Міністерство освіти і науки України

Криворізький національний університет

Кафедра моделювання та програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти магістра

зі спеціальності 121 – Інженерія програмного забезпечення

На тему: Розробка програмного забезпечення рекомендаційної системи для інформаційного сайту

Засвідчую, що в цій кваліфікаційній роботі немає запозичень із праць інших авторів без відповідних посилань.
Студент гр. ІПЗ-24м

_____/Г. О. Лушов /

Керівник кваліфікаційної роботи _____ / А. М. Стрюк /

Завідувач кафедри _____ / А. М. Стрюк /

Кривий Ріг 2025

Криворізький національний університет

Факультет: Інформаційних технологій

Кафедра: Моделювання та програмного забезпечення

Ступінь вищої освіти: магістр

Спеціальність: 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:

Зав. кафедри

_____ А.М.Стрюк

«__» _____ 20__р.

ЗАВДАННЯ

на кваліфікаційну роботу

студенту групи ІПЗ-24м Лушов Гліб Олександрович

1. Тема: Розробка програмного забезпечення рекомендаційної системи для інформаційного сайту. Затверджено наказом по КНУ №__ від «_» _____ 2025р.
2. Термін подання студентом закінченої роботи : «_» _____ 2025р.
3. Вихідні дані по роботі: вихідні данні повинні виводити рекомендації по схожим для користувачів.
4. Зміст пояснювальної записки (перелік питань, що їх треба розробити): проаналізувати існуючі рекомендаційні системи що використовуються для сайтів, розглянути технології що можна використати для розробки, розробити програмне забезпечення для реалізації системи.
5. Перелік ілюстративного матеріалу: блок-схеми алгоритмів, знімки екрану з відображенням інтерфейсу, графи нейронних мереж, схеми взаємодії програмних модулів.

Календарний план:

№	Найменування етапів кваліфікаційної роботи	Термін виконання етапів роботи
1	Огляд літератури за тематикою роботи та пошук необхідної інформації та даних для її виконання	25.12.2024 – 15.03.2025
2	Проведення аналізу існуючих теоретичних методів дослідження і вирішення проблеми	16.03.2025 – 25.03.2025
3	Проведення критичного порівняльного аналізу існуючих аналогів програмних систем	25.03.2025– 31.03.2025
4	Формулювання актуальності і завдань роботи	01.04.2025– 15.04.2025
5	Оформлення матеріалів першого розділу роботи	16.04.2025 – 31.04.2025
6	Розробка математичних, структурних і функціональних моделей	01.05.2025– 15.05.2025
7	Розробка структур даних, основних класів та алгоритмів програмного комплексу	16.05.2025– 30.05.2025
8	Оформлення матеріалів другого розділу роботи	31.06.2025 – 10.06.2025
9	Розробка баз даних і знань, програмних модулів, бібліотек та інтерфейсу програмного комплексу	11.06.2025– 15.07.2025
10	Оформлення матеріалів третього розділу роботи	16.07.2025– 30.07.2025
11	Дослідження та узагальнення результатів роботи з точки зору наукової та технічної цінності	01.08.2025 – 31.08.2025
15	Остаточне оформлення пояснювальної записки	11.11.2025 – 20.11.2025

Дата видачі завдання: «____» _____20___р.
Студент _____/ Г. О. Лушов /
Керівник роботи _____/ А.М.Стрюк /

РЕФЕРАТ

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ ДЛЯ ІНФОРМАЦІЙНОГО САЙТУ.

Пояснювальна записка: 42 с., 4 табл., 27 рис., 2 дод., 5 джерел.

За останні роки спостерігається стрімке зростання кількості інформаційних сайтів, які надають користувачам відомості про події, свята та фестивалі. Однак, через велику кількість даних виникає проблема ефективного пошуку та персоналізованого відбору подій, що відповідають інтересам користувачів. У зв'язку з цим актуальним є розроблення рекомендаційної системи, яка дозволить автоматизовано підбирати події відповідно до вподобань відвідувачів сайту.

Метою кваліфікаційної роботи є розробка програмного забезпечення рекомендаційної системи для інформаційного сайту, що містить календар різноманітних заходів. Ця система повинна аналізувати інтереси користувачів та надавати їм персоналізовані рекомендації щодо майбутніх подій, використовуючи сучасні алгоритми машинного навчання та обробки даних.

