перегенерується, доки не буде згенеровано допустиме ім’я. Таким чином, в результаті функція повертає перевірене згенероване фентезі ім’я персонажа [33-35].

Інші функції генерації даних персонажа та локацій мають подібну реалізацію. Втім, сюжет створюється на Python з використанням бібліотеки Transformers для роботи моделі GPT-2. Розглянемо реалізацію створення сюжетних ідей. Його подано у лістингу 3.2.

Лістинг 3.2 Реалізація генерації сюжету за допомогою GPT-2:

```
import random
import torch
from transformers import GPT2LMHeadModel, GPT2Tokenizer
class TextGenerator:
    def __init__(self, model_name="gpt2"):
        self.tokenizer = GPT2Tokenizer.from_pretrained(model_name)
        self.model = GPT2LMHeadModel.from_pretrained(model_name)
    def generate_story_text(self, character_data, locations_data, max_length=300):
        character_prompt = self.generate_character_prompt(character_data)
        locations_prompt = self.generate_locations_prompt(locations_data)
```

```
        prompt = f"{character_prompt} {locations_prompt} Go ahead and make interesting history. Use data about main character: "
        generated_text = self.generate_text_from_prompt(prompt, max_length=max_length)
        generated_text = generated_text.replace("Go ahead and make interesting history. Use data about main character:", "").strip()
        return generated_text
    def generate_text_from_prompt(self, prompt, max_length=300):
        input_ids = self.tokenizer.encode(prompt, return_tensors="pt", max_length=512, truncation=True)
        attention_mask = torch.ones_like(input_ids)
        output = self.model.generate(input_ids, attention_mask=attention_mask, max_length=max_length, num_return_sequences=1, temperature=0.3, repetition_penalty=1.2, pad_token_id=self.tokenizer.eos_token_id)
        generated_text = self.tokenizer.decode(output[0], skip_special_tokens=True)
        return generated_text
    def generate_character_prompt(self, character_data):
        character_name = character_data.get('characterName', 'Unknown')
        age = character_data.get('age', 'Unknown')
        magical_ability = character_data.get('magic', 'Unknown')
        prompt = f"The story revolves around {age}-year-old {character_name}. "
        if magical_ability != 'Unknown':
            prompt += f"{character_name} possesses {magical_ability} magical ability. "
        return prompt
    def generate_locations_prompt(self, locations_data):
        country_name = locations_data.get('countryName', 'Unknown')
```

country_feature = locations_data.get('countryFeature', 'Unknown')

prompt = f"The story takes place in the fantasy country of {country_name}. This country is famous for {country_feature}."

return prompt

Для використання мовної моделі був реалізований клас TextGenerator, призначений для генерації сюжету, виходячи із даних про персонажа та локації.

У методі __init__ прописана ініціалізація об'єкта класу. У ньому завантажуються попередньо навчені токенізатор та модель GPT-2 з використанням імені моделі.

Генерація тексту проходить наступним чином: метод generate_story_text приймає дані про персонажа (character_data) та локації (locations_data) у виді словників, а також максимальну довжину створюваного тексту, записану у змінну max_length.

У цьому методі спочатку викликаються методи generate_character_prompt та generate_locations_prompt для генерації промптів про персонажа та локації відповідно. Вони беруться зі вже створених даних та використовуються для логічного поєднання контенту [36-38].

Далі ці промпти поєднуються в один текст для генерації, до якого додається ключова фраза, що призначена для роз'яснення задачі для моделі GPT-2. Вона пояснює, що саме слід зробити із вхідними даними. Тобто, використати їх у генерації тексту сюжету.

Після цього викликається метод generate_text_from_prompt, який генерує текст сюжету на основі створеного промпту. Повертає результат.

Ключова фраза, що входила до тексту, видаляється, так як її не має бачити користувач. Згенерований текст повертається користувачу.

Метод generate_text_from_prompt приймає промпт та максимальну довжину тексту, які потім використовуються для генерації тексту за допомогою моделі GPT-2.

Промпт токенізується та подається на вхід моделі для генерації тексту сюжету.

Отриманий згенерований текст декодується з числової послідовності в текст природною мовою і повертається.

Метод generate_character_prompt генерує промпт персонажа на основі даних про персонажа, які, як вже було зазначено, вже визначені. Аналогічно із методом generate_locations_prompt – відтворює локації відповідно. У цих методах формуються фрази про персонажа та локацію згідно з даними, які потім об’єднуються та повертаються [39].

Таким чином реалізується генерація тексту сюжету на основі даних про персонажа і локацій за допомогою моделі GPT-2.

Отже, штучний інтелект використовується посередництвом мови Python через використання бібліотеки Transformer.

3.2 Інструкція користувача

Користувач отримує посилання на генератор. При переході за посиланням на вебзастосунок користувач потрапляє на головну сторінку (рис. 3.1).

Рисунок 3.1 – Головна сторінка вебзастосунку генерації творчих ідей

У верхньому лівому кутку сторінки знаходиться значок «Меню». Якщо користувач натисне на нього лівою кнопкою миші, він побачить лістинг сторінок вебзастосунку (рис. 3.2).

Рисунок 3.2 – Меню вебзастосунку

Можна побачити, що меню передбачає навігацію між головною сторінкою «Home», інформаційним блоком «How to use?», сторінками реєстрації «Registration» та авторизації «Sign in».

Якщо користувач натисне «How to use?» та перейде за цим посиланням, або ж закриє сторінку меню та проскролить вниз, знаходячись на головній, він побачить короткий опис щодо користування вебзастосунком (рис. 3.3).

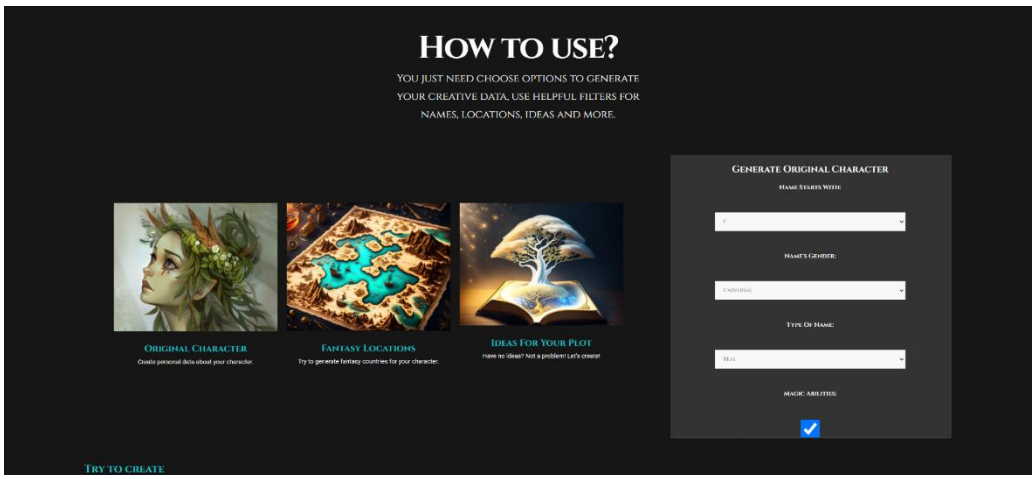

Рисунок 3.3 – Інформаційний блок «How to use?» головної сторінки застосунку

У цьому блоці стисло та зрозуміло продемонстровано функціонал вебзастосунку.

Перша картинка зліва показує, що користувач матиме 3 категорії генерації ідей, а картинка, що справа, – приклад форми з фільтрацією генерованих даних. Так буде виглядати фільтр майбутніх даних, коли користувач отримує доступ до створення контенту.

Якщо користувач проскролить нижче, він побачить дві кнопки – «Create your account» та «Log in», відповідно, – посилання на сторінки реєстрації (якщо користувач ще не має акаунту) та автентифікації (якщо вже зареєстрований).

Аналогічним чином він може перейти за цими посиланнями у меню, якщо ж ні, то скориставшись відповідними кнопками на головній сторінці (рис. 3.4).

