НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
(повна назва інституту/факультету)

Кафедра інформатики та програмної інженерії
(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2023 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення інформаційних систем»

спеціальності «121 Інженерія програмного забезпечення»

на тему: Веб-сервіс для автоматичної категоризації художніх творів

Виконав студент IV курсу, групи ІТ-92
(шифр групи)
Татарин Микола Сергійович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник ст. викладач Ковтунець О.В. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з графічної документації ст. викладач Вітковська І.І. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент доцент кафедри ІСТ, к.т.н., доц., Галушко Д.О. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ –2023

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 Інженерія програмного забезпечення

Освітньо-професійна програма – Інженерія програмного забезпечення інформаційних систем

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2023 р.

ЗАВДАННЯ
на дипломний проєкт студенту

Татарину Миколі Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема проєкту Веб-сервіс для автоматичної категоризації художніх творів
керівник проєкту Ковтунець О.В., ст. викладач
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «31» травня 2023 р. №2102-с

2. Термін подання студентом проєкту « 17 » червня 2023 року

3. Вихідні дані до проєкту: технічне завдання

4. Зміст пояснювальної записки
1) Аналіз вимог до програмного забезпечення.
2) Моделювання та конструювання програмного забезпечення.
3) Аналіз якості та тестування програмного забезпечення.
4) Впровадження та супровід програмного забезпечення.

5. Перелік графічного матеріалу

1) Схема структурна варіантів використань _____

2) Схема бази даних _____

3) Схема структурна компонентів програмного забезпечення _____

4) Креслення вигляду екранних форм _____

6. Консультанти розділів проєкту

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

7. Дата видачі завдання «10» березня 2023 року _____

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вивчення рекомендованої літератури	10.03.2023	
2	Аналіз існуючих методів розв’язання задачі	20.03.2023	
3	Постановка та формалізація задачі	23.03.2023	
4	Розробка інформаційного забезпечення	14.04.2023	
5	Алгоритмізація задачі	16.04.2023	
6	Обґрунтування вибору використаних технічних засобів	27.04.2023	
7	Розробка програмного забезпечення	10.05.2023	
8	Налагодження програми	24.05.2023	
9	Виконання графічних документів	01.06.2023	
10	Оформлення пояснювальної записки	02.06.2023	
11	Подання ДП на попередній захист	06.06.2023	
12	Подання ДП рецензенту	12.06.2023	
13	Подання ДП на основний захист	17.06.2023	

Студент _____ Микола ТАТАРИН

(підпис) (ініціали, прізвище)

Керівник _____ Олесь КОВТУНЕЦЬ

(підпис) (ініціали, прізвище)

АНОТАЦІЯ

Пояснювальна записка дипломного проєкту складається з чотирьох розділів, містить 57 таблиць, 23 рисунки та 11 джерел – загалом 68 сторінок.

Дипломний проєкт присвячений розробці веб-сервісу автоматичної категоризації художніх творів.

Мета: полегшити пошук художніх творів.

Об'єкт дослідження: веб-сервіс автоматичної категоризації художніх творів.

Предмет дослідження: методи автоматичної категоризації тексту.

У розділі першому досліджено предметну область, розглянуті технічні рішення і аналоги, розроблені вимоги.

Розділ другий присвячений моделюванню бізнес-процесів, створенню архітектури і конструюванню програмного забезпечення.

У третьому розділі розглядається аналіз якості програмного забезпечення та описуються процеси тестування.

У четвертому розділі розглядається розгортання і випуск програмного забезпечення.

КЛЮЧОВІ СЛОВА: ВЕБ-СЕРВІС, КЛАСИФІКАЦІЯ ТЕКСТУ, NLP, API, БАЗА ДАНИХ,

ABSTRACT

The explanatory note of the diploma project consists of four sections, contains 57 tables, 23 figures and 11 sources - a total of 68 pages.

The diploma project is dedicated to the development of a web service for automatic categorization of artistic works.

Purpose: to facilitate the search for works of art.

Object of research: web service of automatic categorization of artistic works.

Subject of research: methods of automatic text categorization.

In the first chapter, the subject area was investigated, technical solutions and analogues were considered, requirements were developed.

The second chapter is devoted to business process modeling, creation of architecture and software design.

The third chapter deals with software quality analysis and test processes.

The fourth chapter deals with software deployment and release.

KEYWORDS: WEB-SERVICE, TEXT CLASSIFICATION, NLP, API, DATABASE.

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4	КПІ.ІТ-9224.045440.00.90	Завдання на дипломний проєкт	6	
2	А4	КПІ.ІТ-9224.045440.01.91	Технічне завдання	12	
3	А4	КПІ.ІТ-9224.045440.02.81	Пояснювальна записка	68	
4	А4	КПІ.ІТ-9224.045440.03.12	Текст програми	9	
5	А4	КПІ.ІТ-9224.045440.04.51	Програма та методика тестування	6	
6	А4	КПІ.ІТ-9224.045440.05.34	Керівництво користувача	9	
7	А3	КПІ.ІТ-9224.045440.06.99	Графічний матеріал	4	

Змін.	Арк.	№ докум.	Підп.	Дата	КПІ.ІТ-9224.045440.00.90		
Розроб.		Татарин М.С.			Відомість дипломного проєкту	Літ.	Аркуш 1 / Аркушів
Перевір.		Ковтунець О.В.					
Н.контр.		Вітковська І.І.				КПІ ім. Ігоря Сікорського ФІОТ каф. ІПІ гр. ІТ-92	
Затв.		Жаріков Е.В.					

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

ВЕБ-СЕРВІС ДЛЯ АВТОМАТИЧНОЇ КАТЕГОРИЗАЦІЇ ХУДОЖНІХ ТВОРІВ

Технічне завдання

КПІ.ІТ-9224.045440.01.91

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олесь КОВТУНЕЦЬ

Нормоконтроль:
_____ Ірина ВІТКОВСЬКА

Виконавець:
_____ Микола ТАТАРИН

Київ – 2023

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Веб-сервіс для автоматичної категоризації художніх творів.

Галузь застосування:

Наведене технічне завдання поширюється на розробку програмного забезпечення Веб-сервіс для автоматичної категоризації художніх творів, котре використовується для автоматизації процесу класифікації художніх творів за жанражи та призначена для полегшення пошуку маловідомих художніх творів.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки веб-сервісу для автоматичної категоризації художніх творів є завдання на дипломне проєктування, затверджене кафедрою інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для автоматизації процесу класифікації художніх творів.

Метою розробки є полегшення пошуку художніх творів.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Програмне забезпечення повинно забезпечувати виконання наступних основних функції:

4.1.1 Користувацького інтерфейсу

– сторінка реєстрації;
– сторінка автентифікації;
– сторінка отримання токену для API;
– сторінка класифікації тексту;
– сторінка класифікації документу;
– сторінка перегляду результатів.

4.1.2 Програмного інтерфейсу

– endpoint для класифікації тексту;
– endpoint для класифікації документу;
– endpoint для отримання результатів класифікації;
– endpoint для видалення результату класифікації;

4.1.3 Для користувача:

– реєстрація користувача за допомогою адреси електронної пошти і пароля;
– перевірка електронної пошти на відповідність формату електронної пошти;
– перевірка електронної пошти на унікальність;
– перевірка паролю на відповідність рівню складності (довжина, наявність спеціальних символів);
– авторизація користувача за допомогою пошти і паролю;
– отримання токену для доступу до API;

– створення задач класифікації тексту;
  – введення назви задачі;
  – введення тексту для класифікації;
– створення задач класифікації документів;
  – введення назви задачі;
  – завантаження файлу документу для класифікації;
– перегляд результатів класифікації;
– видалення результатів класифікації;

4.1.4 Додаткові вимоги:

Додатково, програмне забезпечення повинно:
– мати легкий у використанні графічний інтерфейс;
– мати зрозумілий програмний інтерфейс;
– забезпечувати конфіденційність даних користувача;
– бути простим у розгортанні.

4.2 Вимоги до надійності

Передбачити контроль введення інформації та захист від некоректних дій користувача. Забезпечити цілісність інформації в базі даних.

4.3 Умови експлуатації

Умови експлуатації згідно СанПін 2.2.2.542 – 96.

4.3.1 Вид обслуговування

Вимоги до обслуговування не висуваються

4.3.2 Обслуговуючий персонал

Вимоги до обслуговуючого персоналу не висуваються

4.4 Вимоги до складу і параметрів технічних засобів

Серверна частина сервісу повинна функціонувати на IBM-сумісних персональних комп‘ютерах.

Мінімальна конфігурація технічних засобів:

– тип процесору: Intel Core i5;

– об‘єм ОЗП: 8 Гб;

– підключення до мережі Інтернет.

Рекомендована конфігурація технічних засобів:

– тип процесору: Intel Core i5;

– об‘єм ОЗП: 8 Гб;

– графічний процесор з підтримкою CUDA з об’ємом відеопам’яті не менше 4 Гб;

– підключення до мережі Інтернет.

4.5 Вимоги до інформаційної та програмної сумісності

Програмне забезпечення повинно працювати під управлінням операційних систем сімейства Linux.

4.5.1 Вимоги до вхідних даних

Вхідні дані повинні бути представлені у наступних форматах:

– звичайний текст;

– файли формату pdf з текстовим шаром.

4.5.2 Вимоги до вихідних даних

Результати повинні бути представлені в наступному форматі:

– список жанрів для класифікованого файлу.

4.5.3 Вимоги до мови розробки

Розробку виконати на мові програмування Python.

4.5.4 Вимоги до середовища розробки

Розробку виконати на платформі ОС з сімейства Linux за допомогою середовища розробки на основі текстового редактора Emacs.

4.5.5 Вимоги до представленню вихідних кодів

Вихідний код програми має бути представлений у вигляді файлів з розширеннями .py і .html.

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не висуваються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не висуваються.

4.8 Спеціальні вимоги

Згенерувати інсталяційну версію програмного забезпечення.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Попередній склад програмної документації

У склад супроводжувальної документації повинні входити наступні документи на аркушах формату А4:

– пояснювальна записка;
– технічне завдання;
– текст програми;
– програма та методика тестування;
– керівництво користувача;

Графічна частина повинна бути виконана на аркушах формату А3 та містити наступні документи:

– схема структурна варіантів використання;
– схема структурна компонентів програмного забезпечення;
– схема бази даних;
– креслення вигляду екранних форм;

5.2 Спеціальні вимоги до програмної документації

Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

№	Назва етапу	Строк	Звітність
1.	Вивчення літератури за тематикою проекту	21.02	
2.	Розробка технічного завдання	03.03	Технічне завдання
3.	Аналіз вимог та уточнення специфікацій	19.03	Специфікації програмного забезпечення
4.	Проектування структури програмного забезпечення, проектування компонентів	30.03	Схема структурна програмного забезпечення та специфікація компонентів (діаграма класів, схема алгоритму)
5.	Програмна реалізація програмного забезпечення	05.04	Тексти програмного забезпечення
6.	Тестування програмного забезпечення	10.04	Тести, результати тестування
7.	Розробка матеріалів текстової частини проекту	14.04	Пояснювальна записка
8.	Розробка матеріалів графічної частини проекту	20.04	Графічний матеріал проекту
9.	Оформлення технічної документації проекту	29.04	Технічна документація

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Пояснювальна записка
до дипломного проєкту

на тему: Веб-сервіс для автоматичної категоризації художніх творів

КПІ.ІТ-9224.045440.02.81

Київ – 2023

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

IDE	–	Integrated Development Environment – інтегроване середовище розробки.
API	–	Application Programming Interface, прикладний програмний Інтерфейс
ER	–	Entity-Relation diagram
ОС	–	Операційна система.
БД	–	База даних.
PDF	–	Portable Document Format
REST	–	Representational State Transfer

ВСТУП

У сучасному світі, що переповнений текстовою інформацією, ефективний її пошук є надзвичайно важливим завданням. З кожним роком обсяг інформації зростає, що ставить перед нами виклик знаходити нові способи обробки та організації такої величезної кількості даних.

У даному проєкті наш фокус зосереджений на художніх творах, оскільки часто зустрічаються твори, які залишилися маловідомими та малооціненими. Існує велика кількість літературних робіт, які, незважаючи на свій потенціал, не залучають достатньо уваги аудиторії, та про які майже немає відгуків або оцінок.

З метою полегшення пошуку таких художніх творів, ми ставимо перед собою завдання розробити веб-застосунок, який дозволить користувачам автоматично класифікувати ці твори за жанрами. Наш застосунок працюватиме як з простим текстом, так і з файлами, дозволяючи використовувати або веб-інтерфейс, або API.

Розроблений веб-застосунок має потенціал стати незамінним інструментом для книжкових блогерів, критиків, бібліотекарів та інших фахівців, що працюють з великим обсягом художніх творів. Він сприятиме ефективному пошуку та вибору літературних творів, допомагатиме автоматично додавати ключові слова і теги до текстів, що полегшить взаємодію між користувачами та зробить пошук більш зручним та швидким.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

На сьогоднішний день у світі існує величезна кількість текстової інформації (книг, статей, блогів, і т.д.). Через це виникає проблема пошуку потрібної інформації. Те, наскільки легко чи важко можна знайти той чи інший шматок інформації, залежить від декількох факторів.

Одним із таких факторів, що впливають на легкість пошуку та доступу до інформації, є наявність метаданих, таких як заголовки, описи та ключові слова. Ключові слова мають особливе значення, оскільки допомагають користувачам звузити пошук та швидко знайти потрібний контент. Однак, не завжди автори текстів надають такі ключові слова, або ж метадані можуть бути втрачені в процесі передачі інформації.

У будь-якому випадку, для того щоб отримати ключові слова для певного тексту, цей текст повинен хтось прочитати. Існують платформи (наприклад, goodreads), що дозволяють читачам колективно редагувати теги, проте було б непогано, якби це можна було робити автоматично, за допомогою сервісу класифікації.

1.2 Змістовний опис і аналіз предметної області

Основною задачею цього проєкту є розробка веб-застосунку, який дозволяє користувачу класифікувати за жанрами художні твори, у вигляді простого тексту, або у вигляді файлу, використовуючи для цього веб-інтерфейс, або API.

1.3 Аналіз існуючих технологій та успішних ІТ-проєктів

Проаналізуємо відоме на сьогодні алгоритмічне забезпечення у даній області та технічні рішення, що допоможуть у реалізації застосунку для категоризації художніх творів. Далі будуть розглянуті допоміжні програмні засоби, засоби розробки та готові програмні рішення.

1.3.1 Аналіз допоміжних програмних засобів та засобів розробки

Для розробки застосунку ми обрали Python – високорівневу інтерпретовану мову програмування загального використання. Такий вибір зумовлений наступними причинами:

– як вже було сказано, Python – високорівнева інтерпретована мова, що пришвидшує розробку;

– для цієї мови існує декілька великих фреймворків для машинного навчання;

– для цієї мови існує декілька веб-фреймворків;

– Python має досить багато сторонніх бібліотек, наприклад бібліотеки для парсингу веб сторінок, або бібліотеки для обробки і конвертації різних файлових форматів.

Розглянемо основні фреймворки для машинного навчання, що доступні для обраної мови:

– Tensorflow [1] – розроблений компанією Google, надає високорівневі програмні інтерфейси для побудови моделей (Keras) і підтримки розподіленого навчання на декількох пристроях;

– PyTorch [2] – розроблений компанією Facebook, цей фреймворк аналогічний Tensorflow, але має більш інтуїтивний програмний інтерфейс;

– MXNet [3] – ще одна альтернатива, розроблена Apache Software Foundation. MXNet підтримує купу інших мов програмування, а також надає інстументи для комп’ютерного зору, обробки природних мов та часових рядів.

Для навчання класифікатора ми використаємо Pytorch. У нас був досвід використання і Pytorch, і Tensorflow, і він показав, що Pytorch працює як заявлено у більшій частині випадків, ніж Tensorflow, і в цілому з ним менше мороки. До того ж, Pytorch має одну з найбільших бібліотек переднавчених моделей для обробки природніх мов – Huggin Face Transformers [4].

Розглянемо кандидати на роль веб-фреймворка, який ми використаємо для розробки веб-сервісу:

– Django [5] – веб-фреймворк для перфекціоністів з дедлайнами. Це високорівневий фреймворк, заснований на архітектурі model-view-template (MVT). Відзначається своєю швидкістю, безпекою та можливостями до масштабування;

– Flask [6] – легкий і гнучкий WSGI фреймворк, спроєктований для швидкої розробки застосунків з можливістю подальшого їх розширення у більш копмлексні рішення;

– Bottle [7] – швидкий, простий, легкий мікро-фреймворк. Розповсюджується як однофайловий модуль що залежить тільки від стандартної бібліотеки мови Python.

Для розробки веб-сервісу ми обрали Django, оскільки на наш погляд це найбільш зрілий з перелічених фреймворків, а ще ми вже маємо досвід роботи з ним.

Для збору датасету ми використаємо модулі gutenbergpy і beautifulsoup4. Перший надає API для зручного завантаження текстів з бібліотеки Project Gutenberg, що містить книги з публічного домену. Другий дозволяє парсити веб сторінки. Ми використаємо це для збору тегів.

Для конвертації файлів у звичайний текст ми використаємо бібліотеку Pandoc. Вона підтримує великий набір файлових форматів і має інтерфейс для обраної мови (сама бібліотека написана на мові Haskell).

У якості середовища розробки веб-сервісу ми використаємо Emacs, для розробки класифікатору – Jupyter Notebook.

1.3.2 Аналіз відомих програмних продуктів

Напевно, слід виділити два типи програмних продуктів або сервісів, що могли б бути використані для вирішення нашої задачі:

– онлайн бібліотеки і каталоги, що індексують літературу і збирають метадані, у тому числі теги. Існує вірогідність, що потрібний твір уже є у подібному каталозі, і тоді задача зводиться до того, щоб витягнути звідти уже

існуючі метадані. Звичайно, що такий трюк не спрацює для довільного тексту, якого немає в каталозі;

– онлайн класифікатори і API для навчання моделей.

Розглянемо декілька готових програмних продуктів першого типу:

– Goodreads – популярна платформа для соціального каталогування. Має досить велику базу книг, їх метаданих і коментарів. До 2020 року мала API, але його закрили;

– LibraryThing – схожа на Goodreads платформа. На відміну від останньої, все ще має API, але вибірка книг і якість тегів не найкраща.

І декілька програмних продуктів другого типу:

– MonkeyLearn – сервіс текстової аналітики, що орієнтується на бізнес. Надає досить багато інструментів, у тому числі класифікатори для топік лейбелінгу, проте цей класифікатор пропонує трохи інші теги, ніж ті, що нам потрібні. Окрім цього, дозволяє вчити свої моделі;

– uClassify – безкоштовний веб-сервіс, що дозволяє створювати і використовувати класифікатори тексту. Має доволі великий набір готових класифікаторів.

Занесемо усі переваги і недоліки кожного продукту до таблиці, щоб порівняти з нашим проєктом (таблиця 1.1). Як можна бачити, наш проєкт надає API для роботи з тегами і підходящий класифікатор, чого не роблять інші продукти.

Таблиця 1.1 – Порівняння функціоналу дипломної роботи і аналогів

Програмний продукт	Наявність API	Якість тегів	Наявність тегів для довільного тексту	Можливість навчання своїх моделей	Наявність підходящого класифікатора
Дипломна робота	✓		✓	✗	✓
Goodreads	✗	✓	✗	n/a	n/a

Продовження таблиці 1.1

LibraryThing	✓	✗	✗	n/a	n/a
MonkeyLearn	✓	✗	✓	✓	✗
uClassify	✓	✗	✓	✓	✗

1.4 Аналіз вимог до програмного забезпечення

Головними функціями програмного забезпечення, що планується розробити при виконанні даної дипломної роботи є класифікація користувачами тексту та документів, і робота з результатами класифікації. Детальніше на рис. 1.1.

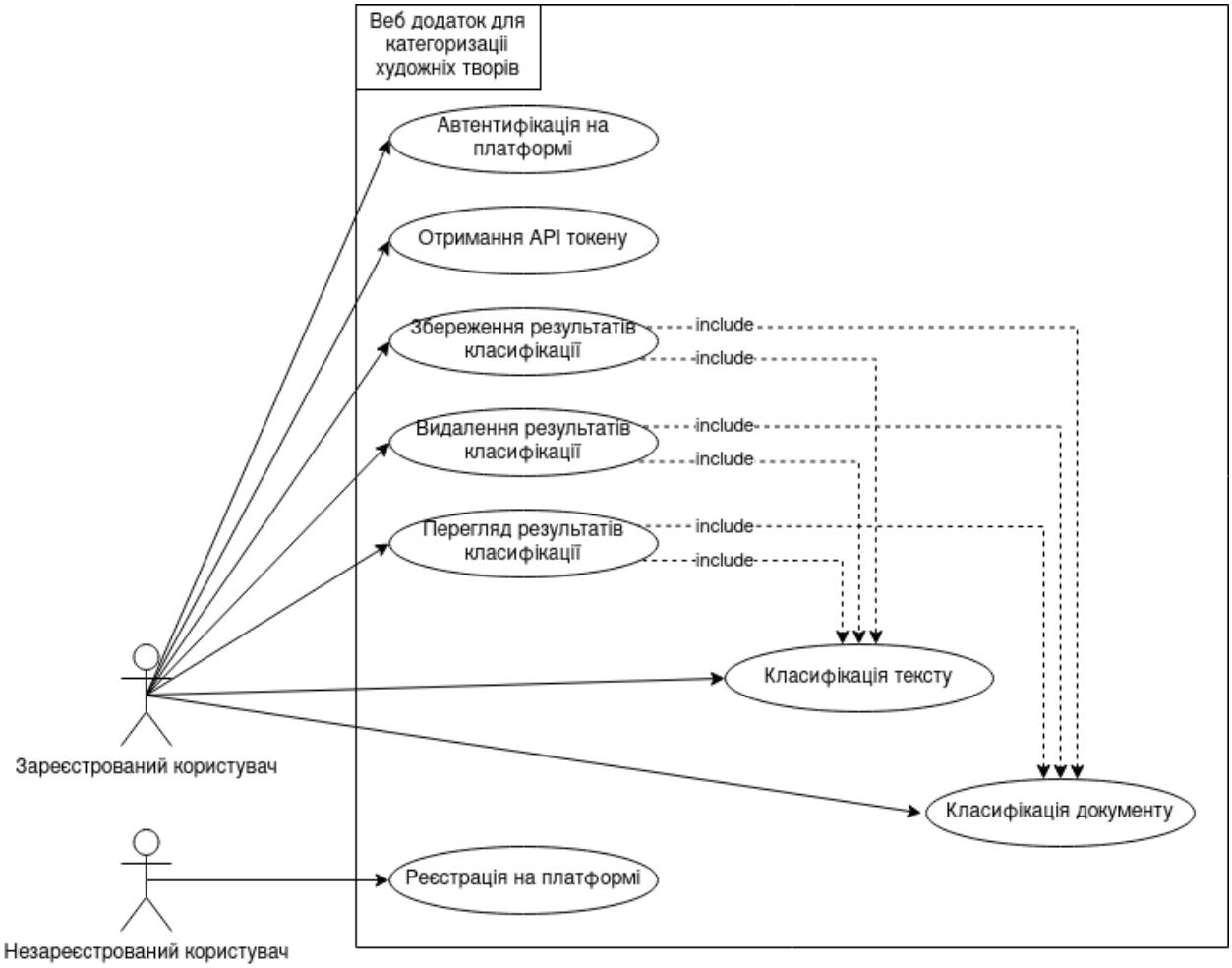


Рисунок 1.1 – Діаграма варіантів використання

В таблицях 1.2-1.9 наведені варіанти використання програмного забезпечення.