Практичне значення роботи полягає в покращенні користувацького досвіду завдяки автоматизованому підбору подій, що може сприяти підвищенню залученості відвідувачів сайту. Розроблена система допоможе зменшити час на пошук релевантних заходів та забезпечить ефективне використання інформації про події. Це також може бути корисним для організаторів заходів, які отримають цільову аудиторію для своїх подій.

Дослідження включає аналіз існуючих підходів до розробки рекомендаційних систем, вибір оптимальних алгоритмів для побудови рекомендацій, розробку програмного забезпечення та його тестування. Також буде проведена оцінка ефективності запропонованого рішення та порівняння його з іншими аналогічними системами.

ABSTRACT

DEVELOPMENT OF A SOFTWARE RECOMMENDATION SYSTEM FOR AN INFORMATION SITE.

Explanatory note: 42 p., 4 tables, 27 images, 2 appendix, 5 sources.

In recent years, there has been a rapid increase in the number of informational websites that provide users with details about events, holidays, and festivals. However, due to the vast amount of data, there is a challenge in efficiently searching for and selecting personalized events that match users’ interests. Therefore, the development of a recommendation system that can automatically select events according to website visitors' preferences is highly relevant.

The goal of this qualification project is to develop recommendation system software for an informational website that features a calendar of various events. This system should analyze users' interests and provide them with personalized recommendations for upcoming events using modern machine learning and data processing algorithms.

The practical value of this work lies in improving the user experience through the automated selection of events, which can contribute to increased user engagement with the website. The developed system will help reduce the time spent searching for relevant events and ensure the effective use of event-related information. It can also be beneficial for event organizers by helping them reach a targeted audience for their events.

The research includes an analysis of existing approaches to recommendation system development, the selection of optimal algorithms for generating recommendations, software development, and testing. Additionally, the effectiveness of the proposed solution will be evaluated and compared to other similar systems.

ЗМІСТ

Умовні позначення, скорочення

МН – Машинне навчання

НМ – Нейромережа

IDE - Інтегроване середовище розробки

ВСТУП

За останні роки нейронні мережі стали одним із ключових інструментів у багатьох галузях, включаючи медицину, транспорт, бізнес і розваги. Їхнє впровадження дозволило значно вдосконалити автоматизацію процесів, аналіз великих обсягів даних та розв'язання складних задач, таких як розпізнавання зображень, обробка природної мови та прогнозування.

Завдяки швидкому розвитку обчислювальних потужностей і великим наборам даних, моделі нейронних мереж, як-от глибоке навчання, забезпечують все більш точні результати, що змінює спосіб функціонування багатьох сучасних технологій і відкриває нові перспективи для наукових відкриттів та інновацій.

Окрім цього, нейронні мережі дозволяють створювати нові системи, зокрема рекомендаційні системи для інформаційних сайтів. Такі системи аналізують поведінку користувачів, їхні вподобання та інтереси, щоб запропонувати персоналізований контент.

Алгоритми глибокого навчання можуть обробляти історію переглядів, взаємодії з матеріалами та навіть семантичний аналіз текстів, щоб формувати релевантні рекомендації. Що не лише покращує досвід користувача, але й сприяє збільшенню часу перебування на сайті та підвищенню лояльності аудиторії.

Важливо також відзначити роль рекомендаційних систем для інформаційного сайту, адже вони стають ключовим інструментом у боротьбі за увагу користувачів. Завдяки таким системам можна ефективно структурувати великий обсяг контенту та забезпечити його подання відповідно до інтересів і потреб кожного відвідувача. Це не лише полегшує навігацію для користувачів, але й підвищує ймовірність повторних відвідувань сайту, збільшує трафік і сприяє монетизації через рекламу або підписки.

1 ДОСЛІДЖЕННЯ ДОСВІДУ ПРОЄКТУВАННЯ ТА РОЗРОБКИ РЕКОМЕНДАЦІЙНИХ СИСТЕМ.