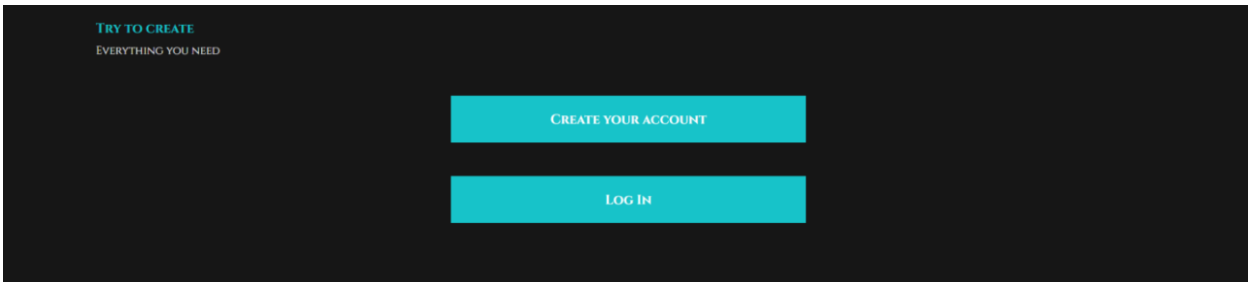

Рисунок 3.4 – Кнопки з посиланнями на сторінки реєстрації та автентифікації

Наприклад, користувач ще не зареєстрований. Він переходить за посиланням на сторінку реєстрації та бачить відповідну форму. Зліва на сторінці користувач може ознайомитися з умовами реєстрації – він має ввести свою електронну адресу та створити пароль.

Пароль має певні обмеження щодо валідності даних. Він повинен складатися не менше, ніж з 6, та не більше, ніж з 15 символів. Має містити латинські літери у верхньому та нижньому регістрах та налічувати хоча б одну цифру (рис. 3.5).

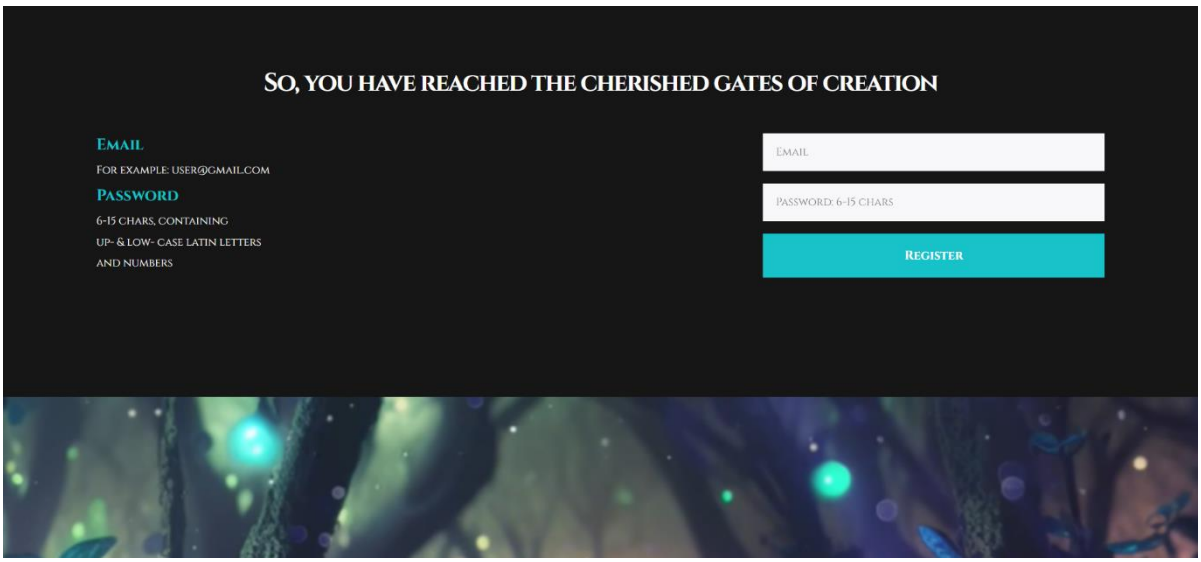

Рисунок 3.5 – Сторінка реєстрації користувача

Наприклад, якщо користувач знехтує правилами реєстрації, його не буде занесено до БД користувачів, та він побачить повідомлення про помилку під час реєстрації (рис. 3.6).

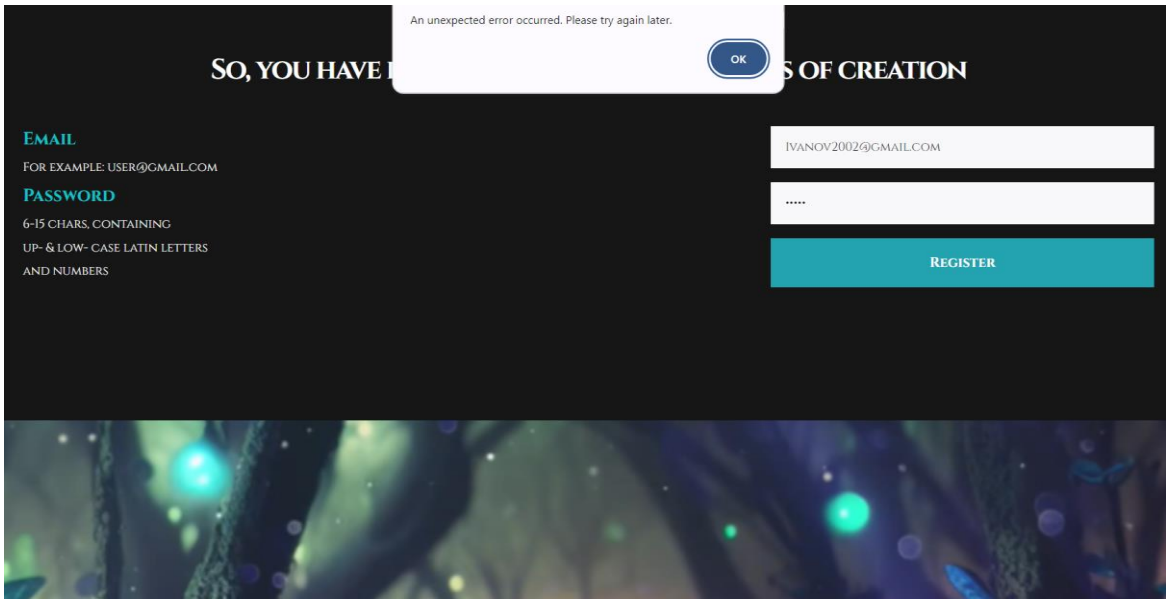

Рисунок 3.6 – Повідомлення про помилку при реєстрації користувача. У даному випадку було введено замалу кількість символів у поле паролю.

Аналогічну форму має сторінка автентифікації користувача (рис. 3.7).

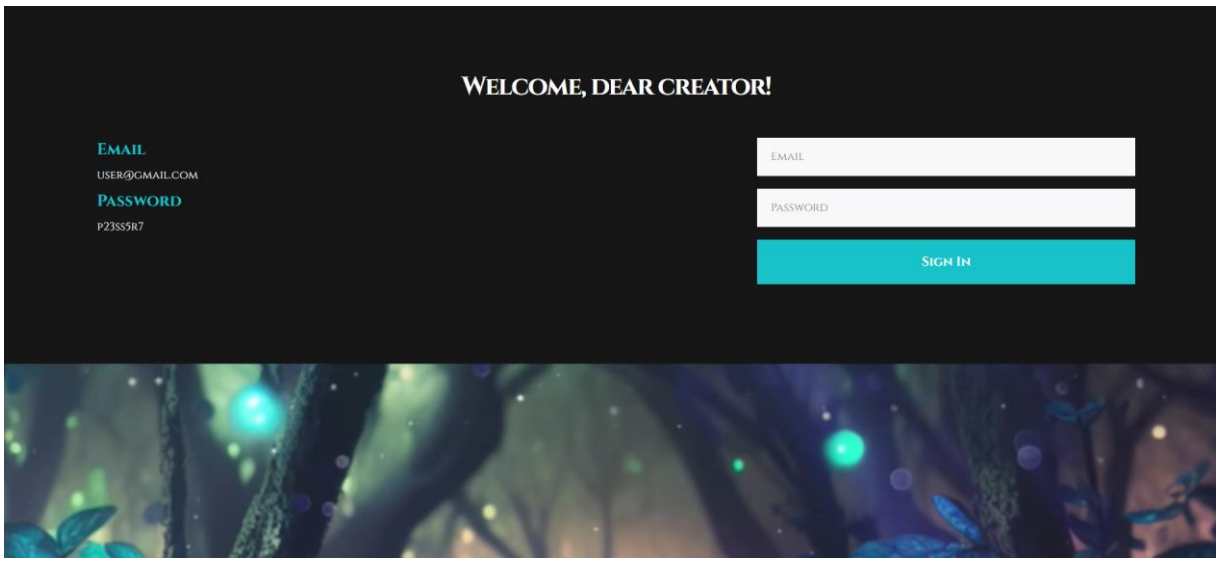

Рисунок 3.7 – Сторінка автентифікації користувача

У випадку, якщо користувач має акаунт та ввів вірні персональні дані, або ж успішно зареєструвався, при натисканні кнопки під формою введення даних він потрапить на персональну сторінку (рис. 3.8).