Таблиця 1.2 – Варіант використання UC-01

Use case name	Реєстрація користувача
Use case ID	UC-01
Goals	Створення нового облікового запису користувача.
Actors	Незареєстрований користувач.
Trigger	Користувач обирає опцію реєстрації.
Pre-conditions	Незареєстрований користувач на сторінці автентифікації.
Flow of Events	Користувач вводить електронну пошту, ім’я користувача, пароль та підтверджує пароль. Система перевіряє введені дані та створює обліковий запис.
Extension	Якщо користувач ввів різні паролі, занадто слабкий пароль або невалідну електронну пошту, система повертає помилку і просить перевірити дані.
Post-Condition	Користувача зареєстровано і перенаправлено на сторінку автентифікації.

Таблиця 1.3 – Варіант використання UC-02

Use case name	Автентифікація користувача
Use case ID	UC-02
Goals	Автентифікація користувача.
Actors	Неавтентифікований користувач.
Trigger	Користувач обирає опцію автентифікації.
Pre-conditions	Користувач у модальному вікні профілю.

Продовження таблиці 1.3

Flow of Events	Користувач вводить ім’я користувача та пароль. Система перевіряє введені дані та автентифікує користувача.
Extension	Якщо користувач неправильні ім’я та пароль, система відхиляє спробу автентифікації.
Post-Condition	Користувача автентифіковано, і тепер він має доступ до сервісу.

Таблиця 1.4 – Варіант використання UC-03

Use case name	Вихід з системи
Use case ID	UC-03
Goals	Вихід з системи.
Actors	Авторизований користувач.
Trigger	Користувач обирає опцію виходу з системи.
Pre-conditions	Користувач у модальному вікні профілю.
Flow of Events	Користувач обирає опцію виходу з системи у модальному вікні профілю.
Extension	-
Post-Condition	Користувач виходить з системи.

Таблиця 1.5 – Варіант використання UC-04

Use case name	Отримання API токену
Use case ID	UC-04
Goals	Отримання API токену.
Actors	Авторизований користувач.
Trigger	Користувач обирає опцію отримання токену.

Продовження таблиці 1.6

Pre-conditions	Користувач у модальному вікні профілю.
Flow of Events	Користувач обирає опцію отримання токену у модальному вікні профілю.
Extension	-
Post-Condition	Користувача перенаправляє на сторінку, де він отримує токен доступу до API.

Таблиця 1.6 – Варіант використання UC-05

Use case name	Класифікація тексту
Use case ID	UC-05
Goals	Класифікація тексту.
Actors	Автентифікований користувач.
Trigger	Система отримує запит класифікації тексту.
Pre-conditions	Користувач надсилає запит класифікації тексту (Через форму класифікації, або через API).
Flow of Events	Система ставить запит у чергу і зберігає його статус у базі даних. Через деякий час запит оброблюється, після цього система оновлює його статус, зберігаючи результати.
Extension	-
Post-Condition	Результат класифікації зберігається у системі, користувач отримує ідентифікатор запиту.

Таблиця 1.7 – Варіант використання UC-06

Use case name	Класифікація документу.
Use case ID	UC-06
Goals	Класифікація документу.

Продовження таблиці 1.7

Actors	Автентифікований користувач.
Trigger	Система отримує запит класифікації документу.
Pre-conditions	Користувач надсилає запит класифікації документу (Через форму класифікації, або через API).
Flow of Events	Система ставить запит у чергу і зберігає його статус у базі даних. Через деякий час запит оброблюється, після цього система оновлює його статус, зберігаючи результати.
Extension	У випадку помилки (наприклад, файл у непідтримуваному форматі), система оновлює статус на негативний.
Post-Condition	Результат класифікації зберігається у системі, користувач отримує ідентифікатор запиту.

Таблиця 1.8 – Варіант використання UC-07

Use case name	Перегляд результатів класифікації
Use case ID	UC-07
Goals	Отриманя результатів класифікації.
Actors	Автентифікований користувач
Trigger	Система отримує запит на отримання результатів класифікації.
Pre-conditions	Користувач надсилає запит на отримання результатів класифікації.
Flow of Events	Система отримує запит, отримує усі результати, що належать даному користувачу і повертає їх список.
Extension	У випадку, якщо у користувача ще немає результатів, система повертає пустий список.
Post-Condition	Користувач отримав список результатів класифікації.

Таблиця 1.9 – Варіант використання UC-08

Use case name	Видалення результатів класифікації
Use case ID	UC-08
Goals	Видалення результатів класифікації
Actors	Автентифікований користувач
Trigger	Система отримує запит на видалення результатів класифікації.
Pre-conditions	Користувач надсилає запит на видалення результатів класифікації.
Flow of Events	Система отримує запит на видалення, перевіряє що результат з наданим ідентифікатором існує та належить користувачу, видаляє результат і повертає повідомлення про успішне видалення.
Extension	У випадку, якщо результату з наданим ідентифікатором не існує, або він не належить користувачу, система повертає повідомлення про відмову.
Post-Condition	Результат видалено з системи.

1.4.1 Розроблення функціональних вимог

У таблиці 1.10 наведено загальну модель вимог. У таблицях 1.11 – 1.27 наведений опис функціональних вимог до програмного забезпечення. Матрицю трасування вимог наведено у таблиці 1.28.

Таблиця 1.10 – Загальна модель вимог

№	Назва функціональної вимоги	ID вимоги
1	Система автентифікації	FR-1
1.1	Реєстрація користувачів	FR-2

Продовження таблиці 1.10

1.2	Автентифікація користувачів	FR-3
1.3	Вихід з системи	FR-4
1.4	Отримання токену для доступу до API	FR-5
2	Класифікація тексту	FR-6
3	Класифікація документу	FR-7
4	Черга задач	FR-8
5	Результати класифікації	FR-9
5.1	Оновлення результатів класифікації	FR-10
5.2	Отримання результатів класифікації	FR-11
5.3	Видалення результатів класифікації	FR-12
6	Програмний інтерфейс	FR-13
6.1	[API] програмний інтерфейс для класифікації тексту	FR-14
6.2	[API] програмний інтерфейс для класифікації документу	FR-15
6.3.1	[API] програмний інтерфейс для отримання результатів класифікації	FR-16
6.3.2	[API] програмний інтерфейс для видалення результатів класифікації	FR-17

Таблиця 1.11 – Функціональна вимога FR-1

Назва	Система автентифікації
Опис	Система повинна мати функції реєстрації, автентифікації, отримання токену для доступу до API та функцію виходу з системи.

Таблиця 1.12 – Функціональна вимога FR-2

Назва	Реєстрація користувачів
Опис	Система повинна мати функцію реєстрації користувачів шляхом введення імені користувача, електронної пошти та паролю.

Таблиця 1.13 – Функціональна вимога FR-3

Назва	Автентифікація користувачів
Опис	Система повинна мати фінукцію автентифікації користувачів шляхом введення імені користувача та паролю.

Таблиця 1.14 – Функціональна вимога FR-4

Назва	Вихід з системи
Опис	Користувач повинен мати можливість вийти зі свого облікового запису, завершивши сеанс.

Таблиця 1.15 – Функціональна вимога FR-5

Назва	Отримання токену доступу до API
Опис	Система повинна мати функції генерації та видачі токенів для доступу до API.

Таблиця 1.16 – Функціональна вимога FR-6

Назва	Класифікація тексту
Опис	Система повинна мати функцію класифікації тексту за жанрами.

Таблиця 1.17 – Функціональна вимога FR-7

Назва	Класифікація документу
Опис	Система повинна мати функцію перетворення документів у текст і подальшої класифікації цих документів за жанрами.

Таблиця 1.18 – Функціональна вимога FR-8

Назва	Черга задач
Опис	Запити класифікації повинні оброблюватись тільки у тому випадку, коли для цього є вільні ресурси. Для цього система повинна мати чергу задач, і одночасно виконувати не більше заданої кількості запитів.

Таблиця 1.19 – Функціональна вимога FR-9

Назва	Результати класифікації
Опис	Система повинна зберігати та надавати доступ до результату та статусу кожної задачі класифікації.

Таблиця 1.20 – Функціональна вимога FR-10

Назва	Оновлення результатів класифікації
Опис	Система повинна оновлювати статус та результат кожної задачі на наступних етапах: 1. Коли задача надійшла у систему (статус PENDING) 2. Коли система почала виконувати задачу (статус STARTED) 3. Коли система виконала задачу (статус SUCCESS або FAILURE). На цьому етапі повинні бути збережені результати, якщо вони є.

Таблиця 1.21 – Функціональна вимога FR-11

Назва	Отримання результатів класифікації
Опис	Система повинна мати функції отримання всіх результатів (або конкретного результату за його ідентифікатором), що належать користувачу.

Таблиця 1.22 – Функціональна вимога FR-12

Назва	Видалення результатів класифікації
Опис	Система повинна мати функцію видалення конкретного результату за його ідентифікатором, за умови що він належать користувачу.

Таблиця 1.23 – Функціональна вимога FR-13

Назва	[API] Програмний інтерфейс
Опис	Система повинна мати програмний інтерфейс, що дозволяє автоматизувати створення задач класифікації і роботу з результатами. Цей інтерфейс повинен бути доступний тільки у разі надання токену доступу до API.

Таблиця 1.24 – Функціональна вимога FR-14

Назва	[API] Програмний інтерфейс для класифікації тексту
Опис	Система повинна мати програмний інтерфейс для створення задач класифікації тексту.

Таблиця 1.25 – Функціональна вимога FR-15

Назва	[API] Програмний інтерфейс для класифікації документів
Опис	Система повинна мати програмний інтерфейс для створення задач класифікації документів.

Таблиця 1.26 – Функціональна вимога FR-16

Назва	[API] Програмний інтерфейс для отримання результатів класифікації
Опис	Система повинна мати програмний інтерфейс для отримання результатів (або конкретного результату за його ідентифікатором) класифікації, що належать користувачу.

Таблиця 1.27 – Функціональна вимога FR-17

Назва	[API] Програмний інтерфейс для видалення результатів класифікації
Опис	Система повинна мати програмний інтерфейс для видалення конкретного результату класифікації за його ідентифікатором за умови, що він належить користувачу.

Таблиця 1.28 – Матриця трасування вимог

UC\FR	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	X	X															
2	X		X														
3	X			X													
4	X				X												
5						X		X	X	X			X	X			
6							X	X	X	X			X		X		
7											X		X			X	
8												X	X				X

1.4.2 Розроблення нефункціональних вимог

Розроблена система повинна:

– мати легкий у використанні графічний інтерфейс;

– мати зрозумілий програмний інтерфейс;

– забезпечувати конфіденційність особистих та просто даних користувачів;

– бути простою у розгортанні.

1.5 Постановка задачі

Таким чином, нашою метою є полегшити пошук художніх творів. Досягнути цієї мети ми зможемо, виконавши наступні задачі:

1. зібрати датасет класифікованих текстів, використовуючи літературу з публічного домену;
2. Розробити і навчити модель класифікації, удостовіритись, що вона працює правильно;
3. Розробити сервіс автоматичної категоризації творів;
4. Розробити функціонал категоризації звичайного тексту;
5. Розробити функціонал категоризації документів;

6. Розробити систему автентифікації користувачів сервісу;
7. Розробити систему черги, що дозволить сервісу продовжувати працювати при великій кількості запитів;

Висновки до розділу

У даному розділі ми сформулювали загальні положення, описали і проаналізували предметну область застосунку для категоризації тексту.

Ми розглянули існуючі алгоритми та моделі для категоризації тексту, і визначено, що розробляти своє рішення не потрібно.

Також ми склали список допоміжних програмних засобів та засобів розробки, що будуть використані у процесі виконання роботи.

Після цього ми провели аналіз вимог до програмного забезпечення, що розробляється, описано варіанти використання, функціональні і нефункціональні вимоги.

У кінці ми сформульовали задачі, що мають бути виконані у процесі написання даної роботи.

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

Розглянемо основні бізнес процеси системи: реєстрацію та автентифікацію користувачів і обробку користувацьких запитів класифікації. Для цього скористаємося BPMN діаграмами (рис. 2.1, 2.2 та 2.3).

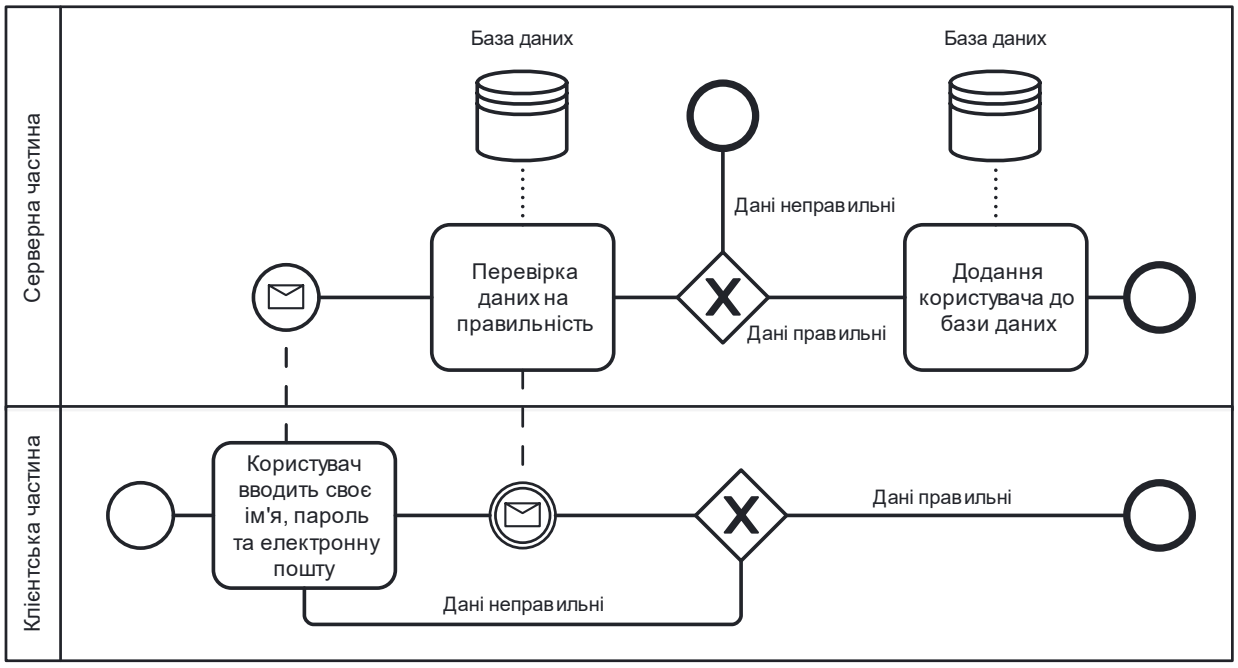


Рисунок 2.1 – BPMN діаграма процесу реєстрації.

Опис послідовності дій для процесу реєстрації:

– користувач переходить на сторінку реєстрації;

– користувач вводить свої ім’я, пароль та електронну пошту і відсилає форму;

– дані пересилаються на сервер, де відбувається їх валідація;

– у випадку, якщо дані правильні, користувач додається до бази даних;

– у іншому випадку, користувач повторно вводить дані.

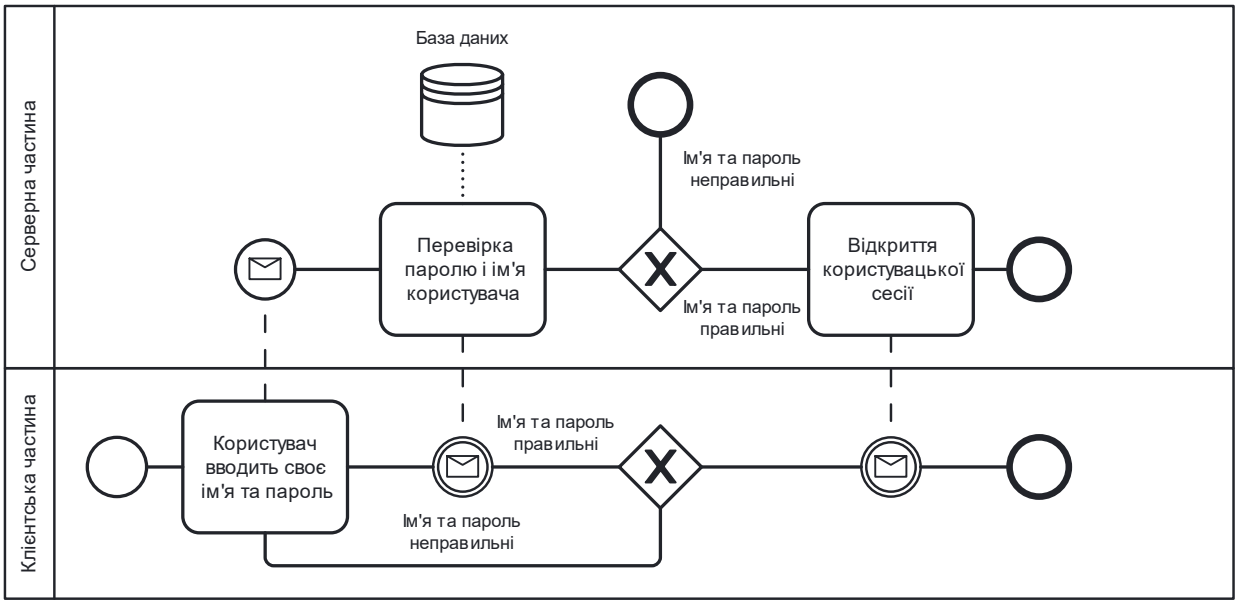


Рисунок 2.2 – BPMN діаграма процесу автентифікації.

Опис послідовності дій для процесу реєстрації:

– користувач переходить на сторінку автентифікації;

– користувач вводить свої ім’я та пароль і відсилає форму;

– дані пересилаються на сервер, де відбувається їх перевірка;

– у випадку, якщо дані правильні, сервер відкриває користувацьку сесію;

– у іншому випадку, користувач повторно вводить дані.

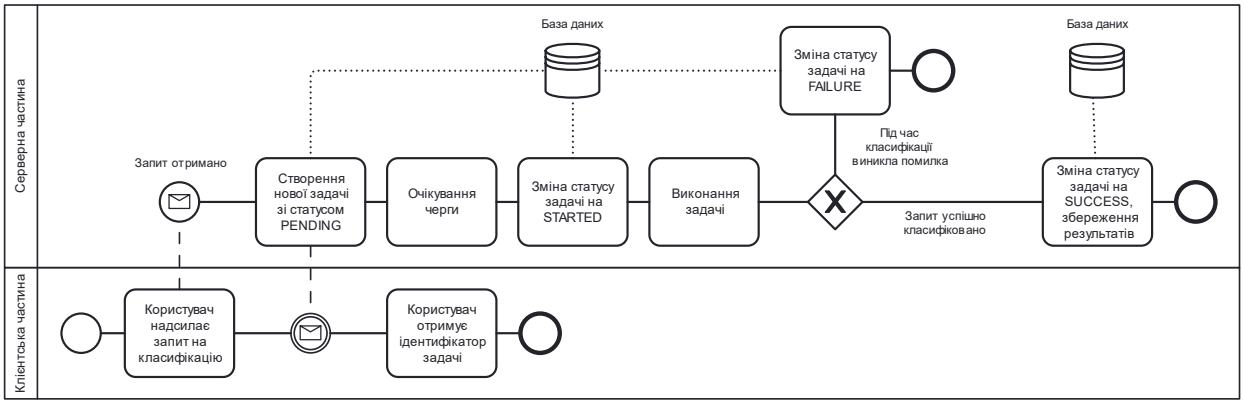


Рисунок 2.3 – BPMN діаграма процесу обробки користувацьких запитів класифікації.

Опис послідовності дій для процесу обробки користувацьких запитів класифікації:

– користувач надсилає запит на класифікацію. Цей запит може містити або текст, або документ, який потрібно класифікувати;

– сервер отримує запит і створює нову задачу, встановлюючи її статус у PENDING, і ставить її в чергу. Після цього він повертає ідентифікатор задачі клієнту;

– користувач отримує ідентифікатор, і після цього його взаємодія у рамках даного процесу завершується;

– коли до задачі доходить її черга, сервер змінює її статус на STARTED і починає її виконувати;

– по завершенню виконання задачі, сервер знову змінює її статус і зберігає результати;

– Після отримання ідентифікатору користувач може виконувати інші дії, наприклад перевірити статус задачі, якій належить ідентифікатор, або видалити її.

2.2 Архітектура програмного забезпечення

Розглянемо архітектуру веб-сервісу, який ми розробили для взаємодії з класифікатором.

Веб-сервіс розроблено з використанням монолітної архітектури за допомогою фреймворку Django, який в свою чергу дотримується шаблону MVT (model-view-template), тому для планування взаємодії між компонентами системи доречно використовувати саме цей шаблон.

Model-view-template – це шаблон, дуже схожий на MVC (model view controller) але з деякими відмінностями. На відміну від MVC, частина функцій представлення зміщена у шаблон – це те, що побачить користувач. Представлення у свою чергу, вміщає в себе частину функцій контролера, і відповідає за рендеринг шаблонів і взаємодію з моделлю. Модель залишається як є.

Розглянемо схему взаємодії компонентів веб-сервісу (рис. 2.4):

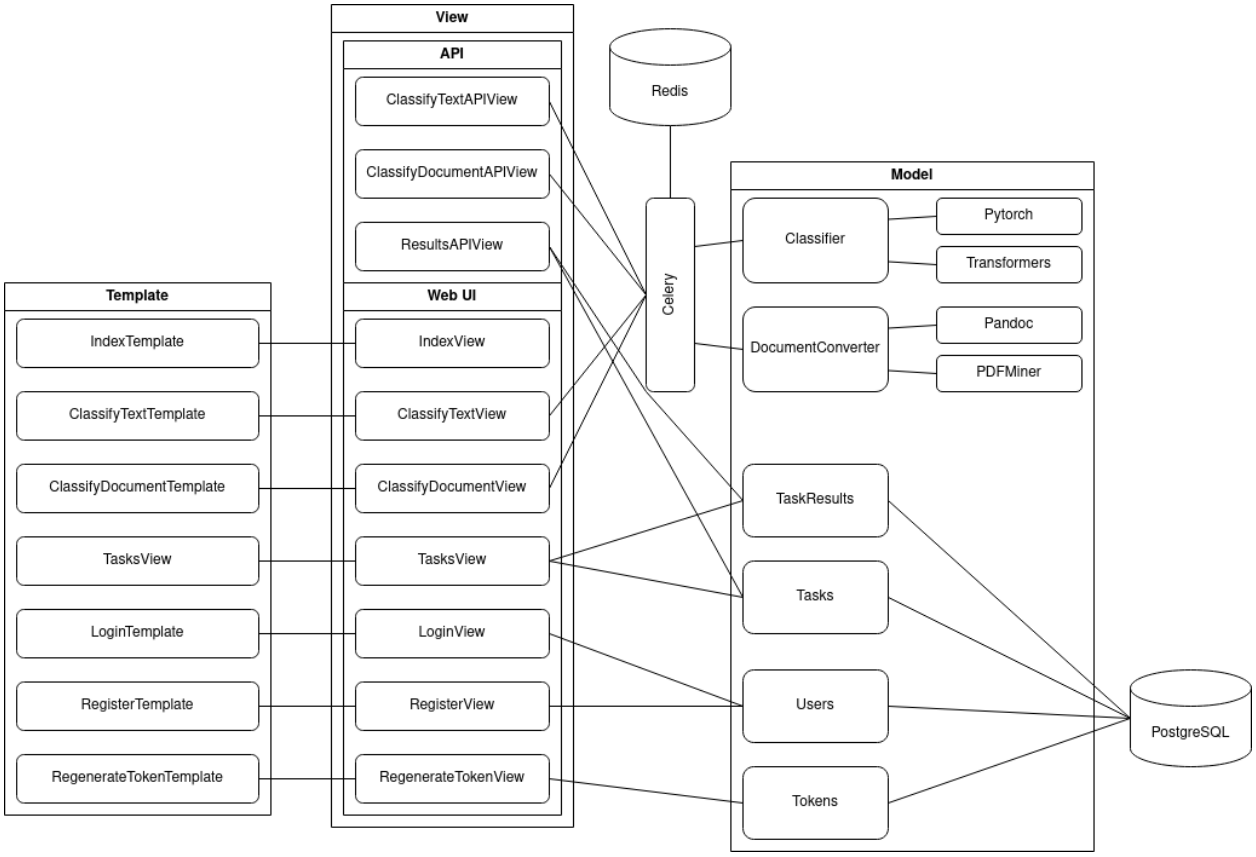


Рисунок 2.4 – Схема взаємодії компонентів сервісу

– Classifier, DocumentConverter – компоненти для роботи з класифікатором і документами. Оскільки функціонал, що надають ці компоненти, потребує досить багато ресурсів, взаємодія з цими компонентами відбувається у черзі задач на базі Celery і Redis;

– Tasks, TaskResults – моделі, призначені для опису результатів виконання задач класифікації у базі даних;

– Users – модель, що описує користувача системи;

– Tokens – модель, що описує API токен, що належить користувачу;

– IndexView – представлення головної сторінки застосунку. Воно не взаємодії з моделлю, тому для його роботи потрібен тільки відповідний шаблон;

– TextClassifierView, FileClassifierView – представлення сторінок класифікації тексту і класифікації файлів. Окрім відповідних шаблонів, для

роботи їм також потрібен класифікатор, який відноситься до моделі. Представлення FileClassifierView також потребує компонент DocumentConverter для конвертації файлів;

– TextClassifierAPIView, FileClassifierAPIView – представлення ендпоінтів API. Для роботи їм не потрібен шаблон, вони взаємодіють тільки з моделлю;

– ResultsView – представлення для роботи з результатами класифікації. Цей компонент взаємодіє з моделями задач і результатів задач, а також з відповідним шаблоном;

– ResultsAPIView – аналогічне представлення для роботи з результатами класифікації, але на стороні API;

– LoginView, RegisterView – представлення сторінок автентифікації та реєстрації. Для їх роботи потрібні модель користувачів та відповідні шаблони;

– RegenerateTokenView – представлення сторінки оновлення токену API;

2.3 Конструювання програмного забезпечення

Почнемо з класифікатора. Його ми розробили з використанням бібліотек Pytorch і Hugging Face Transformers. Остання надає досить великий набір переднавчених моделей обробки природніх мов.