1.1 Роль і задачі рекомендаційних систем в сучасному інформаційному просторі

Рекомендаційні системи відіграють важливу роль у сучасному інформаційному просторі, оскільки вони допомагають користувачам знаходити релевантну інформацію серед величезного обсягу даних. Їх задачі можуть бути дуже різними. Наприклад це може бути:

- *Аналіз поведінки користувачів:* Основною задачею є збирання і аналіз даних про дії користувачів: їхні покупки, перегляди, лайки, коментарі тощо. Це дозволяє створювати персоналізовані рекомендації;
- *Побудова моделі рекомендацій*: Включає використання різноманітних методів (колаборативна фільтрація, контентна фільтрація, гібридні підходи) для прогнозування інтересів користувачів;
- *Оптимізація рекомендацій:* Постійна адаптація та поліпшення моделей рекомендацій на основі зворотного зв'язку від користувачів і змін у їхній поведінці або зовнішніх обставинах;
- *Персоналізація інформації:* рекомендаційні системи дозволяють адаптувати великий обсяг даних до конкретних інтересів кожного користувача, забезпечуючи індивідуальний підхід.
- *Збільшення залученості користувачів:* завдяки точним рекомендаціям, користувачі частіше взаємодіють з платформою, довше залишаються на сайті та повертаються повторно.
- *Підтримка бізнес-рішень:* допомагають компаніям краще розуміти поведінку користувачів, формувати маркетингові стратегії, оптимізувати пропозиції товарів

1.2 Методи машинного навчання та штучного інтелекту в сучасних рекомендаційних системах

Алгоритми машинного навчання являють собою сукупність обчислювальних методів, що забезпечують здатність інформаційних систем до самонавчання на основі аналізу даних, без необхідності жорсткого програмування кожної окремої операції. Їх застосування дає змогу формувати математичні моделі, здатні виявляти приховані закономірності в інформаційних масивах, здійснювати прогнозування майбутніх подій та приймати обґрунтовані рішення, базуючись на попередньому досвіді.

Штучні нейронні мережі — це вид обчислювальних структур, які імітують принципи функціонування біологічного мозку. Вони складаються з великої кількості взаємопов'язаних обчислювальних елементів — штучних нейронів, що здійснюють обробку вхідної інформації, передають її між шарами мережі та генерують вихідний сигнал. Завдяки здатності моделювати складні нелінійні залежності, нейромережеві моделі широко використовуються у задачах прогнозування, класифікації даних, розпізнавання образів і в інших сферах інтелектуального аналізу даних.

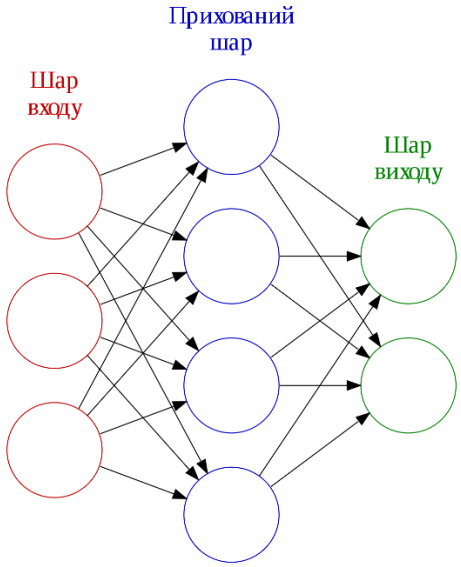


Рисунок 1.1 – Приклад одношарового персептрону нейромережі

Лінійна регресія є одним із базових методів прогнозування, який дозволяє

встановити залежність між незалежними змінними та цільовим показником. Цей підхід застосовується для передбачення значень певної величини на основі наявних параметрів. Метод демонструє високу ефективність у випадках, коли між змінними спостерігається приблизно лінійний взаємозв'язок.