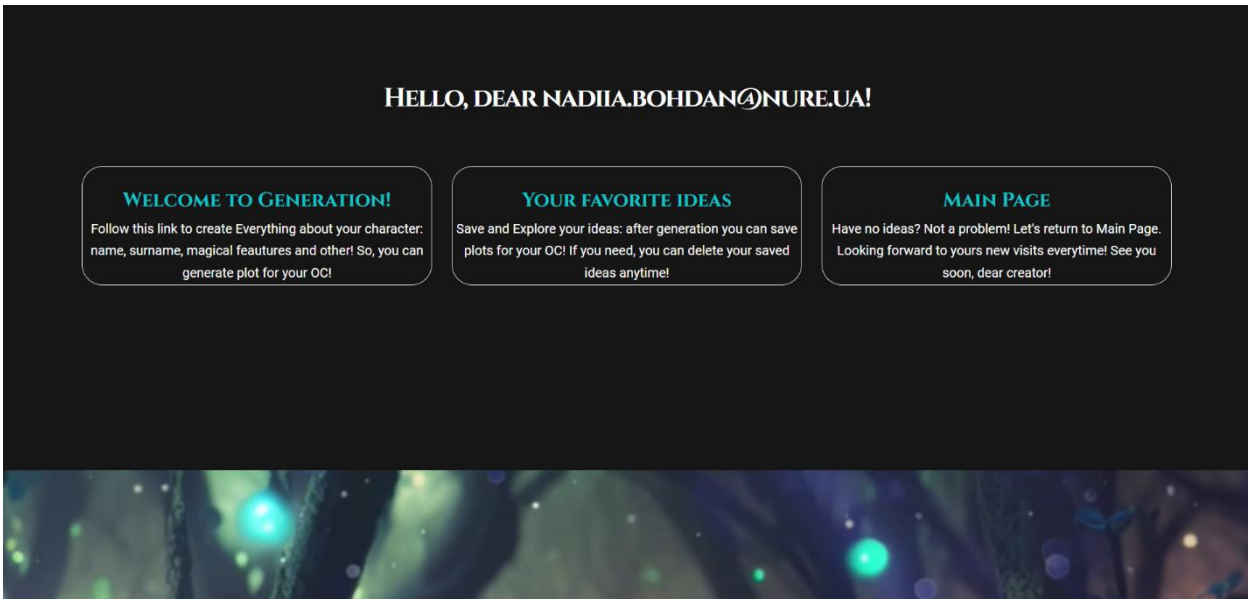

Рисунок 3.8 – Сторінка користувача

Користувач побачить привітання зі зверненням до нього відповідно до персональної електронної адреси у системі генератора та навігацію з відповідною інформацією про зміст сторінок, які може відвідати.

Користувач має доступ до сторінок генерації контенту «Welcome to generation!», своїх збережених ідей «Your favorite ideas» та може повернутися на головну сторінку застосунку «Main page». Опис контенту цих сторінок знаходиться у відповідних їм блоках.

Якщо користувач натисне лівою кнопкою миші на посилання до генерації «Welcome to generation!», то він потрапить на сторінку створення творчого контенту.

Знаходячись на сторінці генерації, можна побачити 3 категорії генерації ідей: створення даних про оригінального персонажа «Original character», вигаданих локацій «Fantasy locations» та оригінальний сюжет на основі попередньої інформації «Ideas for your plot».

Кожна картинка відповідає за навігацію на потрібний блок генерації (рис. 3.9).

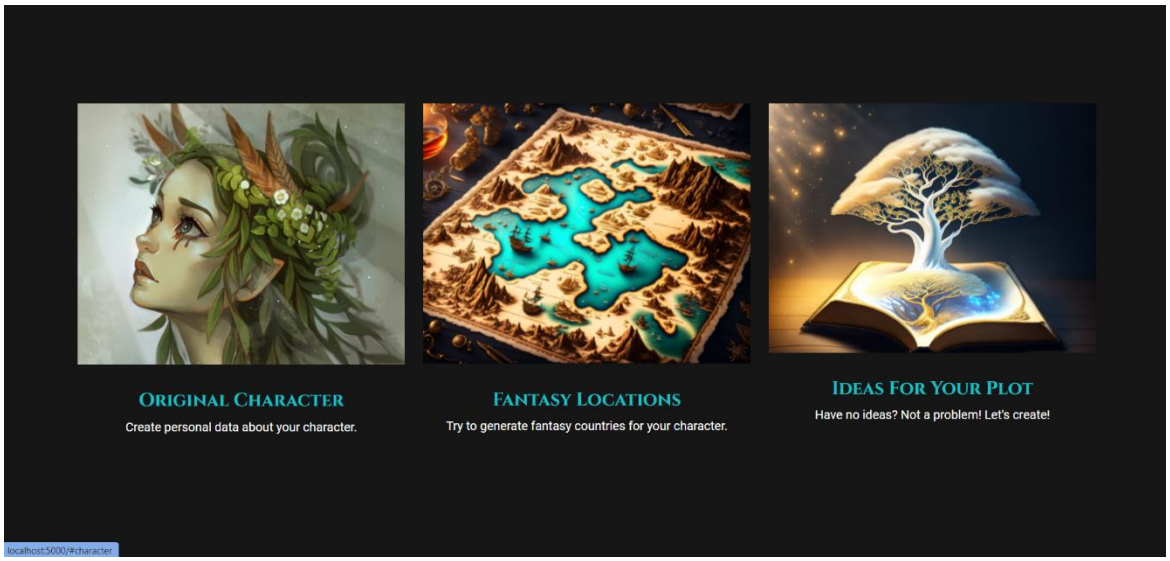

Рисунок 3.9 – Сторінка генерації творчого контенту

Якщо користувач натисне на картинку генерації персонажа (першу зліва картинку на сторінці), він перейде до блоку створення даних про оригінального персонажа. Цей блок передбачає дві форми: форму фільтрації запиту користувача зліва та форму з майбутнім контентом справа (рис. 3.10).

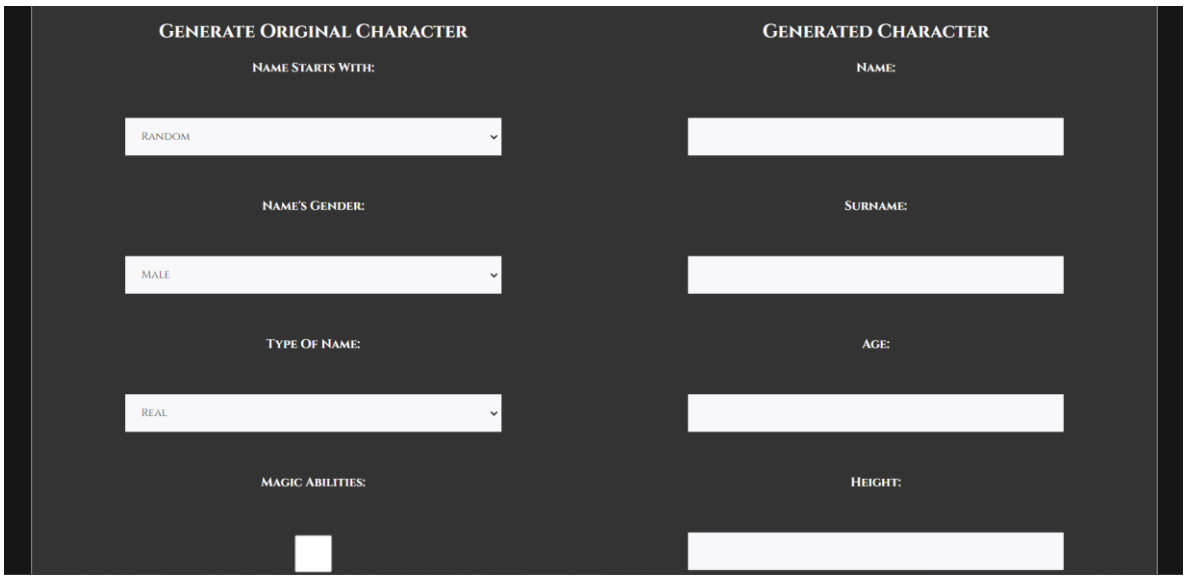

Рисунок 3.10 – Блок генерації оригінального персонажа (форма згенерованих даних продемонстрована не повністю)

Розглянемо фільтрацію за першою літерою імені (рис. 3.11).