У основу класифікатора ми поклали переднавчену модель Longformer [8], готову для подальшого донавчання. В теорії ця модель може обробляти послідовності слів довжиною до 16000 токенів, але наше обладнання дозволяє використовувати лише 4000 токенів. Для того, щоб обійти цю проблему, занадто довгі послідовності спочатку проганяються через алгоритм екстрактивної сумаризації, з метою зменшення їх довжини. У якості такого алгоритму ми обрали TextRank. Загальну структуру класифікатора наведено на рис. 2.5.

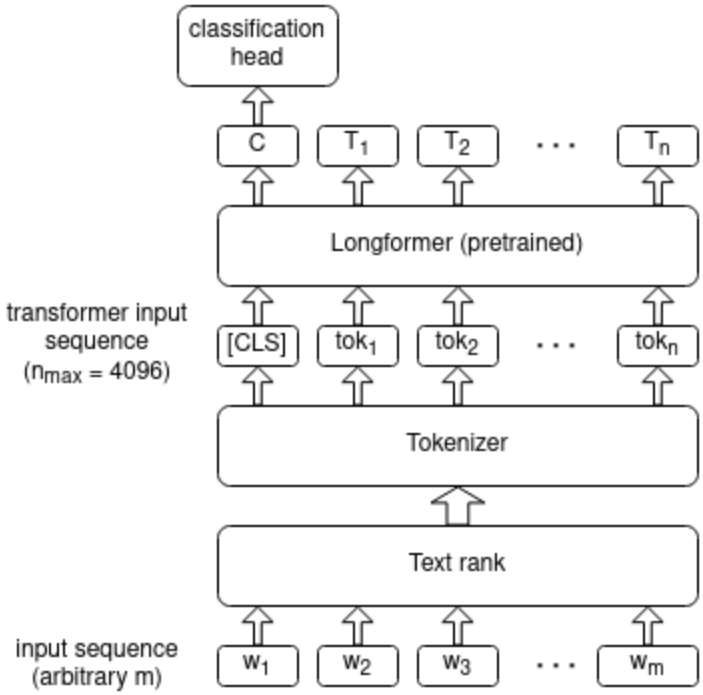


Рисунок 2.5 – Діаграма класів сервісу

Окрім Longformer, ми спробували використовувати інші моделі на основі механізму уваги, як от:

– BERT [9] (Bidirectional Encoder Representations from Transformers) – модель глибокого навчання, що базується на трансформерній архітектурі. Ця модель здатна досягати вражаючих результатів у завданнях розуміння мови, але у неї є суттєвий недолік – квадратна складність механізму уваги, що обмежує довжину послідовностей, які можуть бути оброблені;

– TransformerXL [10] – розширення оригінальної трансформерної архітектури, призначене для обробки послідовностей з довшої залежності. TransformerXL використовує механізми, такі як «посилення пам'яті» (memory masking) і «позиційне кодування» (positional encoding), щоб ефективно моделювати довгострокові залежності в тексті;

– BigBird [11] – ще одна модифікація оригінальної архітектури Transformer, яка використовує структуру «глобальних уваг», яка розбиває послідовність на блоки, і дозволяє моделі обмінюватись інформацією між віддаленими токенами без необхідності напряму обчислювати увагу між ними;

BERT ми відкинули через обмеження максимальної довжини послідовностей, які він може обробляти – всього 512 токенів. TransformerXL і BigBird показували результати, схожі з результатами, що показував Longformer, але Longformer вимагав менше ресурсів для навчання. Саме тому ми вибрали його як фінальний варіант.

Класифікатор ми навчили на творах художньої літератури з публічного домену, взятих з архіву Project Gutenberg. Жанри для кожного такого твору ми отримали з сайту Goodreads методом парсингу веб-сторінок. Для цього використовувалась бібліотека beautifulsoup4.

Перейдемо до конструювання веб-сервісу. Для реалізації компонентів, описаних у попередньому розділі, ми використали абстракції, що надає Django і Django REST Framework. Останній надає більш зручний функціонал для роботи з REST API. Для процесу реєстрації ми використали плагін Django Registration.

Конвертер документів ми реалізували за допомогою бібліотеки pdfminer.six, яка надає функціонал конвертації файлів pdf у звичайний текст.

Для створення черги задач ми використали бібліотеку Celery у зв’язці з Redis. Для збереження результатів задач до бази даних використано плагін Django Celery Results.

У якості системи управління базами даних використовується Postgres. У базі даних зберігаються результати виконання задач та інформація про користувачів. Розглянемо ER діаграму для основних сутностей (рис. 2.6).

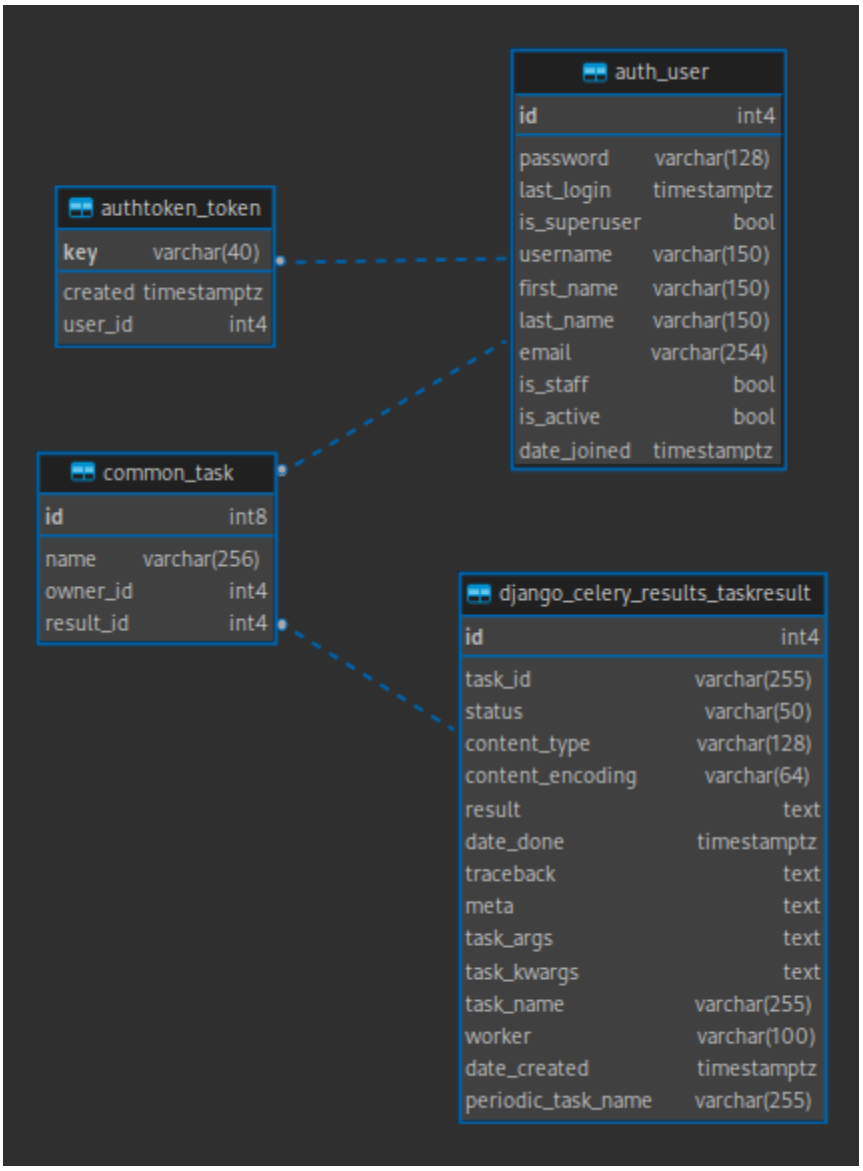


Рисунок 2.6 – ER діаграма для основних сутностей.

Опишемо основні таблиці бази даних (таблиці 2.1-2.4).

Таблиця 2.1 – Опис таблиці task

Назва поля	Тип даних	Опис
id	int8	Ідентифікаційний номер задачі
name	varchar	назва задачі
Назва поля	Тип даних	Опис

Продовження таблиці 2.1

owner_id	int4	ідентифікаційний номер користувача, якому належить задача.
result_id	int4	ідентифікаційний номер результату задачі.

Таблиця 2.2 – Опис таблиці taskresult

Назва поля	Тип даних	Опис
id	int8	Ідентифікаційний номер результату.
task_id	varchar	UUID задачі.
status	varchar	Статус виконання задачі.
content_type	varchar	Content type результату.
content_encoding	varchar	Content encoding результату.
result	text	Результат виконання.
date_done	timestamptz	Дата завершення виконання.
traceback	text	Трасування стеку, для випадків, коли задача кидає виключення.
meta	text	Метадані задачі.
task_args	text	Аргументи, з якими була запущена задача.

Продовження таблиці 2.2

Назва поля	Тип даних	Опис
task_kwargs	text	Іменовані аргументи, з якими була запущена задача.
task_name	varchar	Назва задачі.
worker	varchar	Назва процесу, що виконував задачу.
date_created	timestamptz	Дата створення задачі.
periodic_task_name	varchar	Назва періодичної задачі.

Таблиця 2.3 – Опис таблиці user

Назва поля	Тип даних	Опис
id	int8	Ідентифікаційний номер користувача.
password	varchar	Захешований пароль користувача.
last_login	timestamptz	Дата останнього логіну.
is_superuser	bool	Позначка, чи є користувач суперюзером.
username	varchar	Ім’я користувача.
first_name	varchar	Справжнє ім’я користувача.
last_name	varchar	Справжнє прізвище користувача.
email	varchar	Електронна пошта користувача.
is_staff	bool	Позначка, чи є користувач працівником адміністрації.
Назва поля	Тип даних	Опис
is_active	bool	Позначка, чи є користувач активним.
date_joined	timestamptz	Дата та час реєстрації користувача.

Таблиця 2.4 – Опис таблиці token

Назва поля	Тип даних	Опис
key	varchar	Ключ токену.
created	timestamptz	Час створення токену.
user_id	int4	ідентифікаційний номер користувача, якому належить токен.

Опис утиліт, бібліотек та іншого стороннього програмного забезпечення, використаного у розробці наведено в таблиці 2.5.

Таблиця 2.5 – Опис утиліт

№ п/п	Назва утиліти	Опис застосування
1	Emacs	Emacs - це потужний текстовий редактор, який відомий своєю високою налаштуваністю і розширюваністю. Він надає розширену функціональність для редагування тексту, розробки програмного забезпечення та автоматизації задач. Emacs має вбудовану систему розширень, яка дозволяє користувачам додавати нові функції, налаштовувати його під свої потреби та створювати скрипти для автоматизації повсякденних завдань. Використовується як головне середовище розробки.
2	Jupyter Notebook	Jupyter Notebook - це інтерактивне середовище для розробки програмного забезпечення, яке дозволяє комбінувати виконання коду, створення текстових описів та візуалізацію даних в одному документі. Він підтримує багато мов програмування, включаючи Python, R, Julia та інші. Jupyter Notebook дозволяє виконувати код почастинно, спрощуючи розробку та відлагодження програм. Головне середовище розробки моделі на основі машинного навчання, що потрібна для роботи класифікатора.

Продовження таблиці 2.5

№ п/п	Назва утиліти	Опис застосування
3	DBeaver	DBeaver -- це універсальний клієнт баз даних, який надає зручний інтерфейс для роботи з різноманітними системами керування базами даних (СКБД). Він підтримує багато популярних СКБД, таких як MySQL, PostgreSQL, Oracle, SQL Server, SQLite, MongoDB та багато інших.
4	Pandas	Pandas -- це бібліотека для обробки та аналізу даних в середовищі Python. Вона надає високорівневі структури даних, такі як DataFrame, що дозволяють легко маніпулювати табличними даними.
5	Pytorch	Фреймворк глибокого навчання.
6	Hugging Face Transformers	Hugging Face Transformers -- це бібліотека для роботи з моделями обробки природної мови на основі трансформерів. Вона побудована поверх фреймворку PyTorch і надає широкий спектр готових до використання моделей та інструментів для обробки тексту. Hugging Face Transformers зосереджена на легкості використання та доступності моделей глибокого навчання для розробників. Вона містить набір переднавчених моделей, таких як BERT, GPT, RoBERTa, DistilBERT і багато інших.

Продовження таблиці 2.5

№ п/п	Назва утиліти	Опис застосування
7	Sumy	Sumy -- це бібліотека для автоматичного стиснення та резюмування тексту в середовищі Python. Вона надає простий спосіб створення короткого огляду або стислого викладу великого текстового документа. Sumy використовує різні алгоритми та моделі, такі як LSA (латентно-семантичний аналіз), TextRank та Luhn, для виявлення ключових речень або фраз, які найкраще представляють зміст тексту.
8	Django	Django -- це високорівневий веб-фреймворк, розроблений на мові програмування Python. Він пропонує зручний інтерфейс та набір інструментів для ефективної розробки, розгортання та управління веб-додатками. Django базується на концепції моделі-представлення-контролера (MVC) та розширює його до моделі-представлення-шаблону (MVT).
9	Django REST Framework	Django REST Framework (DRF) -- це розширення для фреймворку Django, яке надає потужні інструменти для розробки веб-сервісів та API з використанням стандартних протоколів, таких як HTTP і REST. Він робить розробку API швидкою, простою і ефективною, забезпечуючи набір готових компонентів і патернів проектування.

Продовження таблиці 2.5

№ п/п	Назва утиліти	Опис застосування
10	Django Registration	Django Registration -- це бібліотека для Django, яка надає зручні інструменти для реалізації системи реєстрації користувачів у веб-додатках. Вона допомагає вам створити реєстраційну форму, обробляти дані користувача, зберігати їх у базі даних та виконувати різноманітні дії після успішної реєстрації.
11	Django Celery Results	Django Celery Results -- це розширення для Django та Celery, яке надає можливість зберігати результати виконання асинхронних завдань, оброблених за допомогою Celery, у базі даних Django. За замовчуванням Celery не зберігає результати завдань, що виконуються асинхронно, проте Django Celery Results дозволяє зручно зберігати та отримувати результати цих завдань.
12	Django Bootstrap5	Django Bootstrap5 -- це інтеграційний пакет для Django, який дозволяє використовувати Bootstrap 5 в Django-проекті. Bootstrap 5 -- це популярний фреймворк для розробки веб-інтерфейсів, який надає набір стилів, компонентів та інструментів для швидкого створення сучасних та адаптивних веб-додатків.
13	Celery	Celery -- це розподілена система обробки завдань (task queue), яка дозволяє виконувати асинхронні завдання в розподіленому середовищі. Вона дозволяє відокремити обробку довгих або трудомістких операцій від основного потоку веб-додатка, що дозволяє збільшити масштабованість та продуктивність системи.

Продовження таблиці 2.5

№ п/п	Назва утиліти	Опис застосування
14	PostgreSQL	PostgreSQL (часто називається просто Postgres) -- це потужна та розширювана система керування базами даних (СКБД), яка надає надійне зберігання та обробку структурованої інформації. Вона використовує мову SQL для взаємодії з базою даних та має велику кількість розширень та функціональності.
15	Redis	Redis (Remote Dictionary Server) -- це швидка in-memory система зберігання даних. Зазвичай вона використовується як кеш.

2.4 Аналіз безпеки даних

Далі ми провели аналіз безпеки розробленого програмного забезпечення.

Завдяки використанню Django вирішується цілих ряд питань безпеки, як от безпечна реєстрація та автентифікація користувачів, валідація даних перед збереженням їх у базі даних, та безпечне зберігання конфіденційних даних, як от паролів.

Також ми обмежили доступ до даних інших користувачів, тобто кожен користувач може переглядати і видаляти тільки свої результати класифікації.

Доступ до API надається за токеном доступу, що є унікальним для кожного користувача.

Звичайно, що для нормальної роботи більшої частини з вищеперелічених заходів безпеки потрібно увімкнути https.

Загалом, можна сказати, що розроблене ПЗ має достатньо високий рівень безпеки і надійності. До того ж, для подальшого зменшення ризиків,

пов’язаних з використанням даного ПЗ, підтримується варіант розгортання його локальної копії у безпечному середовищі.

Висновки до розділу

У даному розділі ми змоделювали основні процеси, які реалізує програмне забезпечення, побудували його архітектуру та виділили його компоненти.

Після цього ми описали деталі імплементації ПЗ, і проаналізували загрози безпеці даних.

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз якості ПЗ

Аналіз якості ПЗ проводився за допомогою сервісу embold. Результати аналізу можна бачити на рис. 3.1.

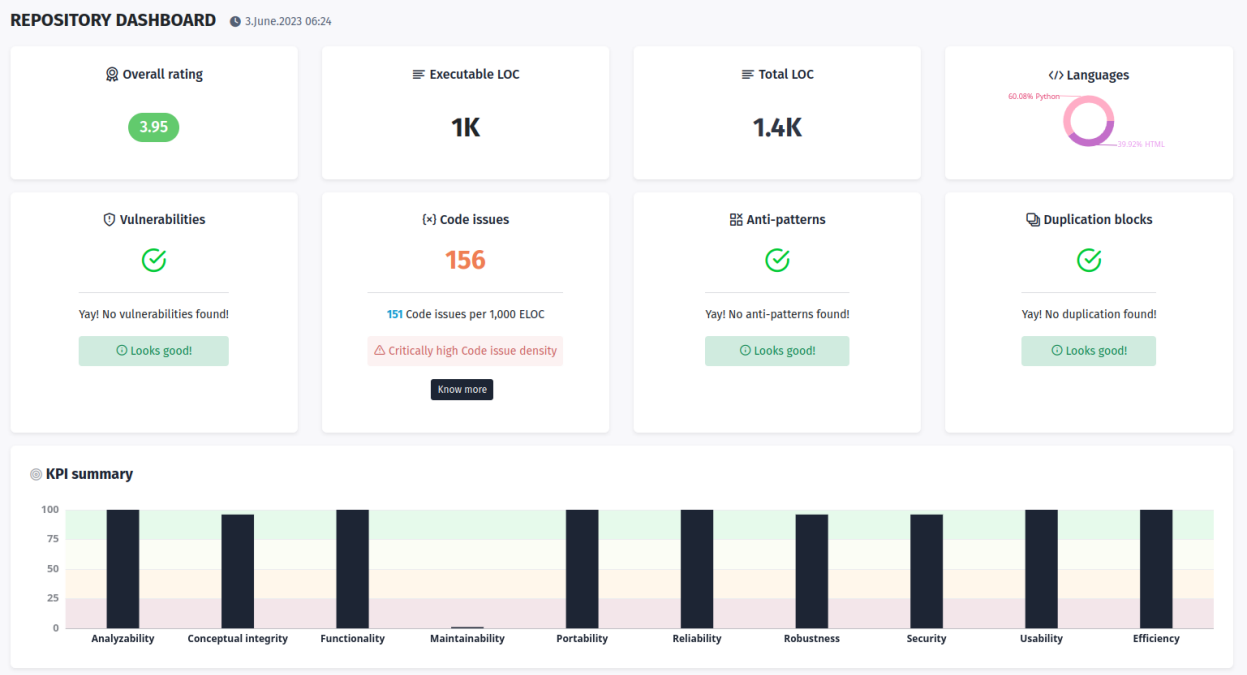


Рисунок 3.1 – Результати аналізу якості ПЗ

Як можна бачити результати загалом задовільні. Так, сервіс показує досить багато проблемних місць у коді, але вони усі з низьким пріорітетом. Через це страждає підтримуваність коду, але усі інші показники досить високі.

3.2 Опис процесів тестування

Ми виконали мануальне тестування програмного забезпечення, опис відповідних тестів наведено у таблицях 3.1 – 3.15.

Таблиця 3.1 – Тест 1.1

Тест	Реєстрація користувача
Модуль	Система автентифікації
Номер тесту	1.1
Початковий стан системи	Користувач знаходиться на сторінці реєстрації
Вхідні дані	Ім’я користувача, електронна пошта, пароль, підтвердження паролю
Опис проведення тесту	У відповідні поля вводяться: – Ім’я користувача, якого ще нема в системі. – Коректна електронна пошта, на яку ще не було зареєстровано користувача. – Пароль довжиною не менше 8 символів, що містить хоча б одну букву та не входить до часто використовуваних. – Підтвердження паролю, що співпадає з раніше введеним паролем. Після цього натискається кнопка «Register»
Очікуваний результат	Реєстрація проходить успішно, користувач додається у систему і перенаправляється на сторінку перегляду задач.
Фактичний результат	Реєстрація проходить успішно, користувач додається у систему і перенаправляється на сторінку перегляду задач.

Таблиця 3.2 – Тест 1.2

Тест	Невдала реєстрація користувача
Модуль	Система автентифікації
Номер тесту	1.2
Початковий стан системи	Користувач знаходиться на сторінці реєстрації

Продовження таблиці 3.2

Вхідні дані	Ім’я користувача, електронна пошта, пароль, підтвердження паролю
Опис проведення тесту	У відповідні поля вводяться: – Ім’я користувача, що вже є в системі. – Некоректна електронна пошта, або та, на яку вже було зареєстровано користувача. – Пароль довжиною менше 8 символів, що складається з одних цифр та входить до часто вживаних. – Підтвердження паролю, що не співпадає з раніше введеним паролем. Після цього натискається кнопка «Register»
Очікуваний результат	Реєстрація не проходить, користувач залишається на сторінці реєстрації і бачить помилку.
Фактичний результат	Реєстрація не проходить, користувач залишається на сторінці реєстрації і бачить помилку.

Таблиця 3.3 – Тест 1.3

Тест	Автентифікація користувача
Модуль	Система автентифікації
Номер тесту	1.3
Початковий стан системи	Користувач знаходиться на сторінці автентифікації
Вхідні дані	Ім’я користувача, пароль.
Опис проведення тесту	У відповідні поля вводяться: – Логін користувача, що зареєстрований в системі. – Коректний пароль для цього користувача. Після цього натискається кнопка «Login»

Продовження таблиці 3.3

Очікуваний результат	Автентифікація проходить успішно, вхід виконано, користувача перенаправлено на сторінку перегляду результатів.
Фактичний результат	Автентифікація проходить успішно, вхід виконано, користувача перенаправлено на сторінку перегляду результатів.