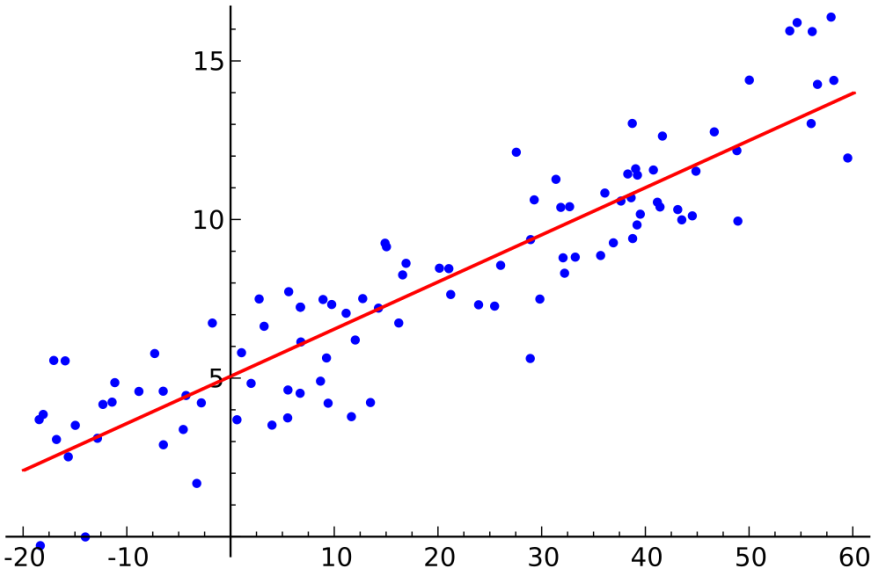


Рисунок 1.2 – Приклад лінійної регреції

Логістична регресія — це статистичний підхід, що використовується для оцінки ймовірності настання певної події за заданими вхідними даними. У контексті функціонування інформаційного ресурсу з календарем подій, логістична регресія може бути використана для моделювання ймовірності зацікавленості користувача в окремій події або здійснення переходу до її детального опису.

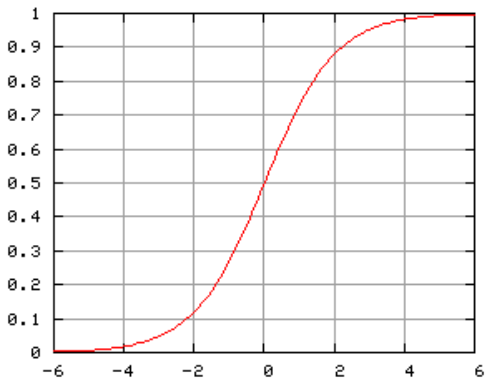


Рисунок 1.3 – Приклад логістичної регреції

Метод k найближчих сусідів (K-Nearest Neighbors, KNN) є одним із найпростіших алгоритмів класифікації та прогнозування, який ґрунтується на принципі схожості. Він дозволяє передбачити інтерес користувача до певної події шляхом аналізу подібності з іншими подіями, які вже були переглянуті або відвідані. Модель визначає "найближчих сусідів" — тобто події зі схожими характеристиками (такими як тематика, локація чи час проведення), або дії користувачів зі схожими вподобаннями, — і на основі цієї інформації формує рекомендації для конкретного користувача.

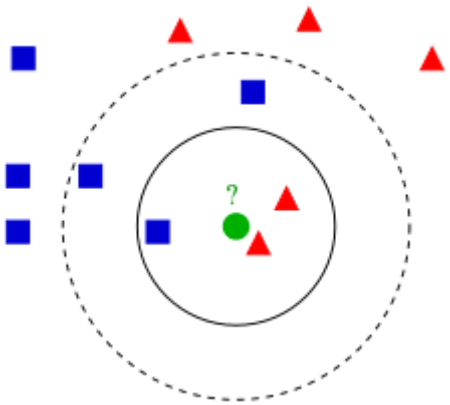


Рисунок 1.4 – K-найближчі сусіди (KNN)

У класичному прикладі роботи алгоритму, зразок, розміщений у межах певної області (наприклад, зеленого кола), потребує класифікації до однієї з можливих категорій — наприклад, «синій квадрат» або «червоний трикутник». Якщо вибрати k = 3, і серед найближчих сусідів більшість становлять червоні трикутники (2 проти 1), то зразок буде віднесено до цієї категорії. Зі збільшенням параметра до k = 5, ситуація може змінитися: наприклад, якщо переважатимуть сині квадрати (3 проти 2), класифікація буде змінена відповідно. Це демонструє, наскільки важливим є правильний вибір значення k.

Згорткові нейронні мережі (Convolutional Neural Networks, CNN) представляють собою спеціалізований тип штучних нейронних мереж, ефективний для обробки даних із просторовою структурою, зокрема

зображень. У контексті інформаційного порталу з календарем подій CNN можуть застосовуватися для аналізу візуального контенту (афіш, банерів, фотографій), автоматичної класифікації подій за стилістичними або тематичними ознаками, а також оцінки популярності заходів на основі візуальних атрибутів.