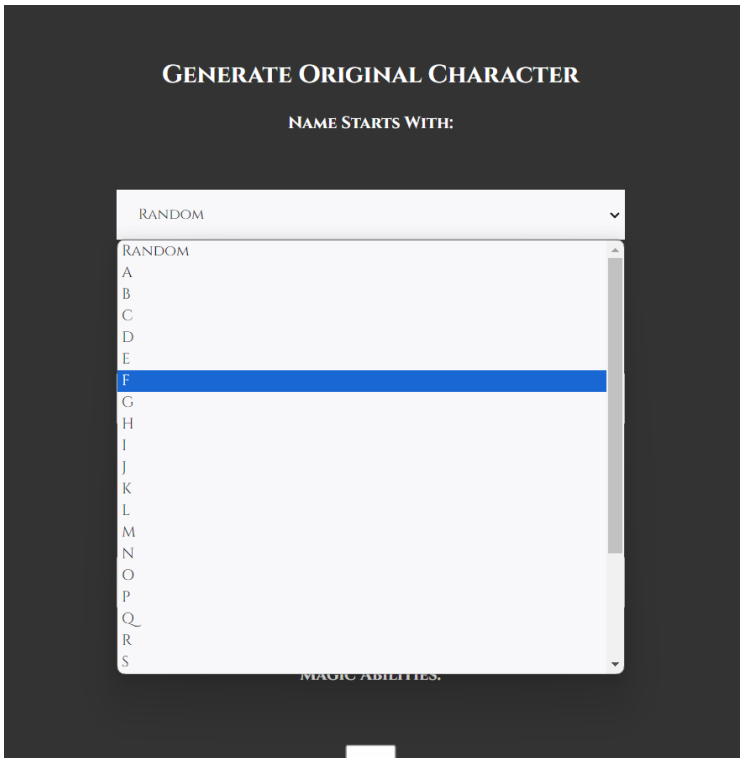

Рисунок 3.11 – Фільтр по першій літері імені

Користувач має змогу обрати, з якої літери буде починатися ім’я персонажа.

Фільтрація передбачає:

– усі літери англійської абетки;

– варіант «Random»: тобто випадкову літеру, та ім’я згенерується, починаючись з будь-якої букви англійської абетки, яка обереться автоматичним чином програмою генератора.

Також користувач може відфільтрувати запит за статтю – у переліку є чоловічі, жіночі та універсальні імена (рис. 3.12).

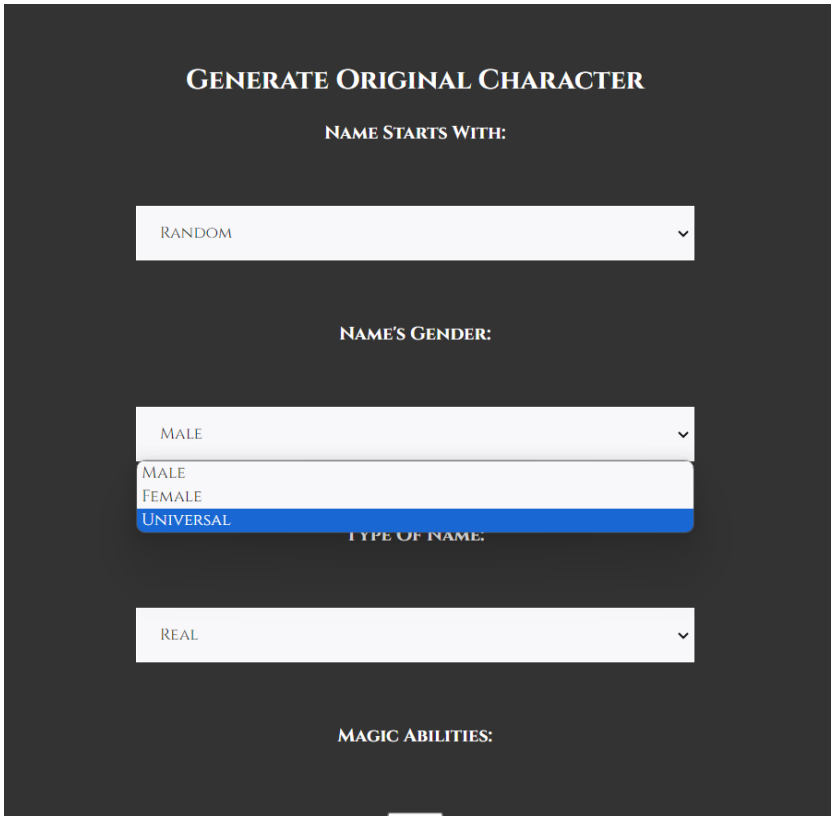

Рисунок 3.12 – Фільтр по статі імені

Після фільтрації по першій літері та статі користувач бачить ще один фільтр – по типу імені.

Види імен мають такі варіанти:

– реальне: імена, які існують у реальному світі;

– фентезі: вигадані імена.

Фільтр по типу імені продемонстрований на рисунку 3.13.

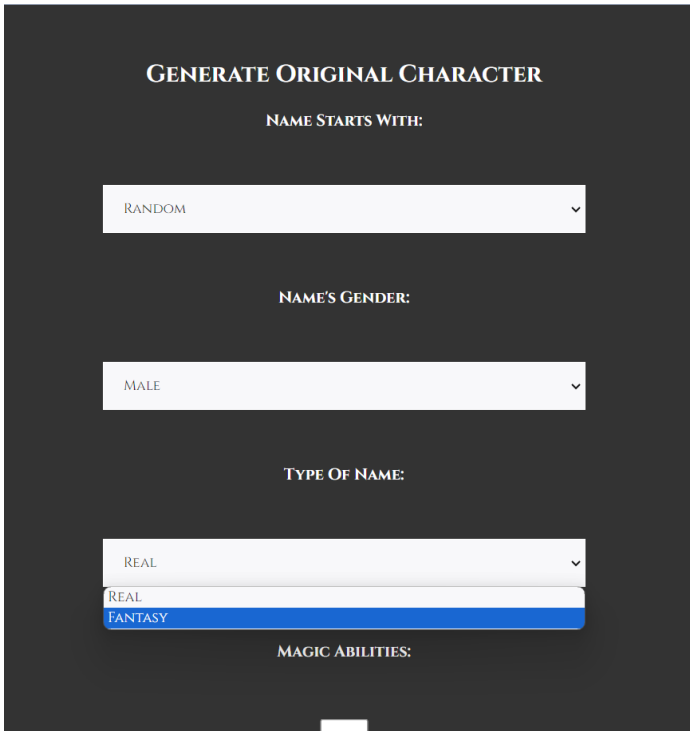

Рисунок 3.13 – Фільтр по типу імені

Перед тим, як натиснути кнопку генерувати «Generate character», користувач може відфільтрувати результат за наявністю магічних здібностей у персонажа (рис. 3.14).

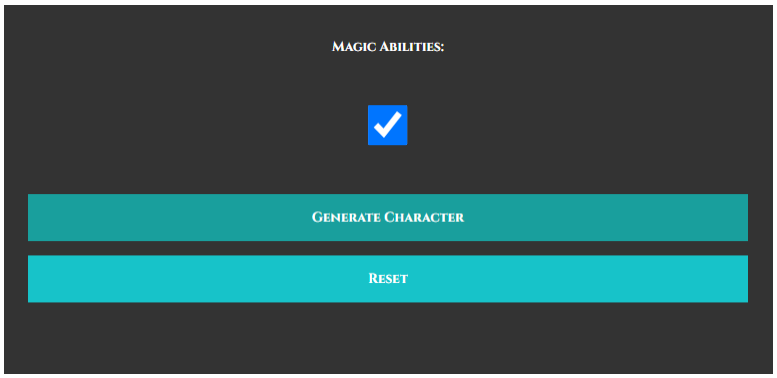

Рисунок 3.14 – Фільтр по наявності магії у персонажа та кнопки генерації та скидання даних, якщо не сподобається майбутній результат

Після відправлення відфільтрованого запиту на генерацію користувач побачить у правій частині екрану автоматично заповнену форму з результатом.