Таблиця 3.4 – Тест 1.4

Тест	Невдала автентифікація користувача
Модуль	Система автентифікації
Номер тесту	1.4
Початковий стан системи	Користувач знаходиться на сторінці автентифікації.
Вхідні дані	Ім’я користувача, пароль.
Опис проведення тесту	У відповідні поля вводяться: – Логін користувача, що не зареєстрований в системі. – Неправильний пароль для цього користувача. Після цього натискається кнопка «Login»
Очікуваний результат	Автентифікація не проходить, користувач залишається на сторінчі автентифікації і бачить помилку.
Фактичний результат	Автентифікація не проходить, користувач залишається на сторінчі автентифікації і бачить помилку.

Таблиця 3.5 – Тест 1.5

Тест	Вихід з системи
Модуль	Система автентифікації
Номер тесту	1.5

Продовження таблиці 3.5

Початковий стан системи	Користувач знаходиться на будь-якій сторінці.
Вхідні дані	-
Опис проведення тесту	Користувач обирає пункт «Logout» у модальному меню «Account».
Очікуваний результат	Сесію завершено, користувача розлогінено з системи, і відображено відповідне повідомлення.
Фактичний результат	Сесію завершено, користувача розлогінено з системи, і відображено відповідне повідомлення.

Таблиця 3.6 – Тест 1.6

Тест	Отримання токену для доступу до API
Модуль	Система автентифікації
Номер тесту	1.6
Початковий стан системи	Користувач знаходиться на будь-якій сторінці.
Вхідні дані	-
Опис проведення тесту	Користувач обирає пункт «Regenerate Token» у модальному меню «Account».
Очікуваний результат	Користувача перенаправляє на сторінку генерації токену, де він бачить новий токен, що відрізняється від старого.
Фактичний результат	Користувача перенаправляє на сторінку генерації токену, де він бачить новий токен, що відрізняється від старого.

Таблиця 3.7 – Тест 2.1

Тест	Класифікація тексту
Модуль	Класифікація тексту
Номер тесту	2.1

Продовження таблиці 3.8

Початковий стан системи	Автентифікований користувач знаходиться на сторінці класифікації тексту.
Вхідні дані	Текст для класифікації.
Опис проведення тесту	Користувач заповнює форму класифікації, вводячи текст або його фрагмент. Після цього він натискає кнопку «Submit».
Очікуваний результат	Запит оброблено, користувача перенаправлено на сторінку перегляду результатів, на якій видно тільки що оновлений результат виконання. Через якийсь час статус цього результату оновлюється на SUCCESS.
Фактичний результат	Запит оброблено, користувача перенаправлено на сторінку перегляду результатів, на якій видно тільки що оновлений результат виконання. Через якийсь час статус цього результату оновлюється на SUCCESS.

Таблиця 3.8 – Тест 3.1

Тест	Класифікація документу
Модуль	Класифікація документу
Номер тесту	3.1
Початковий стан системи	Автентифікований користувач знаходиться на сторінці класифікації документу.
Вхідні дані	Документ для класифікації у форматі pdf.
Опис проведення тесту	Користувач заповнює форму класифікації документу, завантажуючи документ. Після цього він натискає кнопку «Submit».
Очікуваний результат	Запит оброблено, користувача перенаправлено на сторінку перегляду результатів, на якій видно тільки що оновлений результат виконання. Через якийсь час статус цього результату оновлюється на SUCCESS.

Продовження таблиці 3.8

Фактичний результат	Запит оброблено, користувача перенаправлено на сторінку перегляду результатів, на якій видно тільки що оновлений результат виконання. Через якийсь час статус цього результату оновлюється на SUCCESS.

Таблиця 3.9 – Тест 3.2

Тест	Невдала класифікація документу
Модуль	Класифікація документу
Номер тесту	3.2
Початковий стан системи	Автентифікований користувач знаходиться на сторінці класифікації документу.
Вхідні дані	Пошкоджений документ, або документ у непідтримуваному форматі.
Опис проведення тесту	Користувач заповнює форму класифікації документу, завантажуючи документ. Після цього він натискає кнопку «Submit».
Очікуваний результат	Запит оброблено, користувача перенаправлено на сторінку перегляду результатів, на якій видно тільки що оновлений результат виконання. Через якийсь час статус цього результату оновлюється на FAILURE.
Фактичний результат	Запит оброблено, користувача перенаправлено на сторінку перегляду результатів, на якій видно тільки що оновлений результат виконання. Через якийсь час статус цього результату оновлюється на FAILURE.

Таблиця 3.10 – Тест 4.1

Тест	Обробка декількох запитів
Модуль	Черга задач, Результати класифікації

Продовження таблиці 3.10

Номер тесту	4.1
Початковий стан системи	Користувач знаходиться на сторінці класифікації тексту або документу.
Вхідні дані	Тексти та документи для класифікації.
Опис проведення тесту	Користувач за короткий проміжок часу надсилає декілька (5-10) запитів класифікації. Після цього він переходить на сторінку перегляду результатів і спостерігає за статусами створених задач.
Очікуваний результат	Усі новостворені задачі спочатку перебувають у стані PENDING. Через якийсь час одна з задач переходить у стан STARTED, ще через якийсь час ця задача переходить у стан SUCCESS або FAILURE. Таким чином оброблюються усі задачі.
Фактичний результат	Усі новостворені задачі спочатку перебувають у стані PENDING. Через якийсь час одна з задач переходить у стан STARTED, ще через якийсь час ця задача переходить у стан SUCCESS або FAILURE. Таким чином оброблюються усі задачі.

Таблиця 3.11 – Тест 6.1

Тест	[API] Класифікація тексту
Модуль	Програмний інтерфейс
Номер тесту	6.1
Початковий стан системи	-
Вхідні дані	Текст для класифікації
Опис проведення тесту	Користувач за допомогою REST клієнту надсилає запит класифікації тексту на відповідний endpoint.

Продовження таблиці 3.11

Очікуваний результат	Користувач отримує відповідь з ідентифікатором задачі.
Фактичний результат	Користувач отримує відповідь з ідентифікатором задачі.

Таблиця 3.12 – Тест 6.2

Тест	[API] Класифікація документу
Модуль	Програмний інтерфейс
Номер тесту	6.2
Початковий стан системи	-
Вхідні дані	Документ для класифікації
Опис проведення тесту	Користувач за допомогою REST клієнту надсилає запит класифікації документу на відповідний endpoint.
Очікуваний результат	Користувач отримує відповідь з ідентифікатором задачі.
Фактичний результат	Користувач отримує відповідь з ідентифікатором задачі.

Таблиця 3.13 – Тест 6.3

Тест	[API] Отримання результатів
Модуль	Програмний інтерфейс
Номер тесту	6.3
Початковий стан системи	Користувач вже користувався класифікатором.
Вхідні дані	-
Опис проведення тесту	Користувач за допомогою REST клієнту надсилає запит отримання результату на відповідний endpoint.

Продовження таблиці 3.13

Очікуваний результат	Користувач отримує всі результати, пов’язані з його обліковим записом.
Фактичний результат	Користувач отримує всі результати, пов’язані з його обліковим записом.

Таблиця 3.14 – Тест 6.3

Тест	[API] Отримання конкретного результату
Модуль	Програмний інтерфейс
Номер тесту	6.3
Початковий стан системи	Користувач вже користувався класифікатором.
Вхідні дані	-
Опис проведення тесту	Користувач за допомогою REST клієнту надсилає запит отримання результату на відповідний endpoint, вказавши ідентифікатор задачі, що пов’язана з його обліковим записом.
Очікуваний результат	Користувач отримує результат класифікації, ідентифікатор якого він вказав.
Фактичний результат	Користувач отримує результат класифікації, ідентифікатор якого він вказав.

Таблиця 3.15 – Тест 6.4

Тест	[API] Невдале отримання конкретного результату
Модуль	Програмний інтерфейс
Номер тесту	6.4
Початковий стан системи	Користувач вже користувався класифікатором.
Вхідні дані	-

Продовження таблиці 3.1

Опис проведення тесту	Користувач за допомогою REST клієнту надсилає запит отримання результату на відповідний endpoint, вказавши неіснуючий ідентифікатор задачі, або ідентифікатор задачі, що не пов’язана з його обліковим записом.
Очікуваний результат	Користувач отримує відмову.
Фактичний результат	Користувач отримує відмову.

3.3 Опис контрольного прикладу

Опишемо контрольний приклад, звертаючи увагу на моменти, описані у попередньому розділі.

Зайдемо на сайт і оберемо опцію реєстрації. Нас перенаправляє на сторінку, як на рис. 3.2. Введемо правильні дані, окрім пошти, і натиснемо «Register». На рисунку цього не видно, але система покаже помилку.

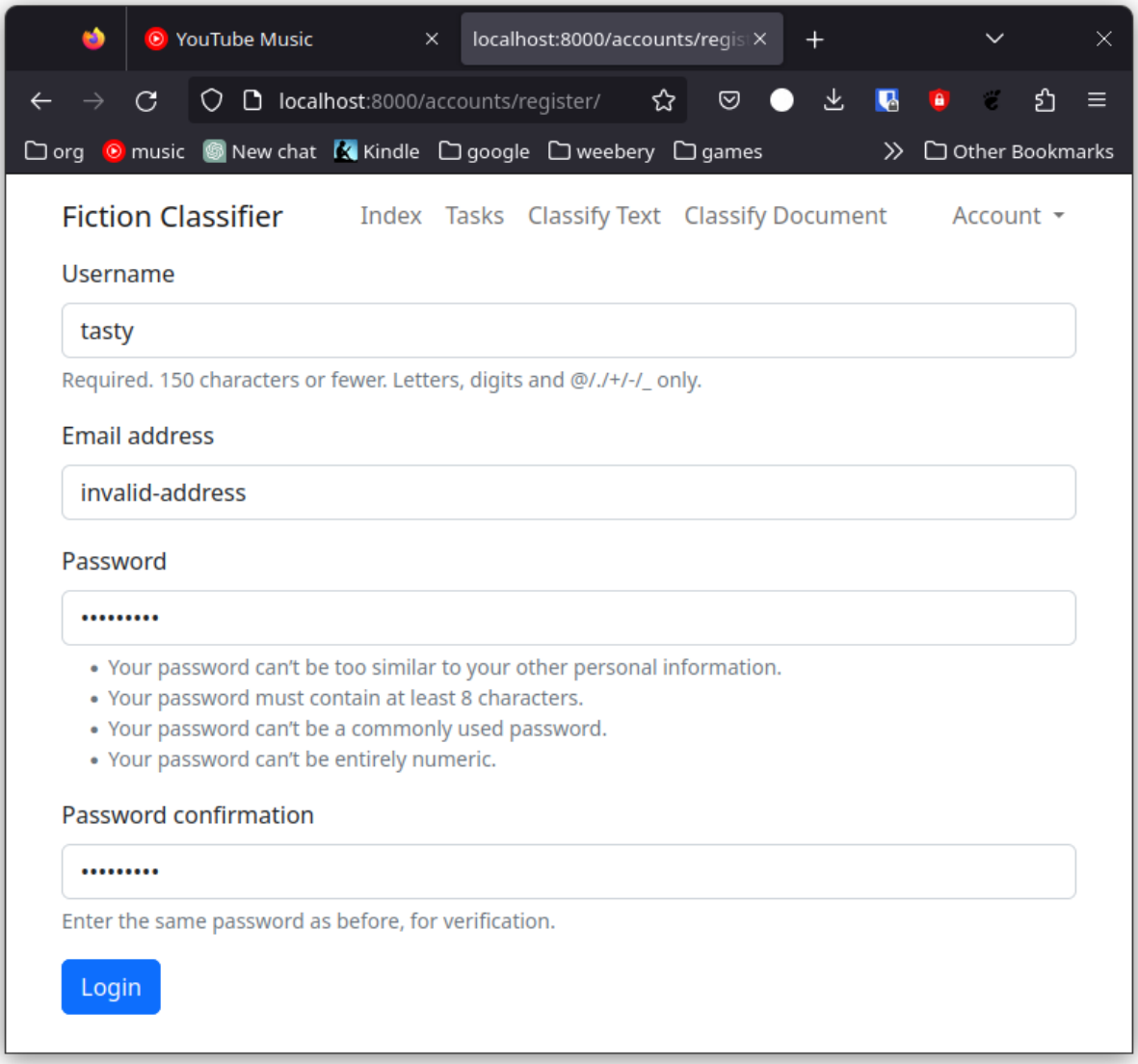


Рисунок 3.2 – Невдала реєстрація

Виправимо адресу електронної пошти і знову натиснемо «Register». В цей раз реєстрація успішна (рис. 3.3).

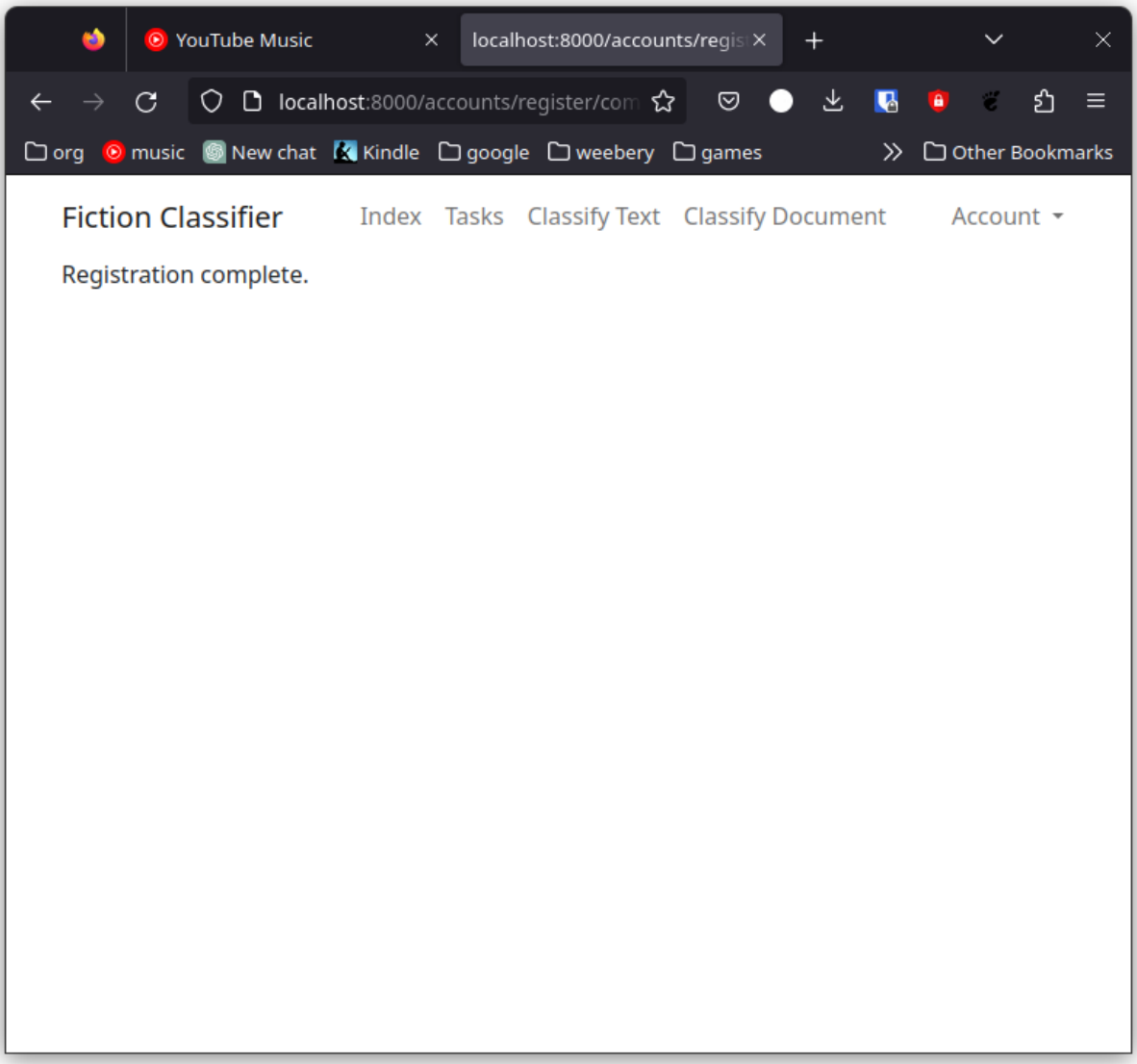


Рисунок 3.3 – Вдала реєстрація

Одразу перевіримо, що ми можемо вийти з облікового запису. Для цього відкриємо модальне меню «Account» і оберемо опцію «Logout» (рис. 3.4).

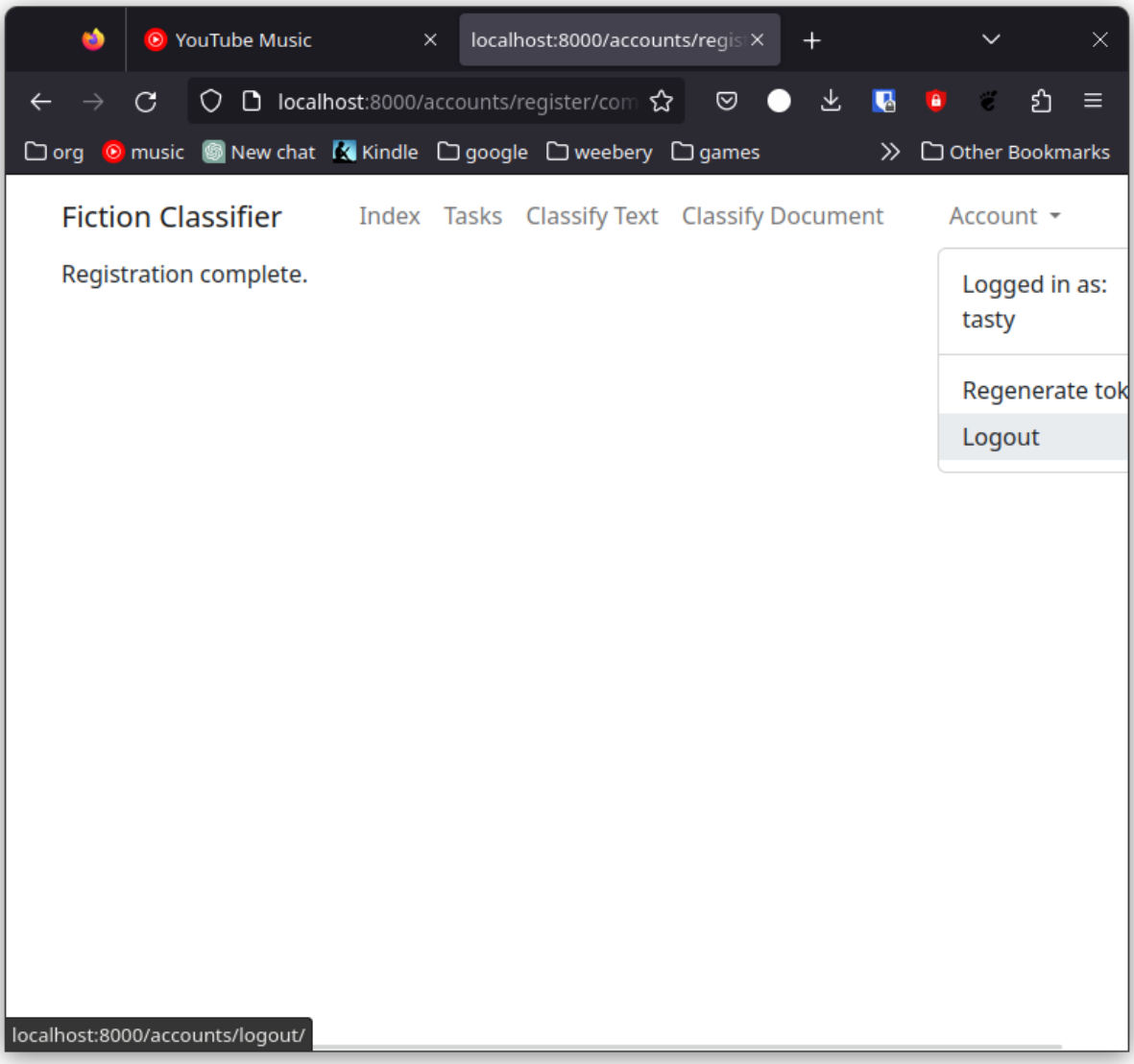


Рисунок 3.4 – Вихід з облікового запису

Нас успішно розлогінить (рис. 3.5).

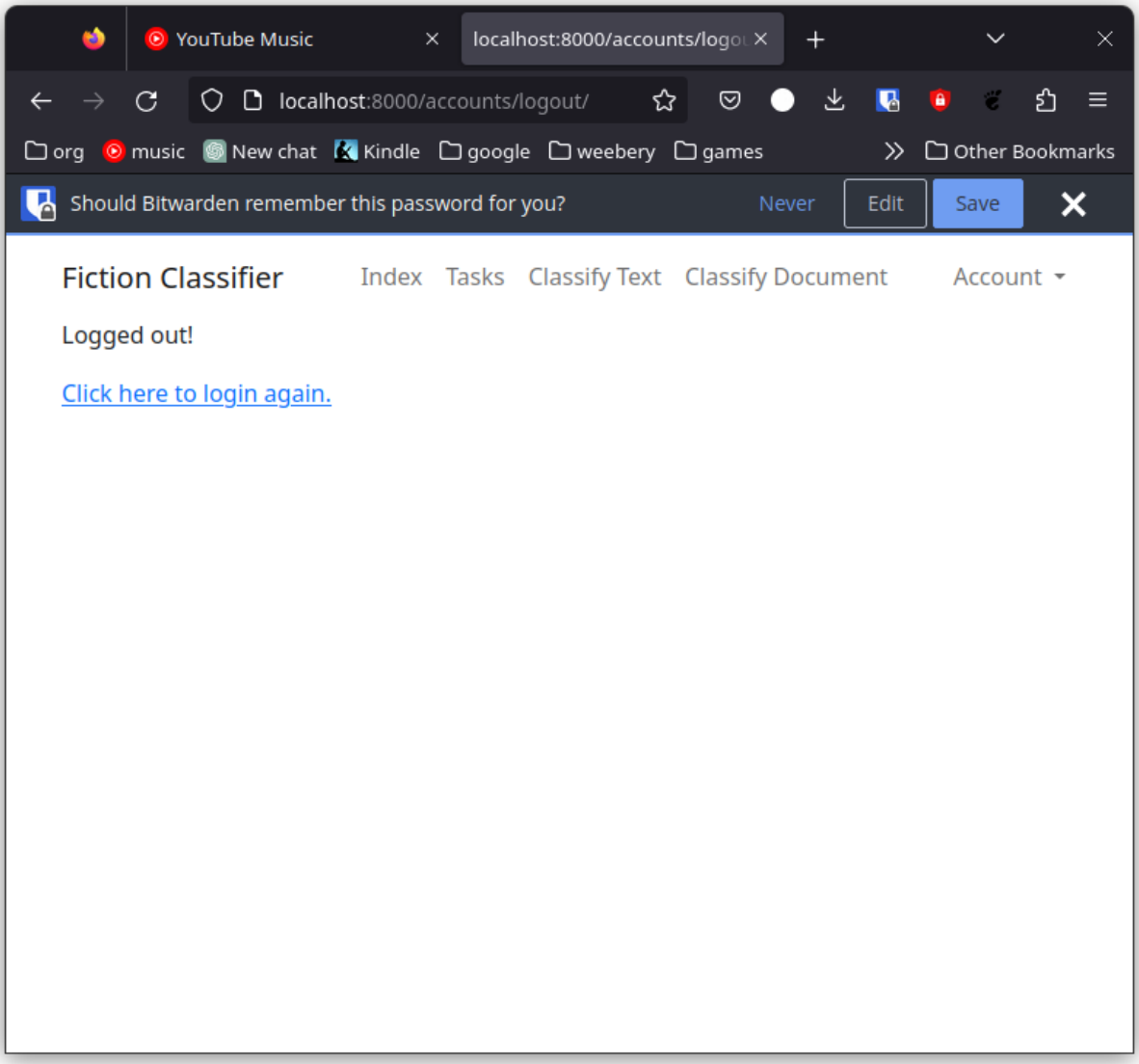


Рисунок 3.5 – Успішний вихід з облікового запису

Але нам все ще треба протестувати сервіс, тому залогінимось назад. Для цього можна натиснути посилання під повідомленням «Logged out!», або ж вибрати опцію «Login» у модальному меню аккаунту. Введемо ім’я користувача, якого ми створили раніше, і неправильний пароль. На скріні знову цього не видно, але система повідомить, що пароль/ім’я неправильні (рис 3.6).