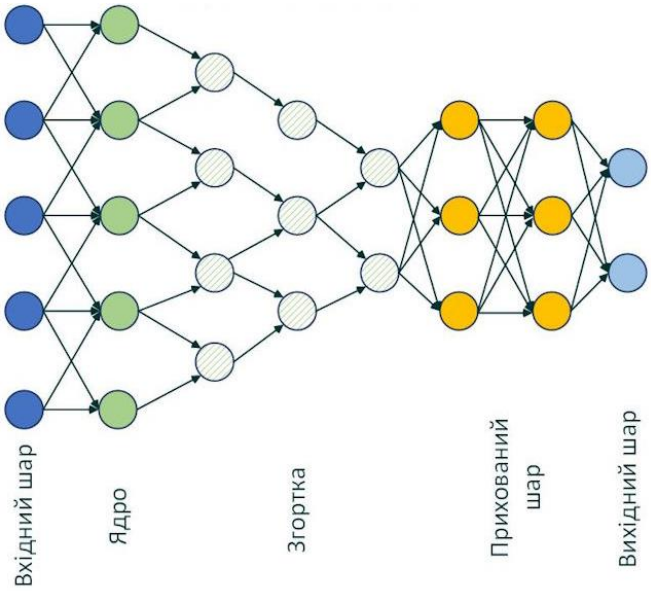


Рисунок 1.5 – Згорткова нейронна мережа (CNN)

Згортка є базовою операцією, що використовується в згорткових нейронних мережах (Convolutional Neural Networks, CNN) для обробки вхідних зображень. Вона передбачає застосування однакових наборів фільтрів (вагових коефіцієнтів) до різних областей вхідного простору. Завдяки спільному використанню параметрів по всій площі зображення, кількість тренованих параметрів значно зменшується, що сприяє спрощенню моделі, підвищує її обчислювальну ефективність і зменшує ризик перенавчання. Така властивість робить CNN особливо придатними для вирішення задач комп’ютерного зору, таких як класифікація зображень, виявлення об'єктів та аналіз візуального контенту.

Рекурентні нейронні мережі (Recurrent Neural Networks, RNN) є класом нейронних архітектур, орієнтованих на обробку послідовних або часових

даних. У контексті інформаційного ресурсу з календарем подій, RNN можуть бути використані для моделювання динаміки змін у вподобаннях користувача з урахуванням історії його взаємодії з контентом. Наприклад, такі моделі здатні аналізувати, які заходи користувач переглядав у певні сезони, дні тижня або на основі тематики подій, тим самим враховуючи тимчасові закономірності в його інтересах.

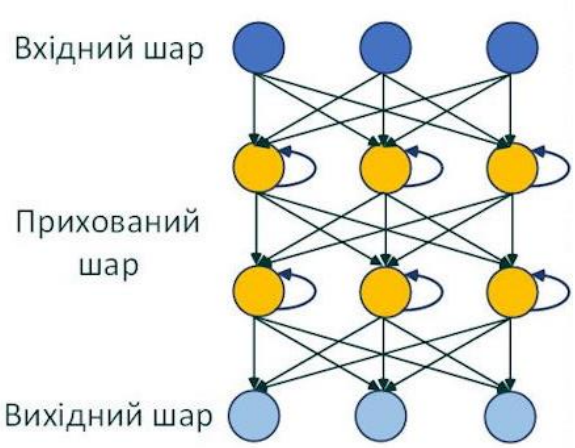


Рисунок 1.6 – Рекурентні нейронні мережі (RNN)

Фундаментальною особливістю RNN є наявність внутрішньої пам’яті, яка дозволяє зберігати інформацію про попередні стани мережі та використовувати її при обробці наступних елементів послідовності. Це забезпечує врахування короткострокових і середньострокових залежностей у даних, що є критично важливим у задачах, пов’язаних з аналізом часових рядів. RNN широко застосовуються в системах прогнозування, автоматичного розпізнавання мовлення, обробки природної мови та виявлення закономірностей у даних із часовою структурою.

1.3 Економічна цінність від даної системи

Впровадження рекомендованої системи на інформаційному сайті з календарем подій має кілька важливих економічних переваг, які можуть значно збільшити дохід від платформи.

Монетизація через рекламу:

Завдяки точним рекомендаціям система може підвищити ефективність рекламних кампаній. Реклама буде націлена на користувачів з певними інтересами, що дозволить рекламодавцям досягати кращих результатів з меншими витратами. Рекламні пропозиції можуть бути інтегровані в календар подій, наприклад, показуючи рекламу на події схожої тематики або що відбудуться в схожі часові проміжки.

Партнерські програми та афілійовані посилання:

Інформаційний сайт може співпрацювати з організаторами подій, театрами, концертними майданчиками або іншими бізнесами. Рекомендаційна система буде пропонувати користувачам події за допомогою афілійованих посилань, що дозволить отримувати комісії з продажу квитків або участі в заходах.