Результат містить ім’я, прізвище, вік, опис зовнішності (зріст, колір волосся та очей) та магічні здібності персонажа, якщо їх було обрано користувачем (рис. 3.15).

Рисунок 3.15 – Результат генерації персонажа

Після успішної генерації даних про персонажа користувач може проскролити нижче та побачити блок із генерацією вигаданих локацій (рис. 3.16).

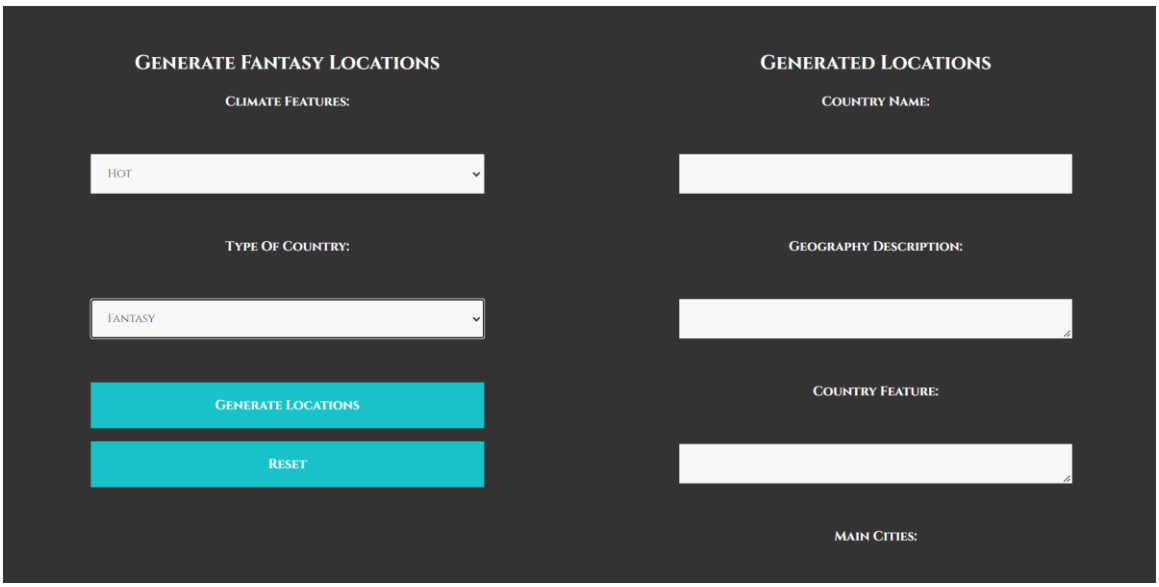

Рисунок 3.16 – Блок із генерацією вигаданих локацій – форми фільтрації та результату

Користувач може відфільтрувати запит за типом клімату вигаданої країни та за особливостями – схожа на реальні або ж фантастична (рис. 3.17 – 3.18).

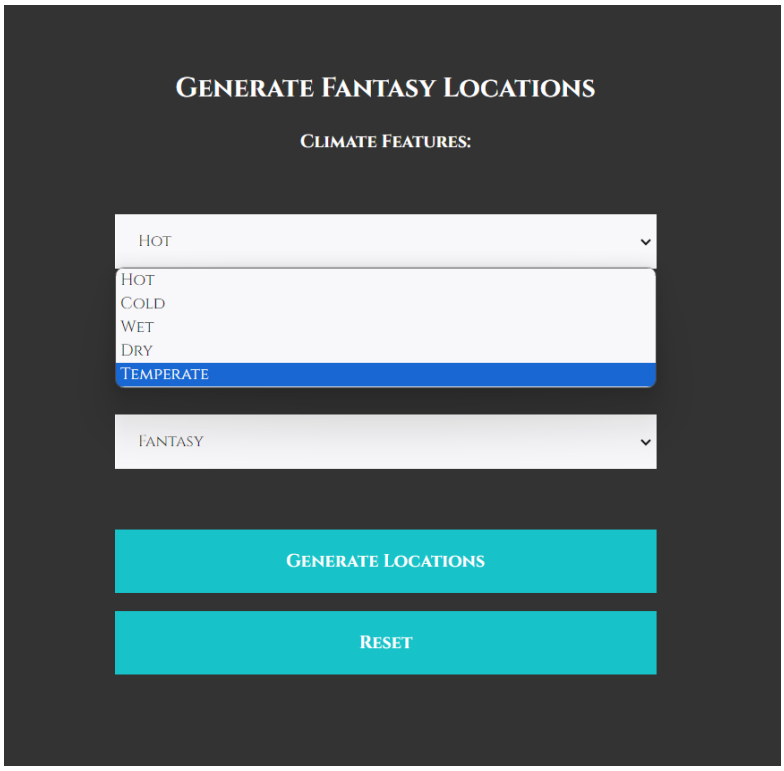

Рисунок 3.17 – Фільтр по типу клімату локації

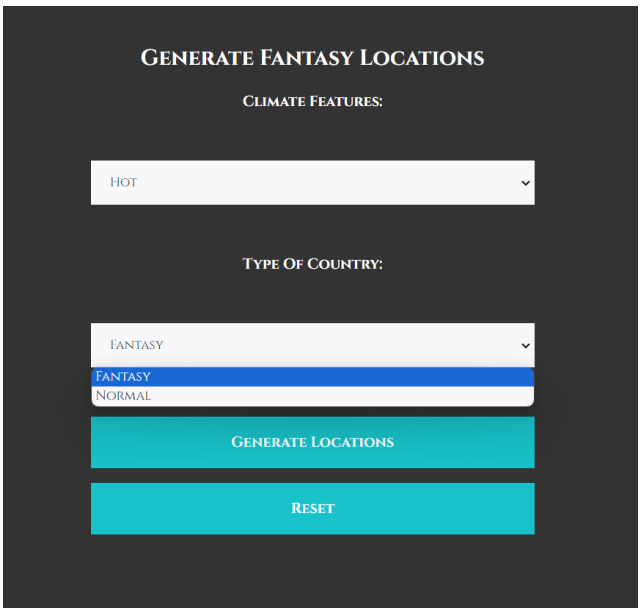

Рисунок 3.18 – Фільтр по типу вигаданої локації

Кнопки генерації та скидання даних мають функціонал, аналогічний попередньому блоку генерації. Форма із результатом також автоматично заповнюється після відправлення відфільтрованого запиту.

Наступний блок генерації – створення сюжету. Він має попередження користувачу про використання мовної моделі GPT-2, отже усі згенеровані нею ідеї залежать від її можливостей. Після натискання кнопки генерувати історію «Generate story» користувач через декілька секунд отримує створений моделлю сюжет (рис. 3.19).

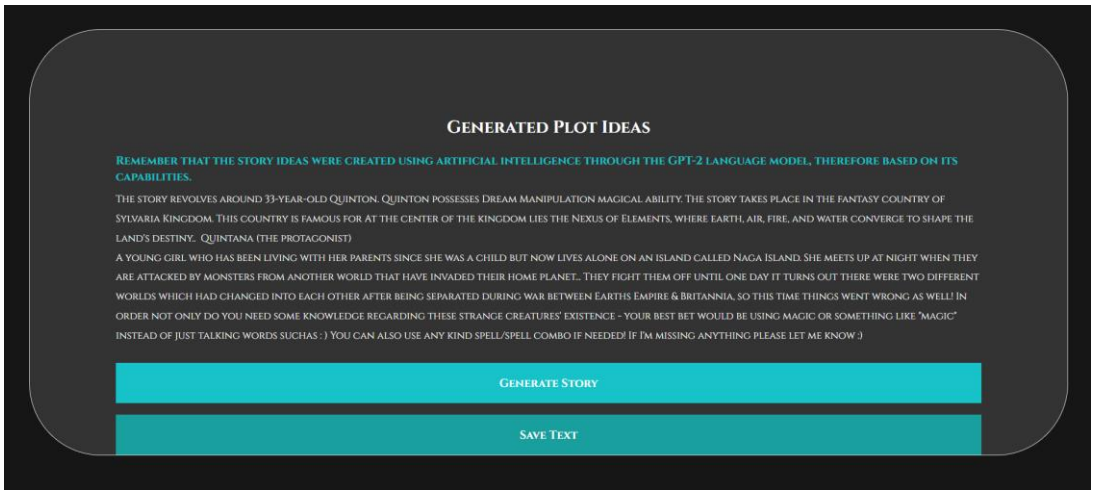

Рисунок 3.19 – Згенерований сюжет

Якщо користувачу сподобається створена ідея, він має змогу зберегти її на сторінку своїх ідей. Для цього йому слід натиснути кнопку збереження «Save text» внизу блоку генерації сюжету. Ідея буде успішно збережена.