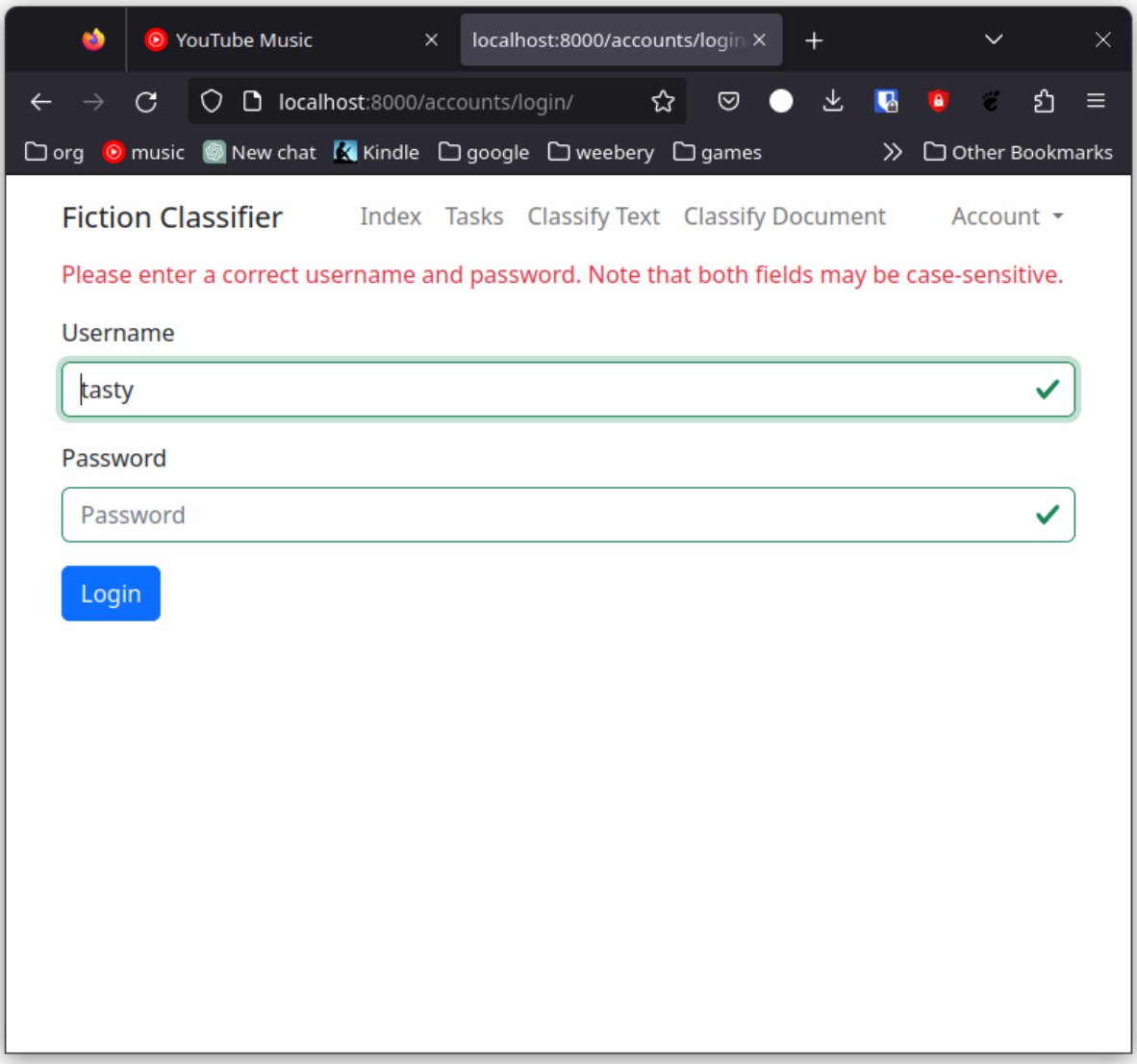


Рисунок 3.6 – Невдала автентифікація

Введемо правильний пароль і знову натиснемо «Login». В цей раз автентифікація успішна, і нас пускає на сторінку перегляду результатів. Поки що даний користувач не користувався класифікатором, тому результатів немає (рис. 3.7).

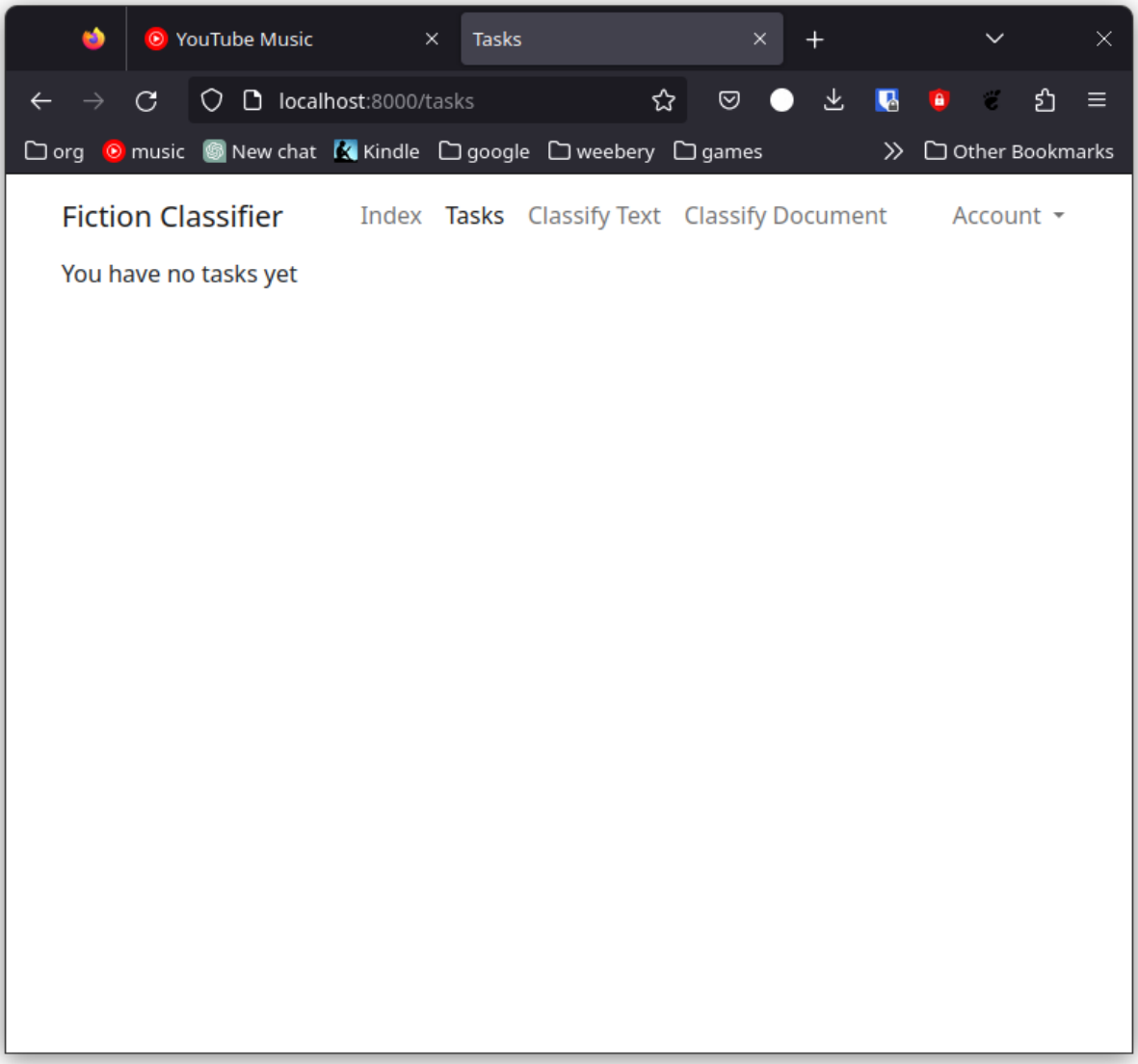


Рисунок 3.7 – Вдала автентифікація і сторінка перегляду результатів

Перейдемо на сторінку класифікації тексту, заповнимо потрібні поля, і натиснемо на кнопку «Submit» (рис. 3.8).

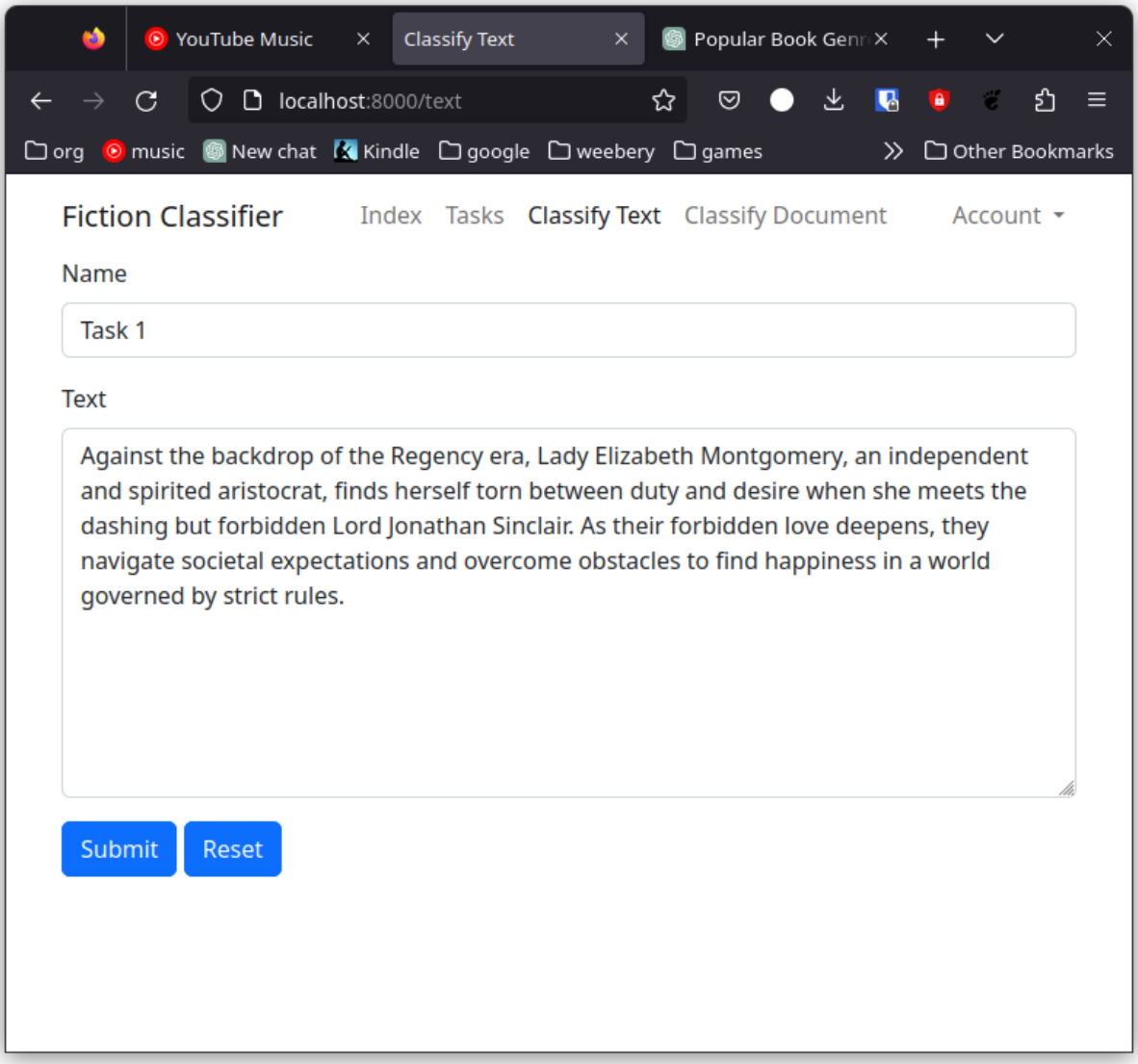


Рисунок 3.8 – Сторінка класифікації тексту

Нас перекине назад на сторінку перегляду результатів, де можна побачити новостворену задачу, яка все ще перебуває у статусі PENDING, тобто вона ще не надійшла в обробку (рис. 3.9).

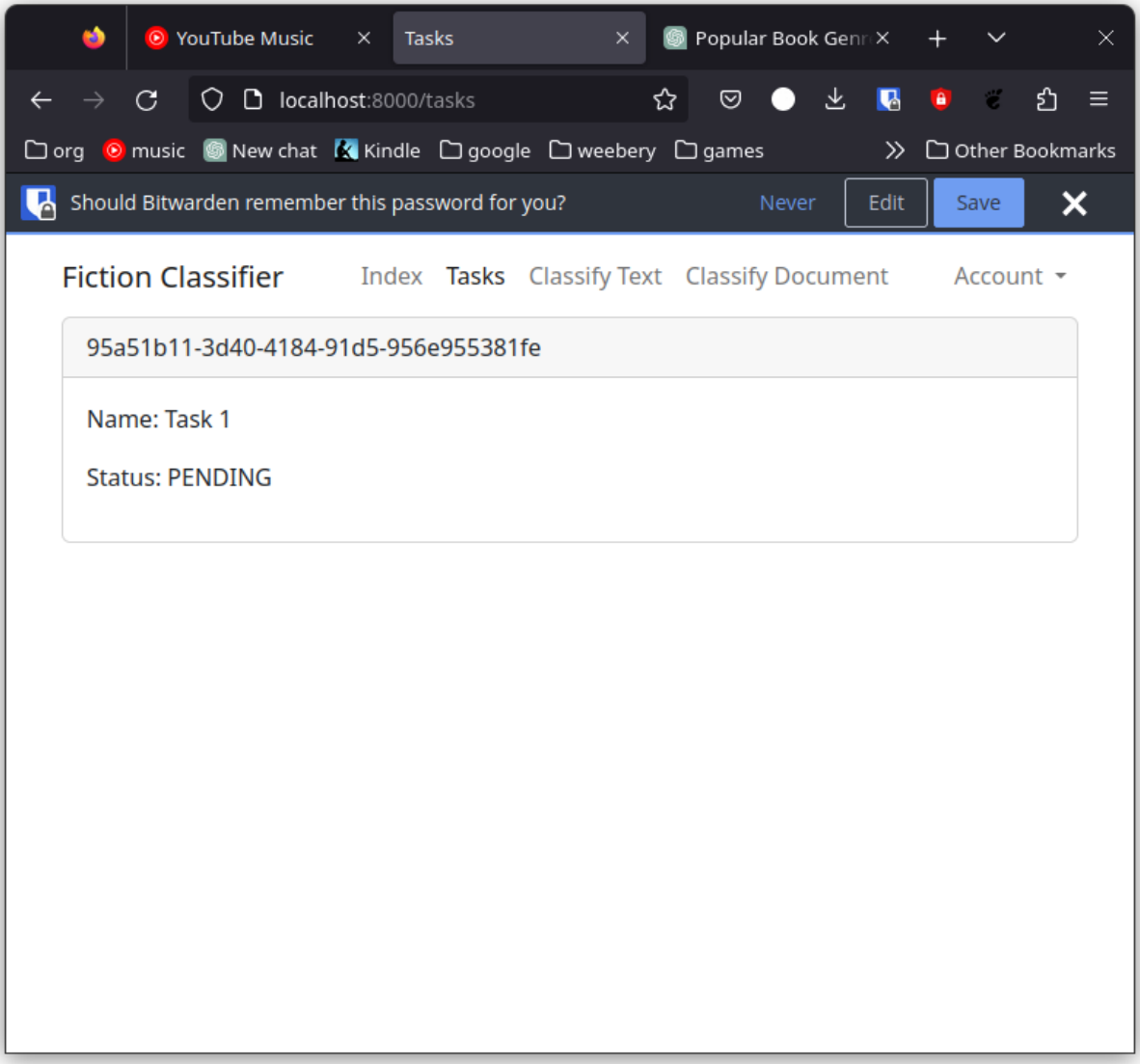


Рисунок 3.9 – Сторінка перегляду результатів

Поки ця задача обробляється, перейдемо на сторінку класифікації документів і завантажимо файл у форматі pdf. Я обрав «Дракулу» Брема Стокера (рис. 3.10).

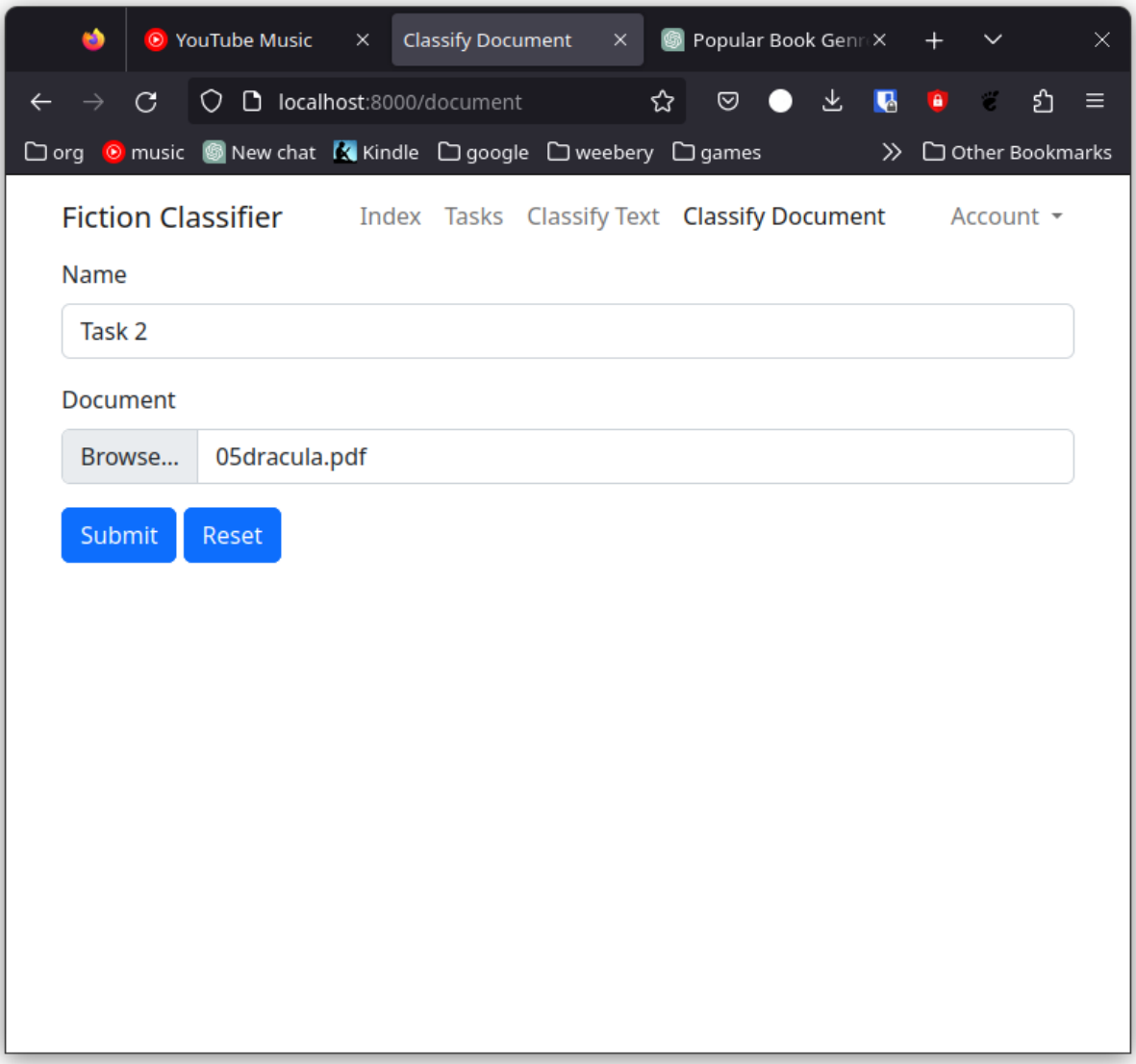


Рисунок 3.10 – Сторінка класифікації документу

Натискаємо «Submit» і бачимо, що нас знову перекинуло на сторінку перегляду результатів. Тепер тут дві задачі, і одна з них уже завершена. Можна бачити, що текст класифікувався як романс (рис. 3.11).

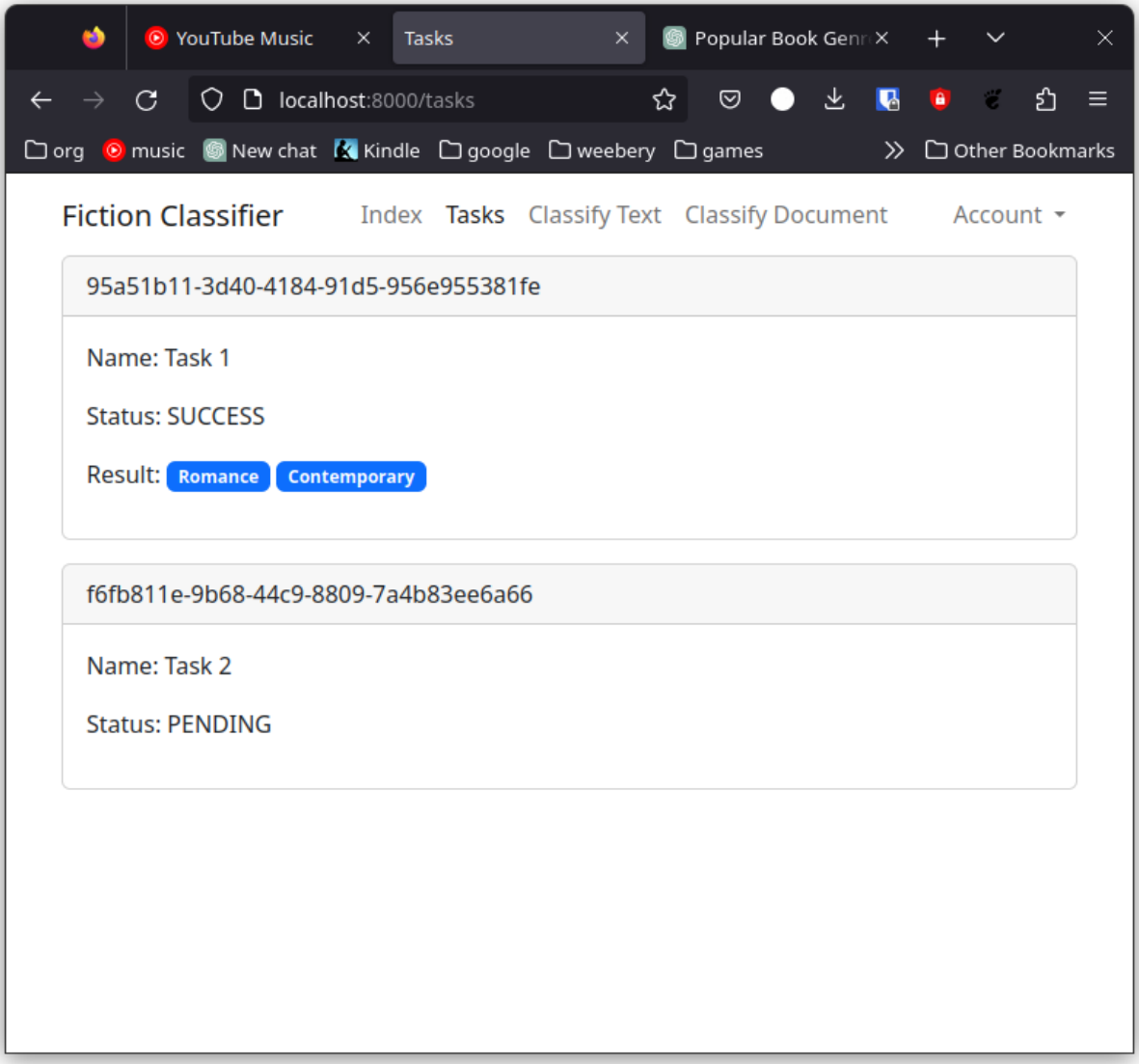


Рисунок 3.11 – Сторінка перегляду результатів з однією виконаною задачею

Трішки почекаємо, і перезавантажимо сторінку. Тепер обидві задачі мають статус SUCCESS, і наш документ класифікувався як «Fantasy», «Horror», «Paranormal», «Gothic», «Supernatural» і «Vampires» (рис. 3.12).

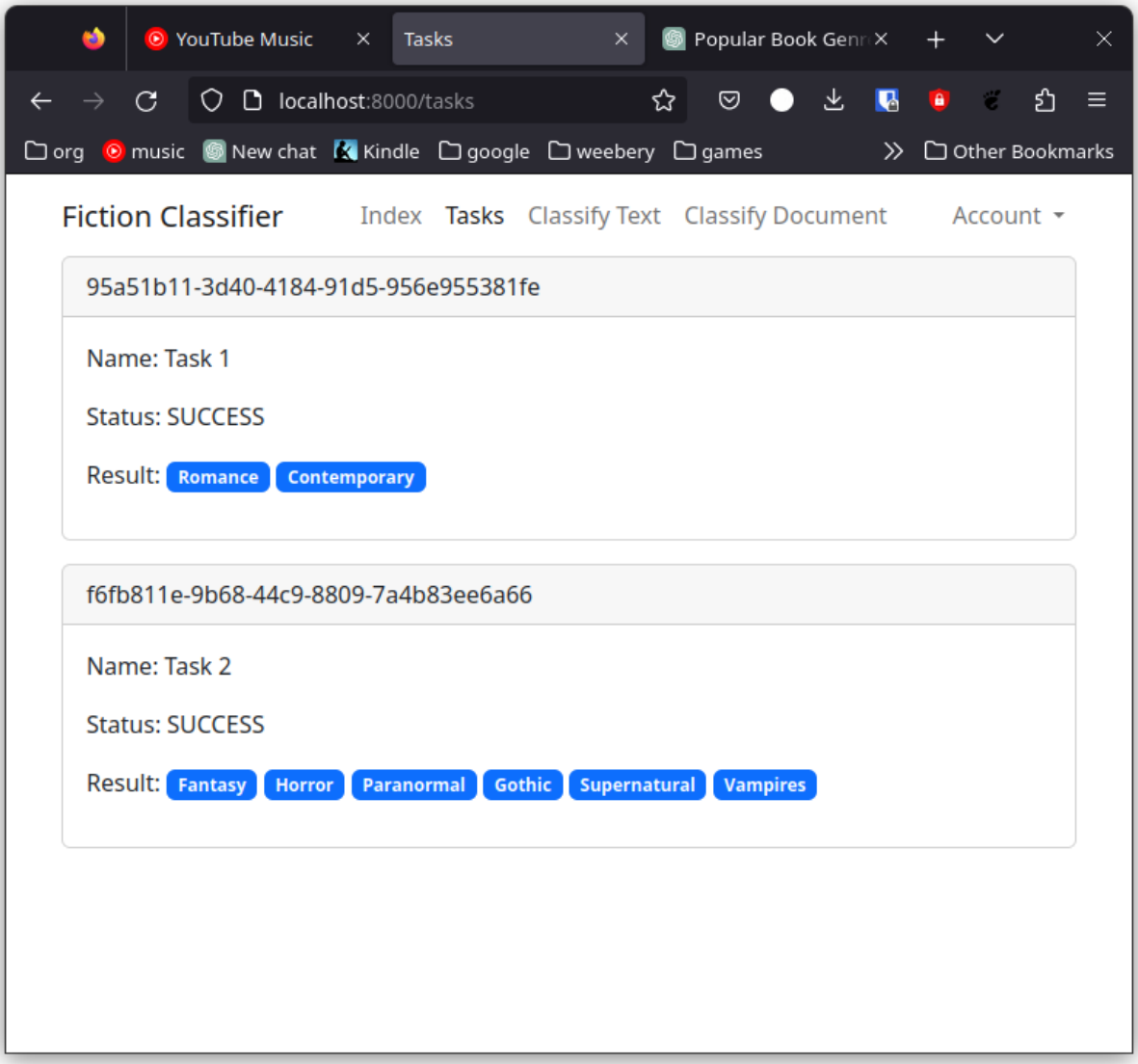


Рисунок 3.12 – Сторінка перегляду результатів з двома виконаними задачами

Залишилось перевірити API, а для цього нам потрібен токен. Оберемо опцію генерації токену у модальному вікні «Account» (рис. 3.13).

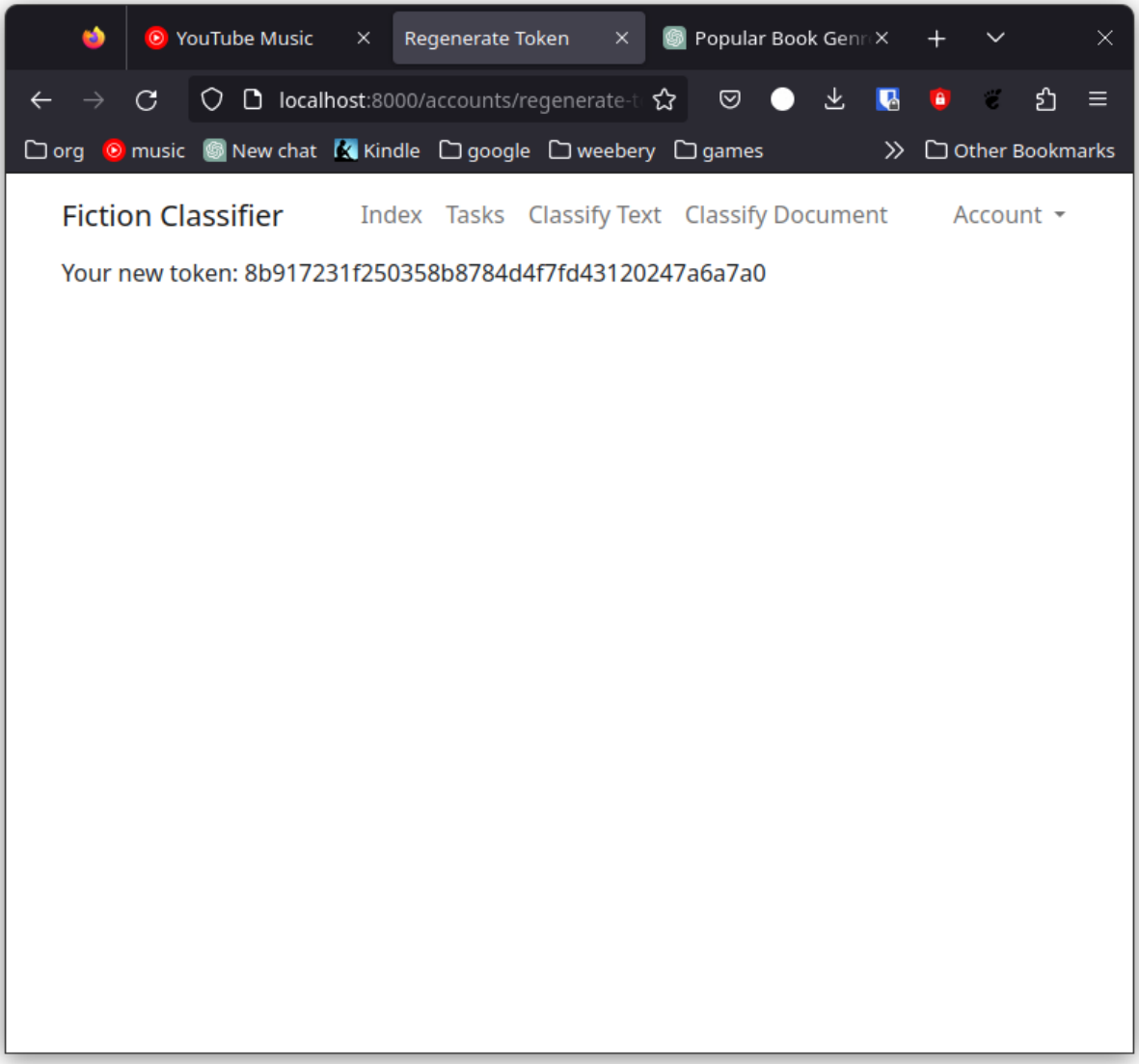


Рисунок 3.13 – Генерація токену доступу до API

Ми отримали токен, а це значить, що ми можемо почати користуватись API. У якості клієнта я буду використовувати curl.

Зробимо запит GET на endpoint api/tasks, вказавши токен у заголовках. Як можна бачити, сервер повернув усі задачі, пов’язані з цим обліковим записом, у форматі json (рис. 3.14).

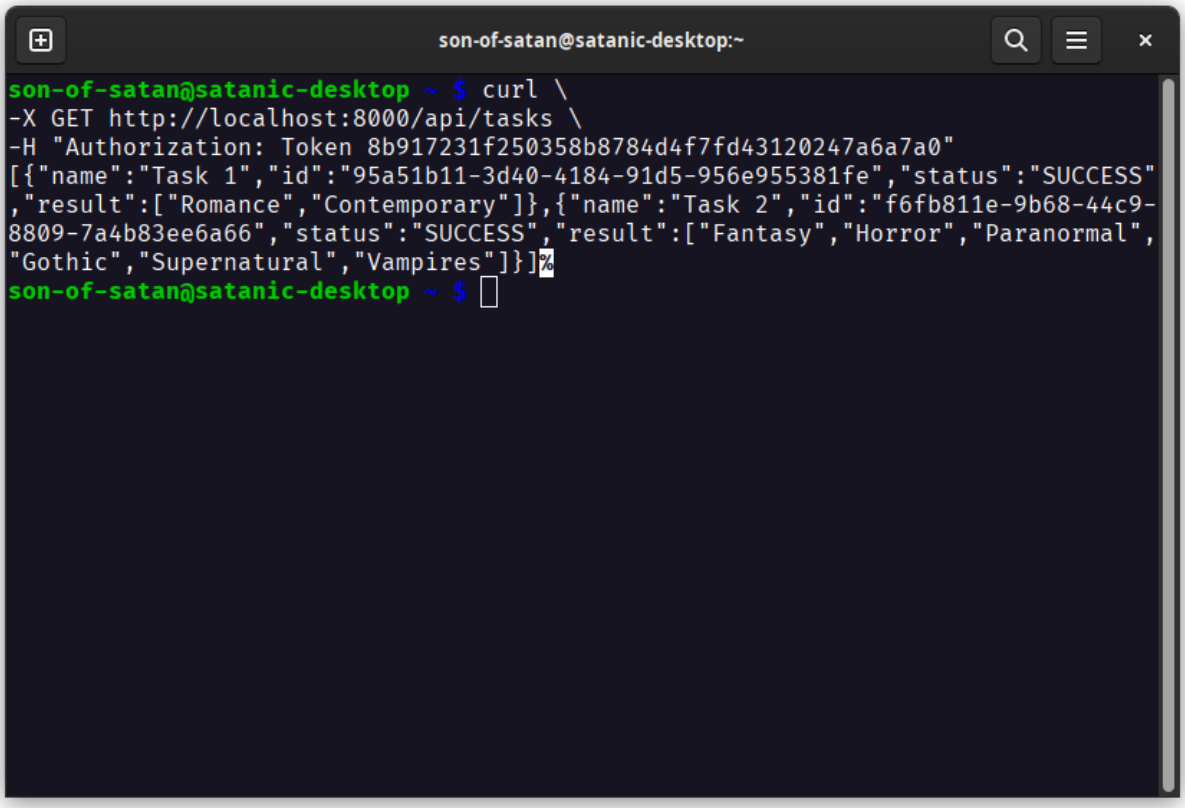


Рисунок 3.14 – Отримання результатів класифікації за допомогою API

Зробимо запит POST на endpoint api/text. Сервер повертає відповідь, що містить ідентифікатор задачі. Спробуємо отримати задачу, якій належить цей ідентифікатор. Для цього знову зробимо запит GET на endpoint api/tasks, але тепер вкажемо ідентифікатор. Як можна бачити, сервер повернув уже оброблену задачу. Наш текст класифікувався як «Пригоди» (рис. 3.15).

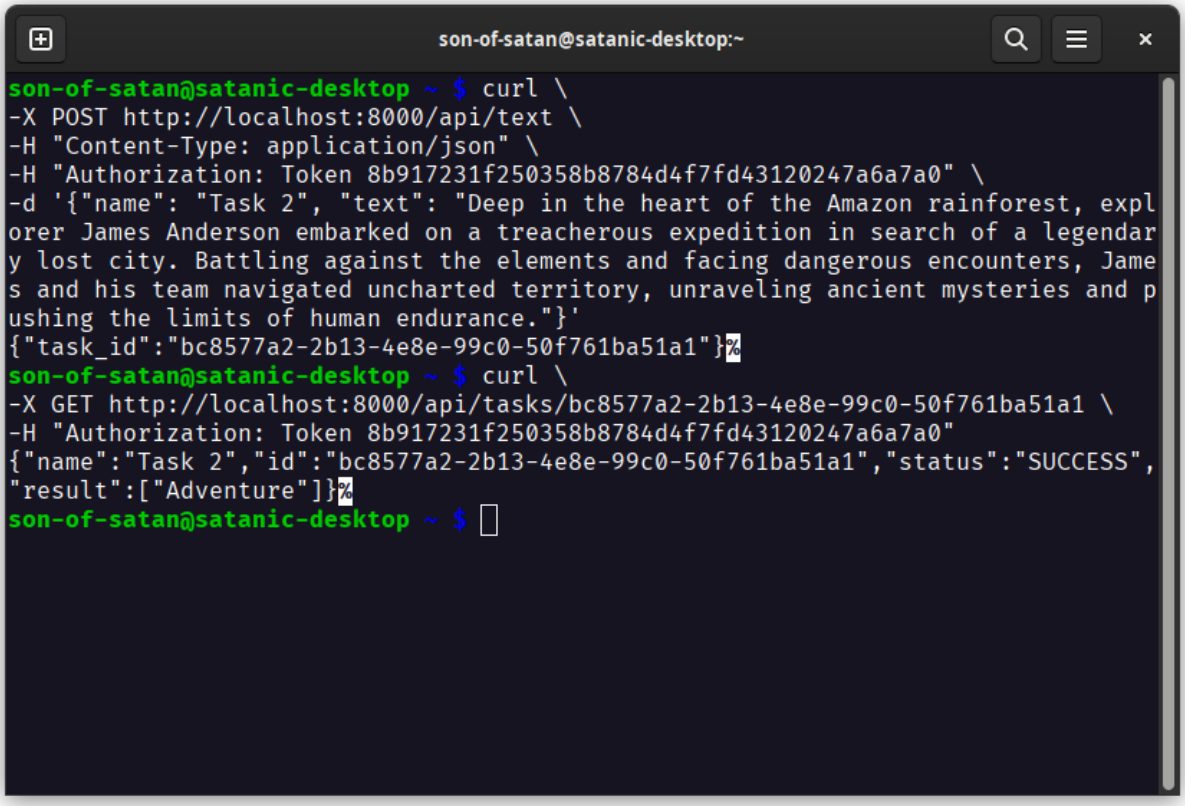


Рисунок 3.15 – Класифікація тексту за допомогою API

Висновки до розділу

У даному розділі ми провели аналіз якості програмного забезпечення за допомогою сервісу embold, і визначили, що розроблене програмне забезпечення хоч і не ідеальне, але має досить високу якість за більшістю показників.

Після цього ми описли процеси тестування і провели перевірку виконання за допомогою мануального тестування. У процесі виявили декілька помилок у роботі програми, які врешті-решт були виправлені.

У кінці ми описали контрольний приклад і основні моменти роботи з сервісом класифікації.

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

Наразі підтримується лише локальне розгортання програмного забезпечення за допомогою Docker і Docker Compose.

Для цього потрібно:

– завантажити код з гітхабу проєкту на сторінці релізів;
– зайти у директорію service;
– виконати команду docker compose build;
– виконати команду docker compose up -d.

4.2 Підтримка програмного забезпечення

Підтримка програмного забезпечення і розповсюдження нових версій виконуватиметься через гітхаб. Для створення нового релізу використовується функціонал гітхабу. Нові релізи з’являтимуться на сторінці релізів проєкту (рис. 4.1).

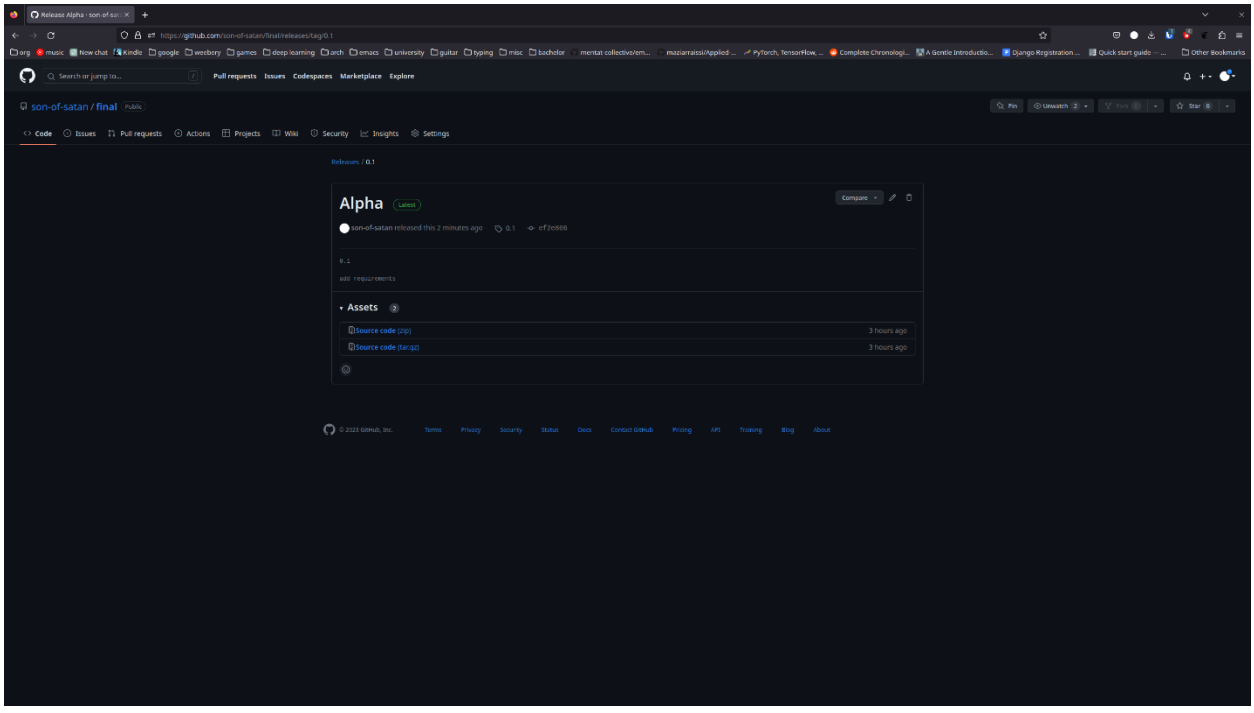


Рисунок 4.1 – Сторінка релізів проєкту

Висновки до розділу

У даному розділі ми розглянули процеси розгортання та підтримки програмного забезпечення.

ПЗ розгортається за допомогою Docker на будь-якому хості, нові версії доступні у релізах проєкту на гітхабі.

ВИСНОВКИ

В процесі виконання дипломного проєкту ми розглянули низку методів обробки природніх мов. Найбільше уваги приділили використанню моделей на основі механізму уваги, як от BERT, RoBERTa, TransformerXL та Longformer. Після цього ми зібрали датасет творів художньої літератури та навчили модель, що може з досить непоганою точністю визначати, до яких жанрів можна віднести вхідний текст.

У результаті виконання дипломного проєкту ми розробили сервіс класифікації художніх творів, що дозволяє користувачам класифікувати текст і документи у форматі pdf за жанрами. Таким чином, ми виконали усі поставлені задачі та досягли мети – автоматизували класифікацію художніх творів. Сервіс має як веб інтерфейс, так і API, яке дозволяє користувачам проводити класифікацію у автоматичному режимі.

В якості середовища розробки ми використали Emacs та Jupyter Notebook. Розробка велась на мові Python.

Після реалізації сервісу провели ручне тестування, у результаті якого виявили та усунули декілька недоліків.

Таким чином програмне забезпечення можна використати у ситуаціях, коли десь (наприклад, у архіві електронної бібліотеки) є деяка кількість некатегоризованих художніх творів, і потрібно спростити їх пошук.

Програмне забезпечення може бути покращено додаванням функціоналу обробки документів інших форматів, а не тільки формату pdf.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Офіційна документація Tensorflow [Електронний ресурс] – Режим доступу до ресурсу: https://www.tensorflow.org/api_docs.
2) Офіційна документація Pytorch [Електронний ресурс] – Режим доступу до ресурсу: https://pytorch.org/docs/stable/index.html.
3) Офіційна документація MxNet [Електронний ресурс] – Режим доступу до ресурсу: https://mxnet.apache.org/versions/1.8.0/api.
4) Офіційна документація Transformers [Електронний ресурс] – Режим доступу до ресурсу: https://huggingface.co/docs/transformers/index.
5) Офіційна документація Django [Електронний ресурс] – Режим доступу до ресурсу: https://docs.djangoproject.com/en/4.2/.
6) Офіційна документація Flask [Електронний ресурс] – Режим доступу до ресурсу: https://flask.palletsprojects.com/en/2.3.x/.
7) Офіційна документація Bottle [Електронний ресурс] – Режим доступу до ресурсу: https://bottlepy.org/docs/dev/.
8) Iz Beltagy. Longformer: The Long-Document Transformer [Електронний ресурс] / Iz Beltagy, Matthew E. Peters, Arman Cohan // Arxiv. – 2020. – Режим доступу до ресурсу: https://arxiv.org/abs/2004.05150.
9) BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding [Електронний ресурс] / Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova // Arxiv. – 2019. – Режим доступу до ресурсу: https://arxiv.org/abs/1810.04805.
10) Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context [Електронний ресурс] / Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell // Arxiv. – 2019. – Режим доступу до ресурсу: https://arxiv.org/abs/1901.02860.
11) Big Bird: Transformers for Longer Sequences [Електронний ресурс] / Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie // Arxiv. – 2021. – Режим доступу до ресурсу: https://arxiv.org/abs/2007.14062.