У цьому легко переконатися. Потрібно перейти до сторінки користувача та в навігації обрати сторінку збережених ідей «Your favorite ideas», перейти за цим посиланням. На цій сторінці користувач побачить усі свої збережені сюжети (рис. 3.20).

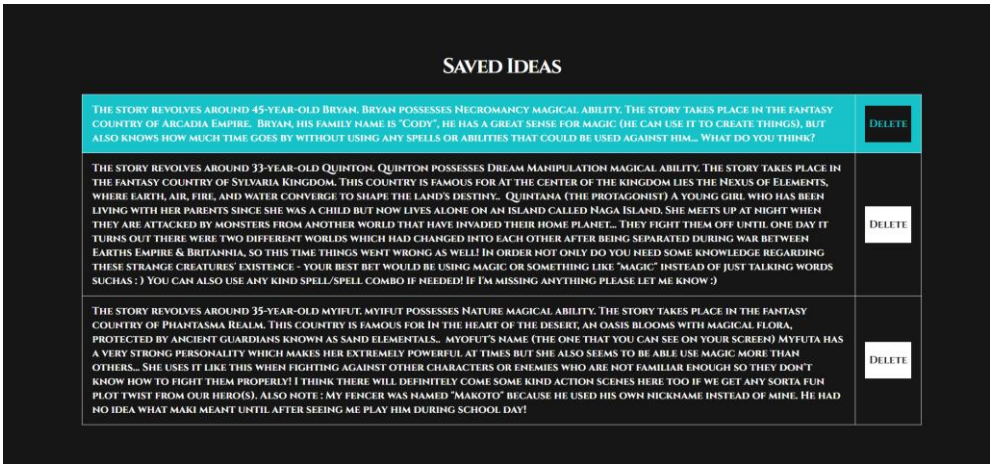

Рисунок 3.20 – Сторінка збережених ідей користувача

Користувач може переглядати ідеї, а також видаляти їх. Для цього він може натиснути на кнопку видалення «Delete» у другому стовпці таблиці збережених ідей відповідного рядку. Ідею буде успішно видалено.

Однак застосунок передбачає, окрім зареєстрованого та незареєстрованого користувачів, роль адміністратора. Слід розглянути можливості, якими володіє саме адміністратор вебзастосунку.

Адміністратор має увійти до свого акаунту під своєю поштою та паролем. Система відслідковує, чи належать ці дані саме йому, так як при авторизації адміністратор потрапляє на свою власну сторінку.

Сторінка адміністратора включає деякі можливості керування БД застосунку. Вона містить таблиці зареєстрованих користувачів та їх збережених ідей (рис. 3.21).

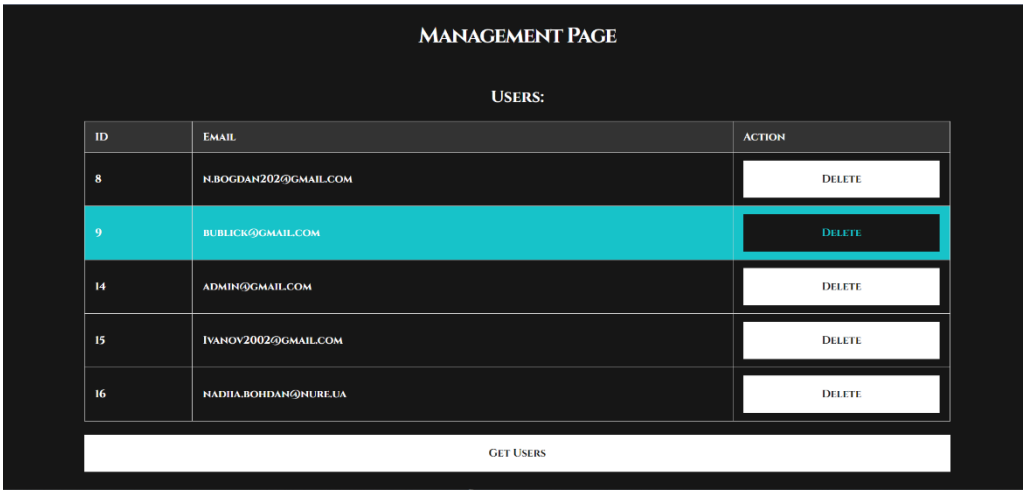

Рисунок 3.21 – Сторінка адміністратора. Зареєстровані користувачі

Адміністратор має змогу переглядати список користувачів та видаляти їх, натискаючи на кнопку видалення «Delete» для відповідного рядку. Однак, слід зазначити, що адміністратор не має змоги видаляти самого себе. Якщо він це зробить, то нічого не станеться, та з’явиться повідомлення про помилку (рис. 3.22).

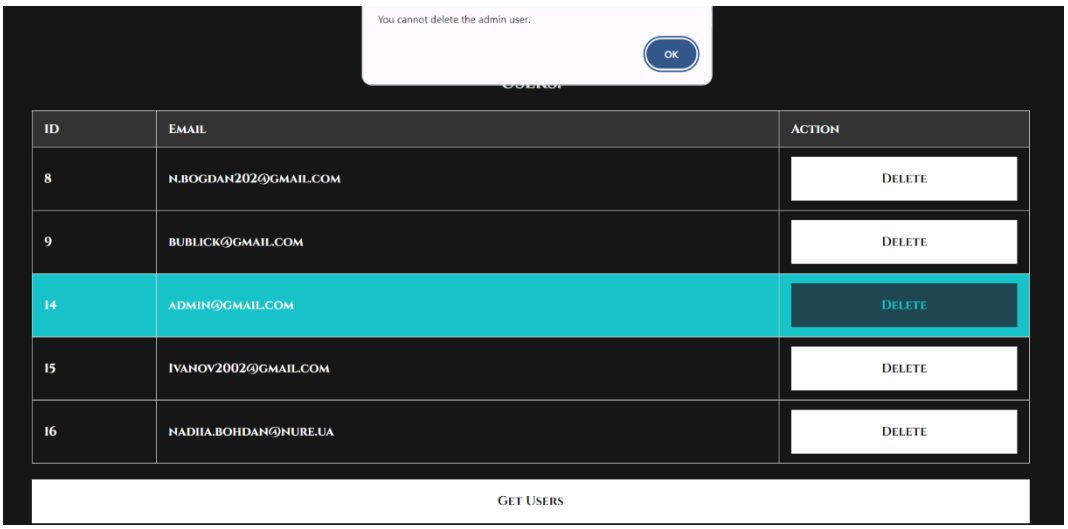

Рисунок 3.22 – Сторінка адміністратора. Повідомлення про помилку

Якщо адміністратор проскролить нижче, він побачить таблицю зі збереженими ідеями користувачів. Він також може видаляти ідеї, аналогічно із процедурою видалення користувачів (рис. 3.23).

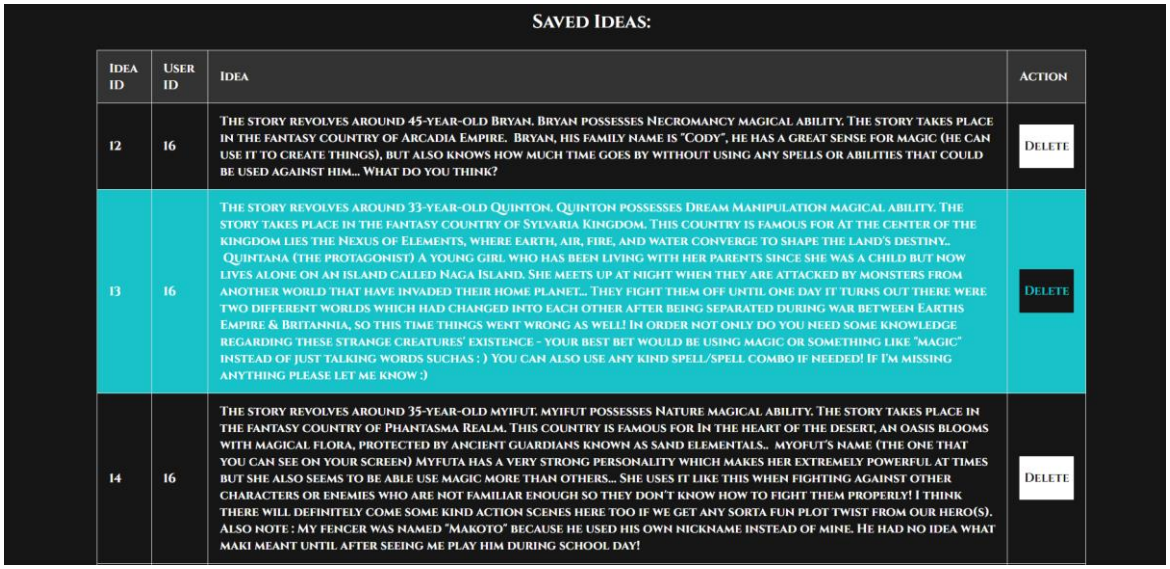

Рисунок 3.23 – Сторінка адміністратора. Повідомлення про помилку

Слід зазначити, що вебзастосунок має адаптацію для мобільних пристроїв. На рисунках 3.24 – 3.26 продемонстровано, як виглядає UI генератора на таких екранах.