ДОДАТОК А ЗВІТ ПОДІБНОСТІ

UNICHECK

Ім'я користувача:
Лісовиченко Олег Іванович

ID перевірки:
1015409927

Дата перевірки:
04.06.2023 07:41:28 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
04.06.2023 07:46:53 EEST

ID користувача:
76913

Назва документа: IT-92_Татарин_ПЗ

Кількість сторінок: 66 **Кількість слів:** 7499 **Кількість символів:** 59734 **Розмір файлу:** 1.74 MB **ID файлу:** 1015073292

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

10.7%
Схожість

Найбільша схожість: 4.39% з джерелом з Бібліотеки (ID файлу: 1015067687)

5.35% Джерела з Інтернету	186	Сторінка 68
10.7% Джерела з Бібліотеки	331	Сторінка 71

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	6
Підозріле форматування	21 сторінка

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

Веб-сервіс для автоматичної категоризації художніх творів

Текст програми

КПІ.ІТ-9224.045440.03.12

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олесь Ковтунець

Нормоконтроль:
_____ Ірина ВІТКОВСЬКА

Виконавець:
_____ Микола Татарин

Київ – 2023

Файл service/urls.py

```
from django.contrib import admin
from django.urls import include, path

from classifier.views import RegenerateTokenView


urlpatterns = [
    path("accounts/", include("django_registration.backends.one_step.urls")),
    path("accounts/", include("django.contrib.auth.urls")),
    path(
        "accounts/regenerate-token",
        RegenerateTokenView.as_view(),
        name="regenerate-token",
    ),
    path("admin/", admin.site.urls),
    path("", include("classifier.urls")),
    path("api/", include("classifier_api.urls")),
]
```

Файл common/utilities.py

```
from django.contrib.auth.models import User
from django.core.files.uploadedfile import UploadedFile
from celery.utils.serialization import base64encode

from django_celery_results.models import TaskResult

import json

from rest_framework import status
from .tasks import classify_text, classify_document
from .models import Task


def handle_text(name: str, text: str, user: User):
    truncated_text = text if len(text) < 64 else f"{text[61]}..."
    name = name if name else truncated_text

    result = classify_text.apply_async(kwargs={"text": text})
    result = TaskResult.objects.get(task_id=result.task_id)

    task = Task(name=name, result=result, owner=user)
    task.save()

    return task.result.task_id


def handle_document(name: str, document: UploadedFile, user: User):
    name = name if name else document.name
    contents = base64encode(document.read()).decode()

    result = classify_document.apply_async(kwargs={"contents": contents})
    result = TaskResult.objects.get(task_id=result.task_id)

    task = Task(name=name, result=result, owner=user)
    task.save()

    return task.result.task_id


def get_tasks(user: User):
    tasks = Task.objects.filter(owner=user)
    ret = []
    for task in tasks:
        name = task.name
        id = task.result.task_id
        status = task.result.status
        result = task.result.result
        genres = json.loads(result) if result else None

        ret.append(
            {
                "name": name,
                "id": id,
                "status": status,
                "result": genres,
            }
        )
```

```
    return ret


def get_task(user: User, task_id):
    result = TaskResult.objects.get(task_id=task_id)
    task = Task.objects.get(owner=user, result=result)

    name = task.name
    id = task.result.task_id
    status = task.result.status
    result = task.result.result
    genres = json.loads(result) if result else None

    ret = {
        "name": name,
        "id": id,
        "status": status,
        "result": genres,
    }

    return ret
```

Файл common/classifier.py

```
import torch
import torchtext
from transformers import LongformerTokenizer, LongformerConfig, LongformerModel


class Model(torch.nn.Module):
    def __init__(self, num_labels):
        super().__init__()
        self.transformer = LongformerModel(
            LongformerConfig.from_pretrained("allenai/longformer-base-4096")
        )
        self.output = torch.nn.Linear(in_features=768, out_features=num_labels)
        self.sigmoid = torch.nn.Sigmoid()

    def forward(self, input_ids=None, attention_mask=None):
        x = self.transformer(input_ids=input_ids, attention_mask=attention_mask)[
            "pooler_output"
        ]  # "pooler_output", "last_hidden_state"
        x = self.output(x)
        x = self.sigmoid(x)
        return x


class Classifier:
    def __init__(self, model_path, tokenizer_path, vocab_path, device=None):
        self.vocab = torch.load(vocab_path)
        self.tokenizer = LongformerTokenizer.from_pretrained(
            "allenai/longformer-base-4096"
        )
        self.model = Model(len(self.vocab))

        self.model.load_state_dict(torch.load(model_path))
        self.model.eval()

        if device:
            pass
        elif torch.cuda.is_available():
            device = "cuda"
        else:
            device = "cpu"

        self.device = torch.device(device)

    def classify(self, texts):
        inputs = self.tokenizer(
            texts, padding="max_length", truncation=True, return_tensors="pt"
        )

        with torch.no_grad():
            probabilities = self.model(**inputs)

        print(probabilities)
        labels = self._decode_labels((probabilities > 0.4).to(int))

        return labels

    def _decode_labels(self, x):
```

```
        x = [torch.flatten(row.nonzero()).tolist() for row in x]
        x = [self.vocab.lookup_tokens(row) for row in x]
        return x
```

Файл common/tasks.py

```
import tempfile
import base64

from celery import states, shared_task
from celery.signals import before_task_publish
from django_celery_results.models import TaskResult

from .classifier import Classifier
from .document_converter import DocumentConverter

classifier = Classifier("./data/model.pth", None, "./data/vocab.pth")
document_converter = DocumentConverter()


@shared_task()
def classify_text(text=None):
    genres = classifier.classify(text)[0]
    return genres


@shared_task()
def classify_document(contents=None):
    temp = tempfile.NamedTemporaryFile()
    temp.write(base64.b64decode(contents))
    temp.flush()

    text = document_converter.convert(temp.name)
    genres = classifier.classify(text)[0]
    return genres


@before_task_publish.connect
def create_task_result_on_publish(sender=None, headers=None, body=None, **kwargs):
    if "task" not in headers:
        return

    TaskResult.objects.store_result(
        "application/json",
        "utf-8",
        headers["id"],
        None,
        states.PENDING,
        task_name=headers["task"],
        task_args=headers["argsrepr"],
        task_kwargs=headers["kwargsrepr"],
    )
```

Файл common/admin.py

```
from django.contrib import admin
from .models import Task

admin.site.register(Task)
```

Файл common/serializers.py

```
from rest_framework import serializers
from .models import Task


class TaskSerializer(serializers.ModelSerializer):
    class Meta:
        model = Task
        fields = ["name", "task_id"]
```

Файл common/models.py

```
from django.db import models
from django.contrib.auth.models import User
from django_celery_results.models import TaskResult


class Task(models.Model):
    name = models.CharField(max_length=256)
    result = models.OneToOneField(to=TaskResult, on_delete=models.CASCADE)
    owner = models.ForeignKey(to=User, on_delete=models.CASCADE)
```

Файл common/document_converter.py

```
import pandoc
```

```
from pdfminer.high_level import extract_text, extract_text_to_fp
import magic
import io
import re


class DocumentConverter:
    def __init__(self):
        self.converters = {
            "application/pdf": self._convert_pdf,
            "application/epub+zip": self._convert_epub,
        }

    def convert(self, filename):
        document_type = magic.from_file(filename, mime=True)
        return self.converters[document_type](filename)

    def _convert_pdf(self, filename):
        text = extract_text(filename)
        text = re.sub("\s+", " ", text)
        text.strip()
        return text

    def _convert_epub(self, filename):
        doc = pandoc.read(source=filename, format="epub")
        text = pandoc.write(doc, format="plain")
        return text


document_converter = DocumentConverter()
```

Файл common/templates/common/base-page.html

```
{% extends 'django_bootstrap5/bootstrap5.html' %}

{% load django_bootstrap5 %}

{% block bootstrap5_title %}{% block title %}{% endblock %}{% endblock %}

{% block bootstrap5_content %}
{% with request.resolver_match.url_name as url_name %}
<nav class="navbar navbar-expand">
  <div class="container">
    <a class="navbar-brand">Fiction Classifier</a>
    <ul class="navbar-nav">
      <li class="nav-item">
        <a class="nav-link {% if url_name == 'index' %} active {% endif %}" href="{% url
'classifier:index' %}">
          Index
        </a>
      </li>
      <li class="nav-item">
        <a class="nav-link {% if url_name == 'tasks' %} active {% endif %}" href="{% url
'classifier:tasks' %}">
          Tasks
        </a>
      </li>
      <li class="nav-item">
        <a class="nav-link {% if url_name == 'text' %} active {% endif %}" href="{% url
'classifier:text' %}">
          Classify Text
        </a>
      </li>
      <li class="nav-item">
        <a class="nav-link {% if url_name == 'document' %} active {% endif %}" href="{% url
'classifier:document' %}">
          Classify Document
        </a>
      </li>
    </ul>
    <ul class="navbar-nav navbar-right">
      <li class="nav-item dropdown">
        <a class="nav-link dropdown-toggle {% if url_name == 'profile' %} active {% endif %}"
           role="button" data-bs-toggle="dropdown" aria-expanded="false">
          Account
        </a>
        <ul class="dropdown-menu">
          {% if user.is_authenticated %}
          <li><a class="dropdown-item"> Logged in as: <br> {{ user }}</a></li>
          <li><hr class="dropdown-divider"></li>
```

```
          <li><a class="dropdown-item" href="{% url 'regenerate-token' %}">Regenerate
token</a></li>
          <li><a class="dropdown-item" href="{% url 'logout' %}">Logout</a></li>
          {% else %}
          <li><a class="dropdown-item" href="{% url 'login' %}">Login</a></li>
          <li><a class="dropdown-item" href="{% url 'django_registration_register'
%}">Register</a></li>
          {% endif %}
        </ul>
      </li>
    </ul>
</nav>
{% endwith %}

{% autoescape off %}{% bootstrap_messages %}{% endautoescape %}

<div class="container">
  {% block content %}{% endblock %}
</div>
{% endblock %}
```

Файл classifier/forms.py

```
from django import forms


class ClassifyTextForm(forms.Form):
    name = forms.CharField(required=False)
    text = forms.CharField(required=True, widget=forms.Textarea)


class ClassifyDocumentForm(forms.Form):
    name = forms.CharField(required=False)
    document = forms.FileField(required=True)
```

Файл classifier/urls.py

```
from django.urls import path

from . import views

app_name = "classifier"
urlpatterns = [
    path("", views.IndexView.as_view(), name="index"),
    path("text", views.ClassifyTextView.as_view(), name="text"),
    path("document", views.ClassifyDocumentView.as_view(), name="document"),
    path("tasks", views.TasksView.as_view(), name="tasks"),
]
```

Файл classifier/views.py

```
from django.shortcuts import render, redirect
from django.views import View, generic
from django.http import HttpRequest, HttpResponse

from django_celery_results.models import TaskResult
from django.contrib.auth.models import User
from django.contrib.auth.mixins import LoginRequiredMixin
from rest_framework.authtoken.models import Token

from common.tasks import classify_text, classify_document
from common.utilities import handle_document, handle_text, get_tasks
from .forms import ClassifyTextForm, ClassifyDocumentForm


class IndexView(generic.TemplateView):
    template_name = "classifier/index.html"


class ClassifyTextView(LoginRequiredMixin, View):
    template_name: str = "classifier/classify-text-page.html"

    def get(self, request: HttpRequest, format=None):
        form = ClassifyTextForm()
        context = {"form": form}
        return render(request, self.template_name, context=context)

    def post(self, request: HttpRequest, format=None):
        form = ClassifyTextForm(request.POST)
        if form.is_valid():
            name = form.cleaned_data["name"]
            text = form.cleaned_data["text"]

            user = request.user
```

```
            handle_text(name, text, user)

            return redirect("classifier:tasks")

        context = {"form": form}
        return render(request, self.template_name, context=context)


class ClassifyDocumentView(LoginRequiredMixin, View):
    template_name: str = "classifier/classify-document-page.html"

    def get(self, request: HttpRequest, format=None):
        form = ClassifyDocumentForm()
        context = {"form": form}
        return render(request, self.template_name, context=context)

    def post(self, request: HttpRequest, format=None):
        form = ClassifyDocumentForm(request.POST, request.FILES)
        if form.is_valid():
            name = form.cleaned_data["name"]
            document = request.FILES["document"]
            user = request.user

            handle_document(name, document, user)

            return redirect("classifier:tasks")

        context = {"form": form}
        return render(request, self.template_name, context=context)


class TasksView(LoginRequiredMixin, View):
    template_name: str = "classifier/tasks-page.html"

    def get(self, request: HttpRequest, format=None):
        user = request.user
        tasks = get_tasks(user)
        context = {"tasks": tasks}
        return render(request, self.template_name, context=context)


class RegenerateTokenView(LoginRequiredMixin, View):
    template_name: str = "classifier/regenerate-token-page.html"

    def get(self, request: HttpRequest, format=None):
        user = request.user
        try:
            Token.objects.get(user=user).delete()
        except:
            pass

        token = Token.objects.create(user=user)

        context = {"token": token}
        return render(request, self.template_name, context=context)
```

Файл classifier/templates/classify-document-page.html

```
{% extends "common/base-page.html" %}

{% load django_bootstrap5 %}

{% block title %} Classify Document {% endblock %}

{% block content %}
  <form method="post" enctype="multipart/form-data">
  {% csrf_token %}

  {% bootstrap_form form %}

  {% bootstrap_button button_type="submit" content="Submit" %}
  {% bootstrap_button button_type="reset" content="Reset" %}
  </form>
{% endblock %}
```

Файл classifier/templates/classify-text-page.html

```
{% extends "common/base-page.html" %}

{% load django_bootstrap5 %}

{% block title %} Classify Text {% endblock %}
```

```
{% block content %}
<form method="post" class="form">
  {% csrf_token %}

  {% bootstrap_form form %}

  {% bootstrap_button button_type="submit" content="Submit" %}
  {% bootstrap_button button_type="reset" content="Reset" %}
</form>
{% endblock %}
```

Файл classifier/templates/index.html

```
{% extends "common/base-page.html" %}

{% block title %}Index{% endblock %}

{% block content %}
<a href="{% url 'classifier:text' %}" class="btn btn-primary" role="button">Classify text</a>
<a href="{% url 'classifier:document' %}" class="btn btn-primary" role="button">Classify
document</a>
{% endblock %}
```

Файл classifier/templates/regenerate-token-page.html

```
{% extends "common/base-page.html" %}

{% block title %}Regenerate Token{% endblock %}

{% block content %}
<p>Your new token: {{token.key}}</p>
{% endblock %}
```

Файл classifier/templates/tasks-page.html

```
{% extends "common/base-page.html" %}

{% block title %} Tasks {% endblock %}

{% block content %}

{% if tasks %}
{% for task in tasks %}

<div class="card mb-3">
  <div class="card-header">
    {{ task.id  }}
  </div>
  <div class="card-body">
    <p> Name: {{ task.name }} </p>
    <p> Status: {{ task.status }}</p>
    {% if task.result != None %}
    <p>
    Result:
      {% for genre in task.result %}
      <span class="badge bg-primary">{{ genre }}</span>
      {% endfor %}
    </p>
    {% endif %}
  </div>
</div>
{% endfor %}

{% else %}
<p> You have no tasks yet </p>
{% endif %}


{% endblock %}
```

Файл classifier_api/urls.py

```
#!/usr/bin/env python3

from django.urls import path

from . import views

app_name = "classifier_api"
urlpatterns = [
    path("tasks", views.TasksAPIView.as_view(), name="tasks"),
    path("tasks/<str:task_id>", views.TasksAPIView.as_view(), name="tasks"),
    path("text", views.ClassifyTextAPIView.as_view(), name="text"),
    path("document", views.ClassifyDocumentAPIView.as_view(), name="document"),
```

```
]
```

Файл classifier_api/views.py

```
from celery.app import task
from rest_framework.views import APIView
from rest_framework.response import Response
from rest_framework.request import Request
from rest_framework import authentication, permissions, status
from django.contrib.auth.models import User

from common.utilities import handle_text, handle_document, get_tasks, get_task


class ClassifyTextAPIView(APIView):
    def post(self, request: Request, format=None):
        name = request.data["name"]
        text = request.data["text"]
        user = request.user
        task_id = handle_text(name, text, user)
        return Response({"task_id": task_id})


class ClassifyDocumentAPIView(APIView):
    def post(self, request: Request, format=None):
        name = request.data["name"]
        document = request.FILES["document"]
        user = request.user
        task_id = handle_document(name, document, user)
        return Response({"task_id": task_id})


class TasksAPIView(APIView):
    def get(self, request: Request, task_id=None, format=None):
        user = request.user

        try:
            if task_id:
                result = get_task(user, task_id)
            else:
                result = get_tasks(user)

            return Response(result)
        except:
            return Response(status=status.HTTP_404_NOT_FOUND)
```

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

ВЕБ-СЕРВІС ДЛЯ АВТОМАТИЧНОЇ КАТЕГОРИЗАЦІЇ ХУДОЖНІХ ТВОРІВ

Програма та методика тестування

КПІ.ІТ-9224.045440.04.51

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олесь КОВТУНЕЦЬ

Нормоконтроль:
_____ Ірина ВІТКОВСЬКА

Виконавець:
_____ Микола ТАТАРИН

Київ – 2023

ЗМІСТ

1 ОБ’ЄКТ ВИПРОБУВАНЬ

Об’єктом випробування є Веб-сервіс для автоматичної категоризації художніх творів. Тестування проводиться на ОС сімейства Linux.

2 МЕТА ТЕСТУВАННЯ

Метою тестування є наступне:

– перевірка правильності роботи програмного забезпечення відповідно до функціональних вимог;

– перевірка збереження даних;

– перевірка сумісності веб-інтерфейсу з останніми версіями сучасних браузерів (Chrome, Opera, Firefox, …);

– знаходження проблем, помилок і недоліків з метою їх усунення;

– перевірка зручності графічного інтерфейсу.

3 МЕТОДИ ТЕСТУВАННЯ

Для тестування програмного забезпечення використовуються такі методи:

– статичне тестування – перевіряється програма разом з усією документацією, яка аналізується на предмет дотримання стандартів програмування;

– мануальне тестування – тестування без використання автоматизації, тест-кейси пише особа, що тестує програмне забезпечення;

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування виконується мануально, з метою знаходження помилок та недоліків як у функціональній частині програмного забезпечення так і в зручності користування. Для того, щоб перевірити працездатність та відмовостійкість застосунку, необхідно провести наступні тестування:

- статичне тестування коду програмного забезпечення;
- тестування веб-застосунку на різних браузерах;
- тестування інтерфейсу користувача;
- тестування зручності використання.

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

ВЕБ-СЕРВІС ДЛЯ АВТОМАТИЧНОЇ КАТЕГОРИЗАЦІЇ ХУДОЖНІХ ТВОРІВ

Керівництво користувача

КПІ.ІТ-9224.045440.05.34

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олесь КОВТУНЕЦЬ

Нормоконтроль:
_____ Ірина ВІТКОВСЬКА

Виконавець:
_____ Микола ТАТАРИН

Київ – 2023

ЗМІСТ

1 ПРИЗНАЧЕННЯ ПРОГРАМИ

У склад програмного забезпечення для автоматизації категоризації художніх творів входить веб-сервіс.

Веб-сервіс надає користувачу веб-інтерфейс для категоризації та перегляду результатів. Окрім цього, сервіс також надає REST API для здійсненя створення задач класифікації та перегляду результатів.

2 ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ

2.1 Системні вимоги для коректної роботи

Для того, щоб скористуватись веб-сервісом необхідне виконання наступних вимог:

– наявність пристрою з остнньою версією браузеру Firefox або Chrome;

– наявність доступу до Інтернету;

2.2 Завантаження застосунку

Для того, щоб розгорнути сервіс, потрібно встановити docker і docker-compose.

Після цього потрібно зробити наступне

– Завантажити код з гітхабу проєкту на сторінці релізів.

– Зайти у директорію service у корені проєкту.

– Виконати команду docker compose build

– Виконати команду docker compose up -d

2.3 Перевірка коректної роботи

По завершенню виконання кроків у попередньому пункті потрібно перейти за адресою «http://localhost:8000/» (якщо ви ро. У випадку коректного розгортання сервісу, ви потрапите на головну сторінку веб-інтерфейсу.

3 ВИКОНАННЯ ПРОГРАМИ

При переході на головну сторінку користувач побачить форму авторизації (якщо він ще не авторизований), що містить два поля – поле для введення ім’я користувача і поле для паролю (рис. 3.1). Якщо вони будуть введені коректно, то після натиснення кнопки «Login» користувача буде автентифіковано та перенаправлено на сторінку перегляду результатів.

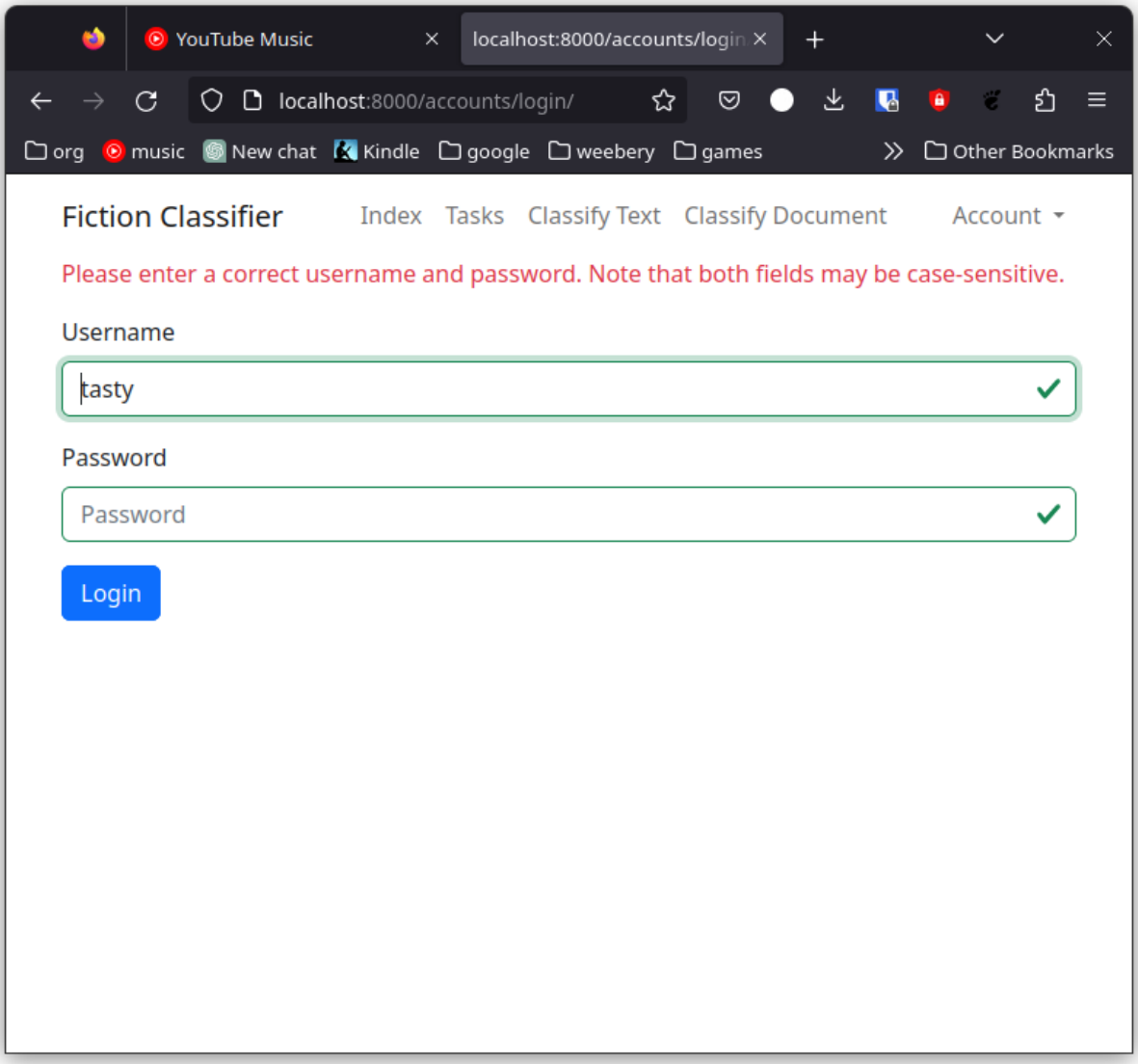


Рисунок 3.1 – Сторінка автентифікації

Якщо ж користувач не зареєстрований, то йому доведеться зареєструватися. Для цього потрібно обрати пункт «Register» у модальному вікні «Account». Після цього йому буде відображено форму реєстрації, де потрібно ввести незайняті адресу електронної пошти та ім’я користувача і пароль та підвердження паролю (рис. 3.2). Якщо все зробити правильно, то користувача буде зареєстровано і перенаправлено на сторінку перегляду результатів.