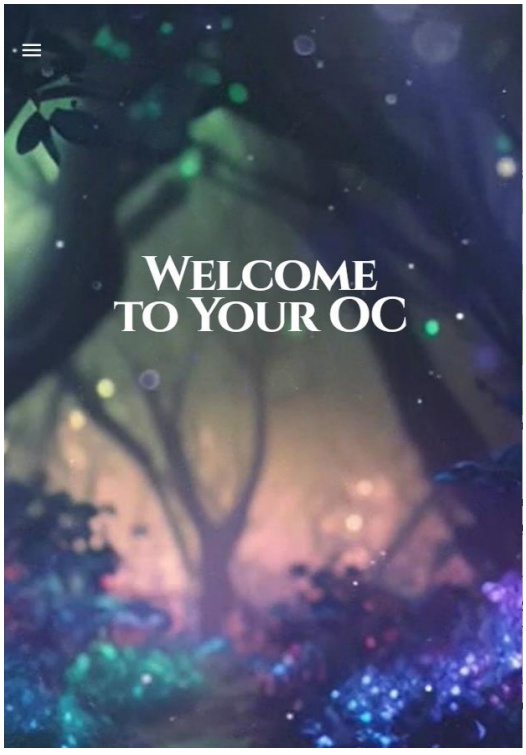

Рисунок 3.24 – Головна сторінка вебзастосунку. Вигляд з роздільної здатності мобільного пристрою

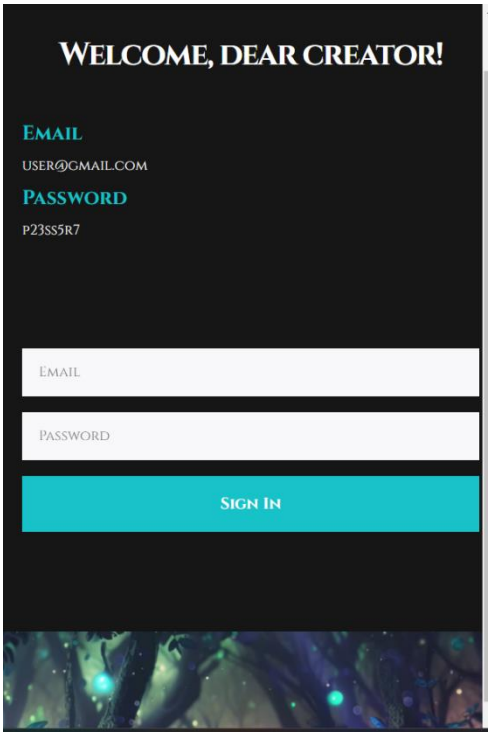

Рисунок 3.25 – Сторінка авторизації. Вигляд з роздільної здатності мобільного пристрою

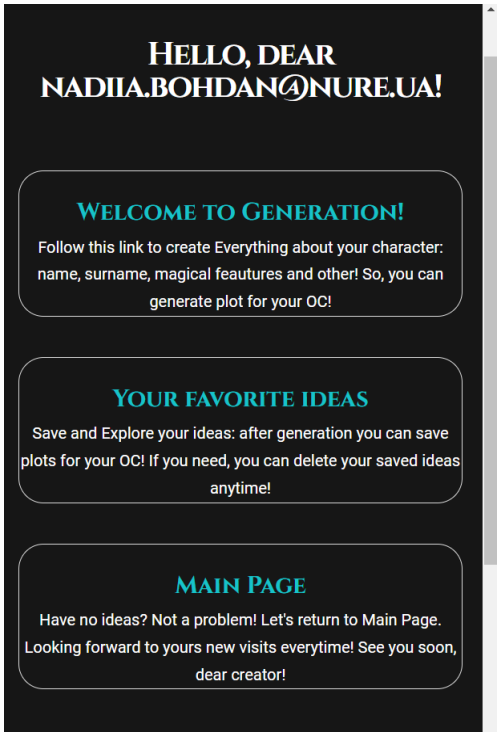

Рисунок 3.26 – Сторінка користувача. Вигляд з роздільної здатності мобільного пристрою

3.3 Заходи щодо поліпшення вебзастосунку

Вебзастосунок генерації творчих ідей має великий потенціал до удосконалення.

Наприклад, у подальшому є сенс поліпшити такі аспекти проєкту:

– технології генерації контенту: впровадити більш спрямовані на штучний інтелект технології генерації, використання розширених мовних моделей при створенні ідей;

– творчий контент, яким забезпечує застосунок: розширити можливості генерації, додати нові більш детальні фільтри для запиту. Наприклад, більш конкретний опис персонажа, можливість створювати декількох пов’язаних між собою героїв за одну ітерацію тощо;

– можливості користувача: додати до творчого поля додаткових дій для більшої участі у процесі створення. Наприклад, функцію редагування створеного контенту: щоб користувач за бажанням міг втілити власні ідеї у згенерований контент. Також можна створити власні критерії генерації, які міг би надавати користувач генератору для покращення бажаного результату;

– розширене використання мовних моделей: впровадити можливість вибору мовної моделі для текстового контенту. Також, окрім текстової генерації, додати функціонал генерації зображень на основі результатів створених ідей для візуалізації, що може взагалі відкрити нові кордони для творчої генерації. Це може бути як візуалізація зовнішності персонажа, так і локацій та незвичайних місць, що забезпечить творче натхнення користувачам та зробить генерацію більш цікавою;

– створення інтеграції із соціальними мережами: надати змогу користувачам ділитися своїми ідеями у популярних соціальних мережах та месенджерах;

– створення функціоналу оберненого зв’язку: додати до застосунку можливість користувачам залишати коментарі. Наприклад, можливість

надавати оцінку певному функціоналу вебзастосунку (якість генерації, різноманітність результатів, логічність контенту);

– реалізація удосконалення сторінки збережених даних: розширити контент збережених ідей. Наприклад, надати змогу зберігати різні категорії контенту, а також створити зручний інтерфейс для цих даних на сторінці. Також надати змогу редагувати збережені ідеї незалежно від категорії;

– створення локалізації: впровадити можливість перекладу інтерфейсу вебзастосунку на мови міжнародного значення. Наприклад, на головній сторінці користувач матиме змогу обрати мову, якою хотів би взаємодіяти із генератором.

Отже, вебзастосунок можна покращити у майбутньому.

ВИСНОВКИ

У рамках кваліфікаційної роботи був розроблений і реалізований вебзастосунок генерації творчих ідей.

Під час розробки проєкту було детально ознайомлено з існуючими програмними засобами генерації творчих ідей, виявлено їхні недоліки та переваги. Проведено їхній аналіз задля розуміння проблематики поставленої задачі.

Розроблено концепцію створення вебзастосунку та опановано необхідні технології розробки. На їх основі створено логіку генерації творчого контенту.

Реалізовано клієнтську частину застосунку згідно із вимогами до користувацького інтерфейсу з використанням JavaScript, HTML та CSS. Розроблено серверну частину мовами програмування Python та Node.js. використано мовну модель GPT-2 посередництвом Python.

Створено БД застосунку згідно з належними умовами.

Розгорнуто робочу програму вебзастосунку. Виявлено варіанти подальшого удосконалення застосунку.

Результатом проєкту є функціональний інструмент, який може бути використаний для стимулювання творчості та генерації нових ідей у різних галузях творчої діяльності.

Результати роботи апробовано у вигляді тез доповідей під час XXVIII Міжнародного молодіжного форуму «РАДІОЕЛЕКТРОНІКА І МОЛОДЬ У XXI СТОЛІТТІ», онлайн конференції «КОМП’ЮТЕРНИЙ ЗІР, СИСТЕМНИЙ АНАЛІЗ ТА МАТЕМАТИЧНЕ МОДЕЛЮВАННЯ» [40].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Build a Streamlit Web App for Text Generation with GPT-2 Models. URL: https://www.toolify.ai/ai-news/build-a-streamlit-web-app-for-text-generation-with-gpt2-models-580622 (дата звернення 17.01.2024).

2. Generating Text with GPT2 in Under 10 Lines of Code. URL: https://medium.com/@majd.farah08/generating-text-with-gpt2-in-under-10-lines-of-code-5725a38ea685 (дата звернення 15.04.2023).

3. Better language models and their implications. URL: https://openai.com/research/better-language-models (дата звернення 14.02.2019).

4. Гороховатський, В. О., Передрій, О. О., Творошенко, І. С., & Марков, Т. Є. (2023). Матриця відстаней для множини компонентів структурного опису як інструмент для створення класифікатора зображень.

5. Web Applications Types – Examples of Web Apps + Benefits and Challenges. URL: https://www.intelivita.com/blog/types-of-web-applications/ (дата звернення 27.12.2023).

6. Tvoroshenko, I. S., & Tabashnyk, V. A. (2018). Development of a spatial model of geoinformation support for people with disabilities in wheelchairs in Kharkiv. Collection of scientific works of KhNUPS, 1(55), 122-128.

7. Ahmad, M. A., Tvoroshenko, I., Baker, J. H., & Lyashenko, V. (2019). Modeling the structure of intellectual means of decision-making using a systemoriented nfo approach.

8. Lyashenko, V., Mustafa, S. K., Tvoroshenko, I., & Ahmad, M. A. (2020). Methods of using fuzzy interval logic during processing of space states of complex biophysical objects.

9. Tvoroshenko, I., Ahmad, M. A., Mustafa, S. K., Lyashenko, V., & Alharbi, A. R. (2020). Modification of models intensive development ontologies by fuzzy logic.

10. Гороховатський, В. О., & Творошенко, І. С. (2021). Методи інтелектуального аналізу та оброблення даних: навч. посібник.

11. Plot Generator. URL: https://www.plot-generator.org.uk/ (дата звернення 01.01.2024).

12. Fantasy Name Generators. URL: https://www.fantasynamegenerators.com/ (дата звернення 01.01.2024).

13. Character Name Generator. URL: https://writerbuddy.ai/name-generator/character-name-generator (дата звернення 01.01.2024).

14. Character Name Generator. URL: https://easy-peasy.ai/templates/character-name-generator (дата звернення 01.01.2023).

15. Тітов, С. В., & Тітова, О. В. (2015). Оцінка юзабіліті освітніх сайтів: методи і технології. Вісник Харківської державної академії культури. Серія: Соціальні комунікації, (47), 127-134.

16. JavaScript Tutorial. URL: https://www.geeksforgeeks.org/javascript/?ref=shm (дата звернення 15.04.2024)

17. HTML Tutorial. URL: https://www.geeksforgeeks.org/html-tutorial/?ref=shm (дата звернення 23.03.2024).

18. CSS Tutorial. URL: https://www.geeksforgeeks.org/css-tutorial/?ref=shm (дата звернення 15.04.2024).

19. Frontend vs Backend. URL: https://www.geeksforgeeks.org/frontend-vs-backend/ (дата звернення 18.04.2023).

20. Node.js Tutorial. URL: https://www.geeksforgeeks.org/nodejs/ (дата звернення 17.04.2024).

21. AI With Python Tutorial. URL: https://www.geeksforgeeks.org/python-ai/?ref=shm#ai-with-python-generative-ai (дата звернення 02.02.2024).

22. Yakovleva, O., Nebeský, L., & Kirichenko, A. (2023). Using the GPT models for responses based on custom content to develop neural consultant for university applicants. In V International Scientific and Practical Conference. Madrid, Spain (pp. 172-178).

23. Gpt2-client. URL: https://pypi.org/project/gpt2-client/ (дата звернення 13.11.2019).

24. Gorokhovatskyi V., Tvoroshenko I., Yakovleva O., Hudáková M., and Gorokhovatskyi O. (2024) Application a committee of Kohonen neural networks to training of image classifier based on description of descriptors set, *IEEE Access*, vol. 12, pp. 73376-73385.

25. Visual Studio Code. URL: https://code.visualstudio.com/ (дата звернення 01.01.2024).

26. Visual Studio vs Visual Studio Code: What's the Key Difference? URL: https://distantjob.com/blog/visual-studio-vs-visual-studio-code/ (дата звернення 19.03.2024).

27. Руденко, Д. О., & Бондар, В. О. (2020, November). ОГЛЯД МОЖЛИВОСТЕЙ ВИКОРИСТАННЯ СТРАТЕГІЙ ОБ'ЄКТНО-ОРІЄНТОВАНОГО МАППІНГУ ДЛЯ ЗІСТАВЛЕННЯ СУТНОСТЕЙ ПРИ РОЗРОБЦІ WEB ДОДАТКІВ. In The 3 rd International scientific and practical conference—Priority directions of science and technology development‖(November 22-24, 2020) SPC—Sci-conf. com. ua‖, Kyiv, Ukraine. 2020. 1488 p. (p. 377).

28. JSON Web Token. URL: https://en.wikipedia.org/wiki/JSON_Web_Token (дата звернення 27.02.2024)

29. Introduction to JSON Web Tokens. URL: https://jwt.io/introduction (дата звернення 01.01.2024).

30. Андрєєва, А. Ю., & Руденко, Д. О. (2022, October). ЯК БЛОКЧЕЙН ТОКЕНІЗАЦІЯ ЗМІНЮЄ СВІТ. In The 2 nd International scientific and practical conference "Science and innovation of modern world"(October 26-28, 2022) Cognum Publishing House, London, United Kingdom. 2022. 948 p. (p. 233).

31. Шафроненко, А. Ю., Бодянський, Є. В., & Руденко, Д. О. (2023). Модифікований рекурентний метод достовірної нечіткої кластеризації з використанням оптимізаційної процедури на основі косяків риб. Системи обробки інформації, (1 (172)), 92-96.

32. Бодянський, Є. В., Шафроненко, А. Ю., & Климова, І. М. (2021). Метод адаптивної достовірної нечіткої кластеризації даних на основі

еволюційного алгоритму. Збірник наукових праць Харківського національного університету Повітряних Сил, (2 (68)), 80-83.

33. Шафроненко, А. Ю., & Бодянський, Є. В. (2023). Нечітка достовірна кластеризація великих масивів даних з гіпереліпсоїдальними класами з довільною орієнтацією осей. Наука і техніка Повітряних Сил Збройних Сил України, (1 (50)), 93-99.

34. Бодянський, Є. В., Шафроненко, А. Ю., & Климова, І. М. (2020). Рекурентна достовірна нечітка кластеризація великих даних з використанням функції належності спеціального типу.

35. Bodyanskiy, Y., Pliss, I., & Shafronenko, A. (2022). Adaptive neurofuzzy clustering of distorted data based on prototype-centroid strategy using evolutionary procedures. Artificial Intelligence, 27, 239-244.

36. Zeleniy, O., Rudenko, D., Lyubchenko, V., & Lyashenko, V. (2022). Image Processing as an Analysis Tool in Medical Research.

37. Valeria, B., Hryhorii, K., Diana, R., & Vyacheslav, L. (2023). Phase Portrait Models as a Tool for Analyzing Banking Activities.

38. Lyashenko, V., & Rudenko, D. (2021). Modeling Deformation of Spur Gear.

39. Руденко Д.О., Танянський О.С. (2021) Принципи передобробки даних для машинного навчання. Матеріали IV Міжнародної науково-практичної дистанційної конференції "TOPICAL ISSUES OF MODERN SCIENCE, SOCIETY AND EDUCATION", Харків, 1-3 листопада 2021р. С.381-385.

40. Богдан Н. І. Розробка веб-додатка генерації творчих ідей. *Радіоелектроніка та молодь у XXI столітті: тези доповідей 28-го Міжнародного молодіжного форуму (Харків, 16–18 квітня 2024 р.). Харків: ХНУРЕ*, 2024. Т. 7. С. 17-18.