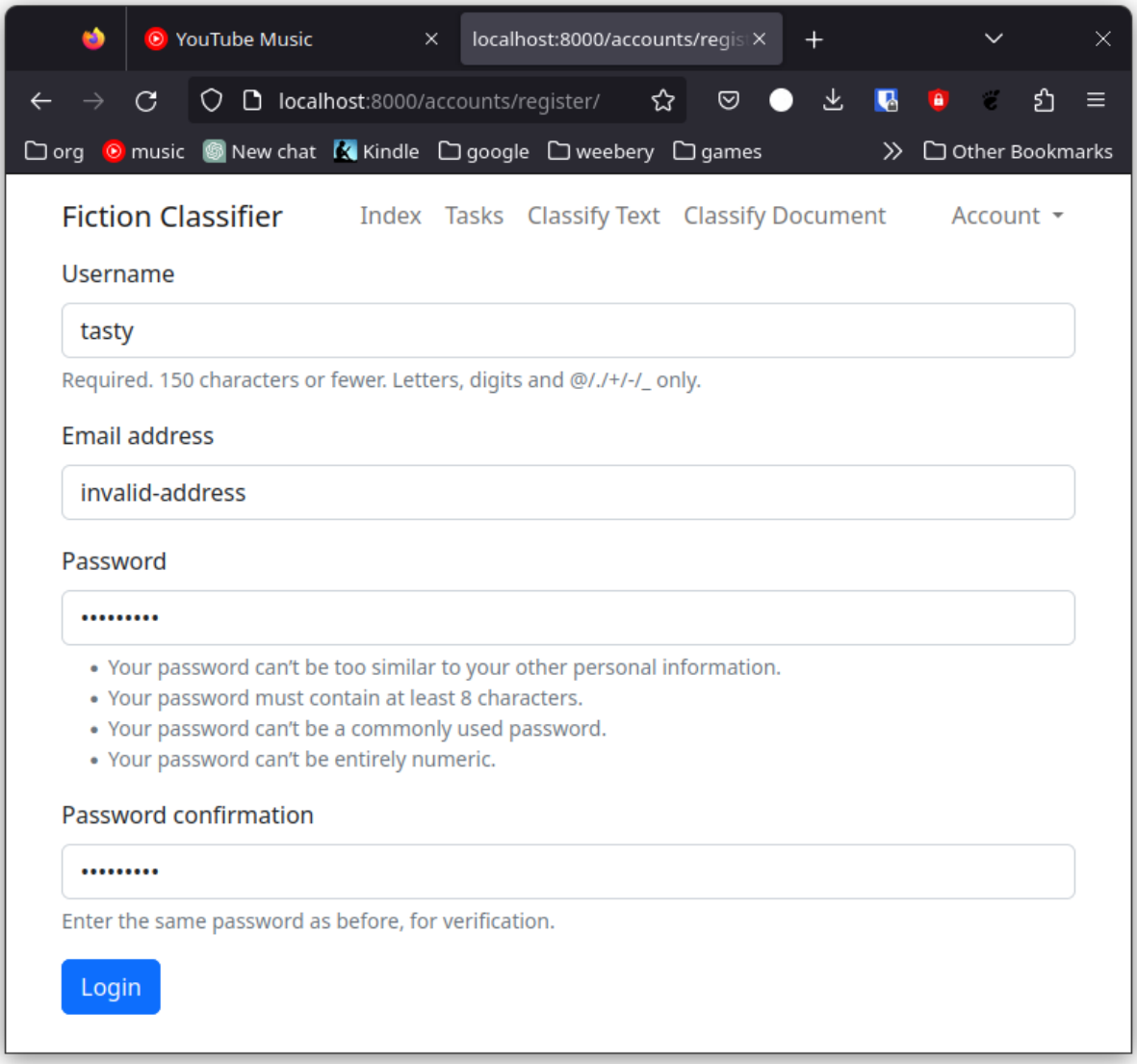


Рисунок 3.2 – Сторінка реєстрації

Після цього користувач зможе почати користуватись сервісом. Для того, щоб класифікувати текст, потрібно перейти на сторінку «Classify Text», ввести опціональну назву задачі і текст для класифікаці та натиснути кнопку «Submit» (рис. 3.3). Після цього користувача буде перенаправлено на сторінку перегляду результатів.

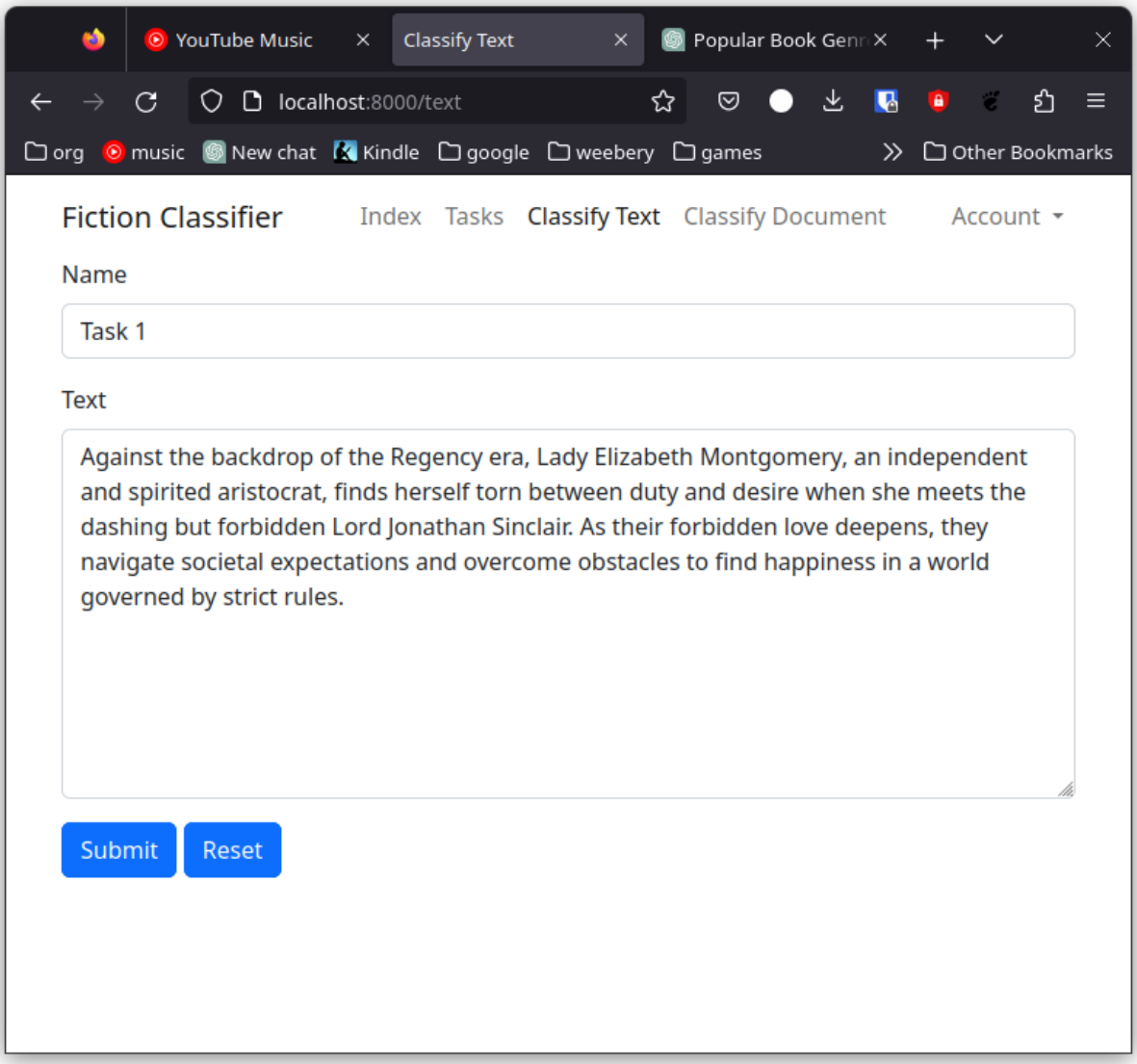


Рисунок 3.3 – Сторінка класифікації тексту

Для того, щоб класифікувати документ, потрібно перейти на сторінку «Classify Document», ввести опціональну назву задачі і завантажити файл для класифікаці та натиснути кнопку «Submit» (рис. 3.4). Після цього користувача буде перенаправлено на сторінку перегляду результатів.

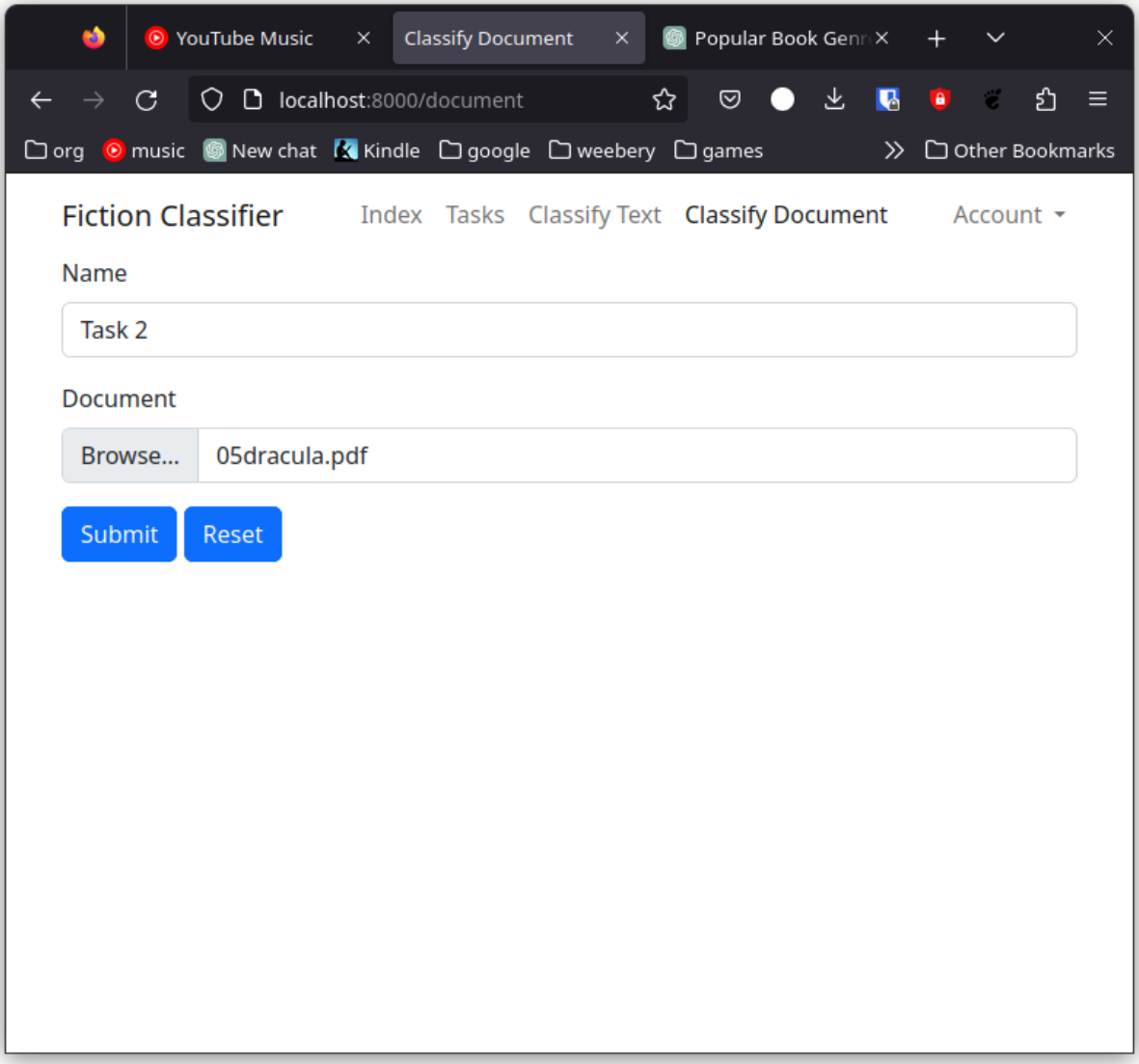


Рисунок 3.4 – Сторінка класифікації документу

Результати і статус виконання користувач зможе побачити на сторінці перегляду результатів (рис 3.5).

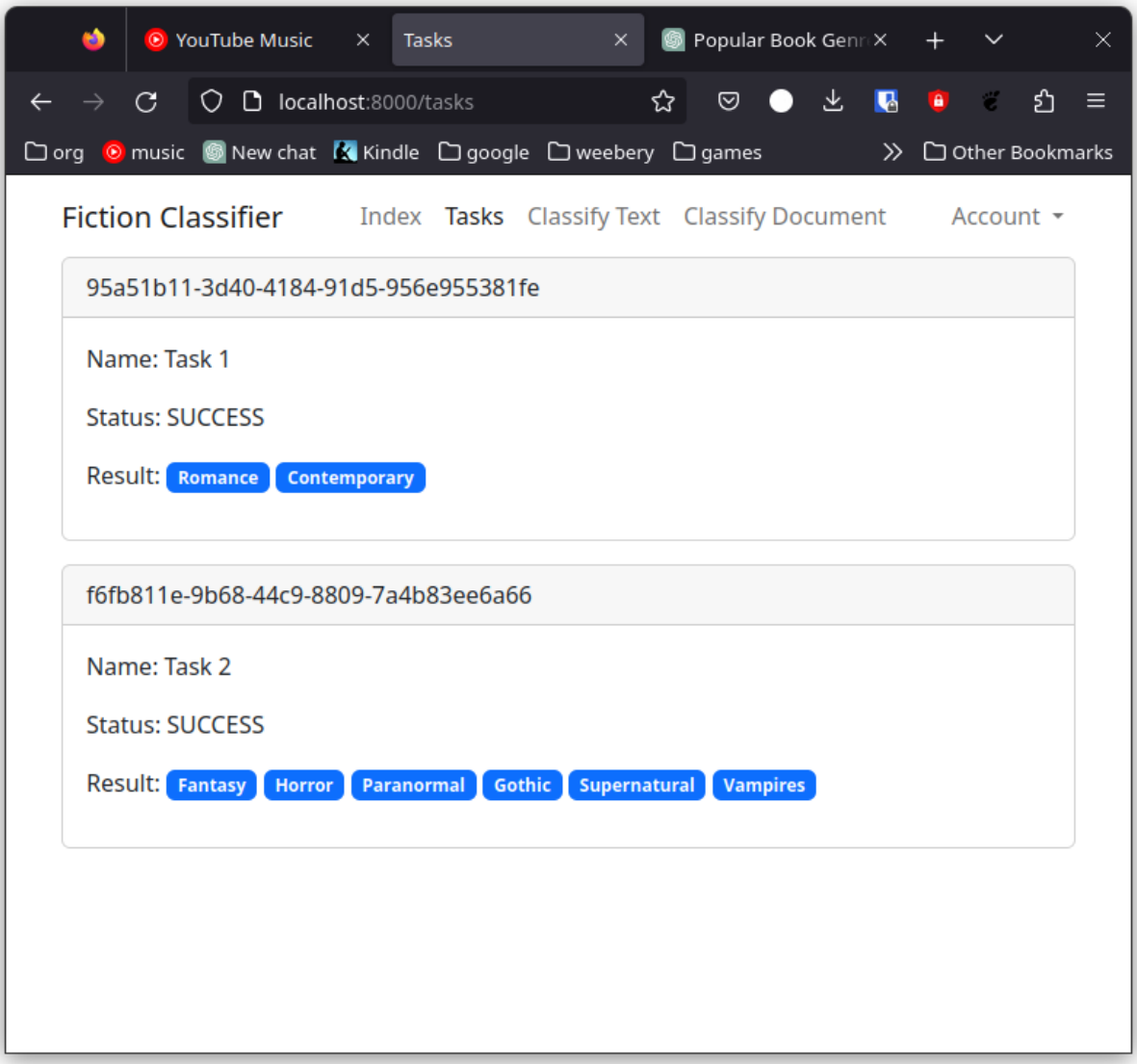


Рисунок 3.5 – Сторінка перегляду результатів

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

ВЕБ-СЕРВІС ДЛЯ АВТОМАТИЧНОЇ КАТЕГОРИЗАЦІЇ ХУДОЖНІХ ТВОРІВ

Графічний матеріал

КПІ.ІТ-9224.045440.05.99

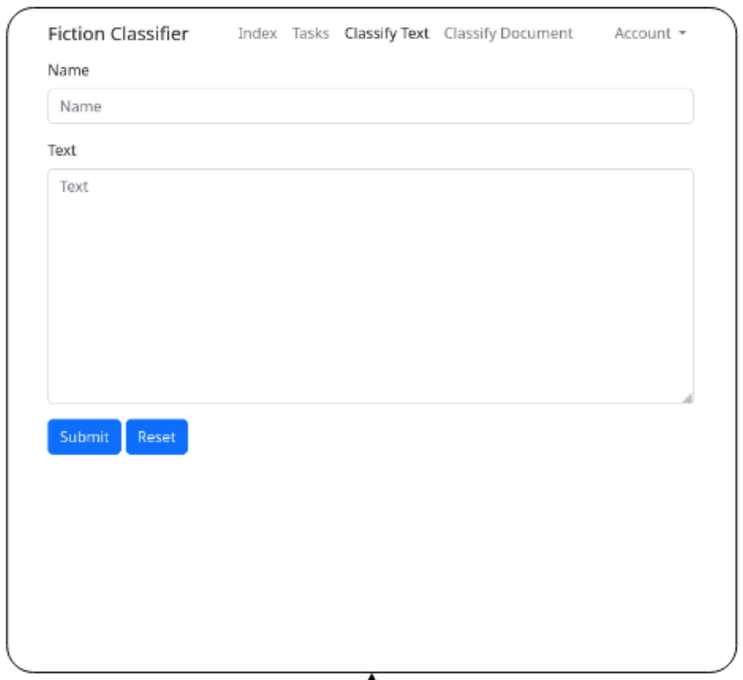
“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олесь КОВТУНЕЦЬ

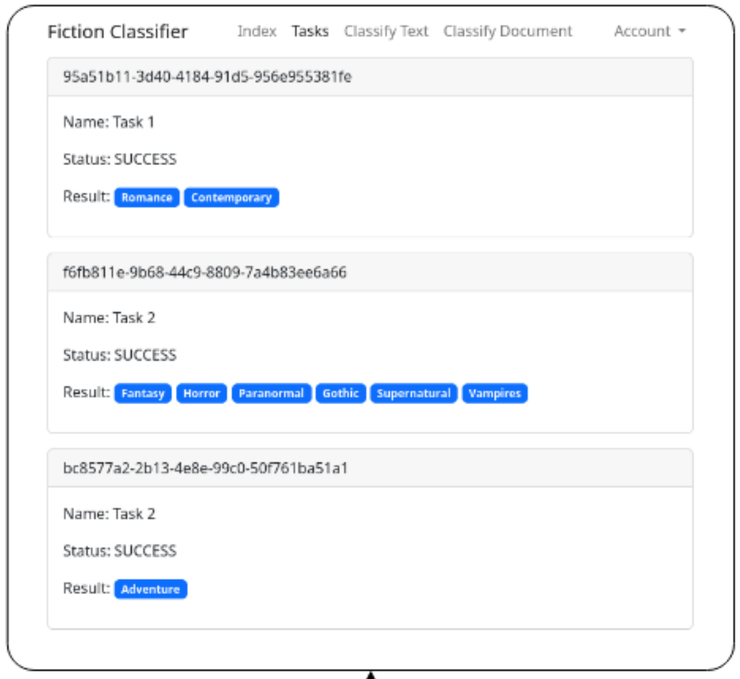
Нормоконтроль:
_____ Ірина ВІТКОВСЬКА

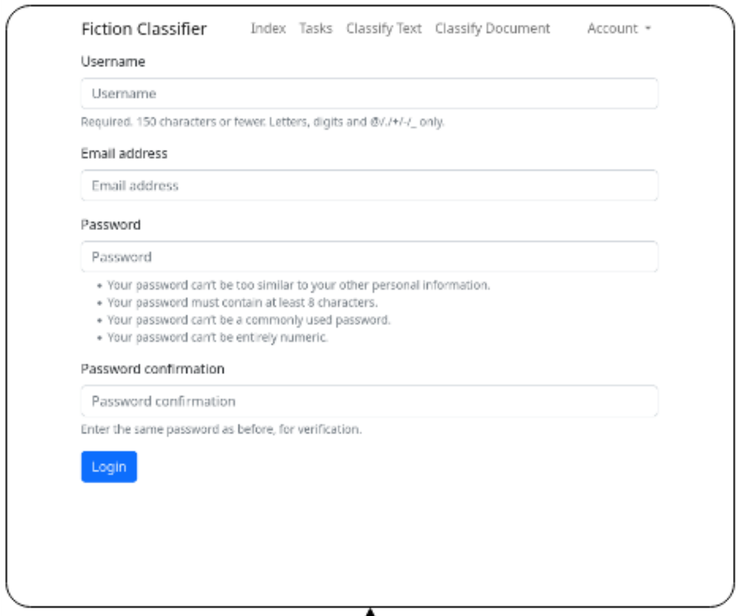
Виконавець:
_____ Микола ТАТАРИН

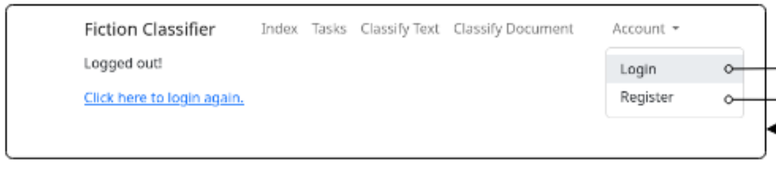
Київ – 2023

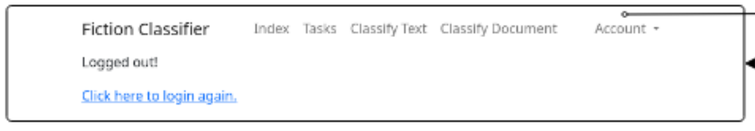
КПІ.ІТ-9224.045440.6.99.КЕ











					КПІ.ІТ-9224.045440.6.99.КЕ	
Зм.	Арк.	№ документа	Підпис	Дата		
Розробив		Татарин М.С.			Креслення екранних форм	Літера: — / Маса: — / Масштаб: —
Перевірив		Ковтунець О.В.				Аркуш: — / Аркушів: —
Т. кон.					Веб-застосунок для автоматичної категоризації художніх творів	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІПТ-92
Н. кон.		Вітковська І.І.				
Затвердив		Жаріков Е.В.				

КПІ.ІТ-9224.045440.6.99.СБД

f

					КПІ.ІТ-9224.045440.6.99.СБД
Зм.	Арк.	№ документа	Підпис	Дата	Схема бази даних
Розробив		Татарин М.С.			Літера: — Маса: — Масштаб: —
Перевірив		Ковтунець О.В.			Аркуш: — Аркушів: —
Т. кон.					Веб-застосунок для автоматичної категоризації художніх творів
Н. кон.		Вітковська І.І.			КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІПТ-92
Затвердив		Жаріков Е.В.			

					КПІ.ІТ-9224.045440.6.99.СС	
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна компонентів програмного забезпечення	Літера / Маса / Масштаб
Розробив		Татарин М.С.				
Перевірив		Ковтунець О.В.				Аркуш / Аркушів
Т. кон.						
Н. кон.		Вітковська І.І.			Веб-застосунок для автоматичної категоризації художніх творів	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІПТ-92
Затвердив		Жаріков Е.В.				

КПІ.ІТ-9224.045440.6.99.ССБ

					КПІ.ІТ-9224.045440.6.99.ССБ	
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна бізнес-процесів	Літера / Маса / Масштаб
Розробив		Татарин М.С.				
Перевірив		Ковтунець О.В.				Аркуш / Аркушів
Т. кон.						
Н. кон.		Вітковська І.І.			Веб-застосунок для автоматичної категоризації художніх творів	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІПТ-92
Затвердив		Жаріков Е.В.				