Аналіз поведінки користувачів для створення нових продуктів:

Зібрані дані про вподобання користувачів допомагають прогнозувати майбутні тренди та попит. Це дозволяє розробляти нові функції або контент, який буде найбільш популярним, і відповідно, знову ж таки, генерувати додаткові доходи через нові платні пропозиції чи продукти.

Покращення лояльності користувачів:

Персоналізовані рекомендації підвищують задоволення користувачів, що може призвести до збільшення їхньої лояльності до платформи. Чим більше користувачі знаходять для себе корисних подій через систему, тим більше вони користуються сайтом, а отже, зростає й потенціал для монетизації через повторні підписки, покупки квитків або взаємодію з рекламою.

2 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ ДЛЯ ІНФОРМАЦІЙНОГО САЙТУ

2.1 Розробка та опис схем програмного забезпечення рекомендаційної системи для інформаційного сайту

Структурна діаграма є графічним представленням архітектури системи або програмного продукту, що відображає основні компоненти та взаємозв'язки між ними. Вона слугує інструментом для візуалізації внутрішньої організації програмного забезпечення, дозволяючи наочно показати модулі, класи, їхні властивості та способи взаємодії в межах системи.

Наступна діаграма допомагає зрозуміти структуру програми, визначити порядок виконання операцій та усвідомити взаємодію її складових.

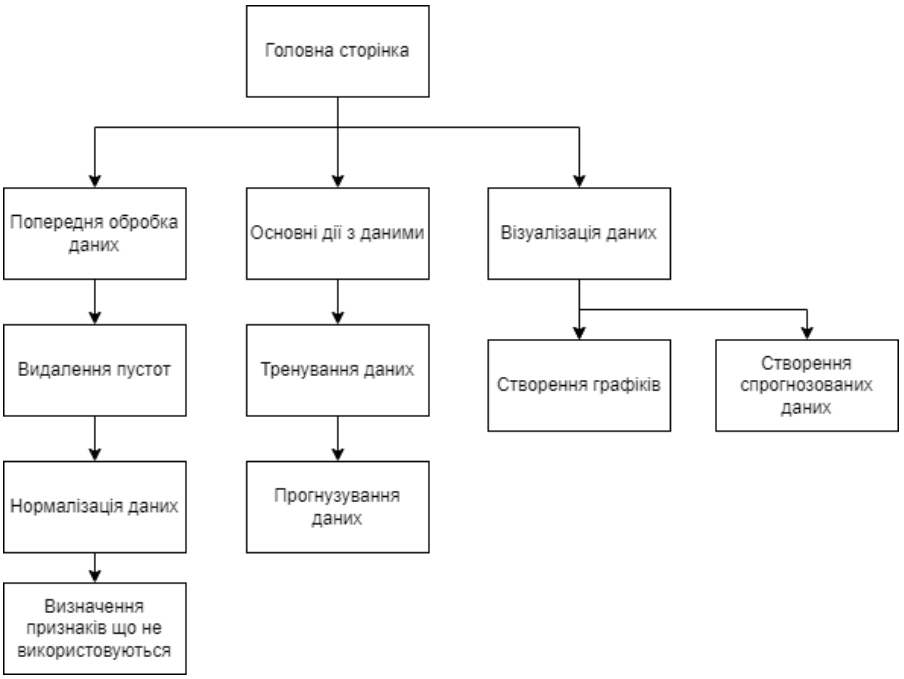


Рисунок 2.1 – Структурна діаграма

Тому на Рисунку 2.1 представлена структурна діаграма, що описує функціональні компоненти програми:

1. Попередня обробка даних: Включає очищення даних, нормалізацію,

видалення порожніх значень та визначення ознак, що не використовуються.

2. Основні дані: Здійснюється аналіз даних для виявлення закономірностей і тенденцій, що охоплює:
    - Тренування моделі: навчання моделі машинного навчання на доступних даних.
    - Прогнозування даних: використання навченої моделі для передбачення нових значень.
3. Візуалізація даних.
    - Створення графіків: відображення закономірностей і тенденцій у даних.
    - Створення таблиць: представлення даних у табличному форматі.

Діаграма потоків даних (DFD) є графічним засобом, який демонструє рух даних у системі та взаємозв’язки між її складовими. Вона дозволяє наочно відобразити, як дані проходять через різні частини системи і взаємодіють між собою.

На Рисунку 2.2 представлена діаграма потоків даних, яка ілюструє взаємодію системи з користувачем та сервером. Користувач передає дані у систему, яка їх обробляє, формує прогноз і відображає результати у вигляді візуалізацій.

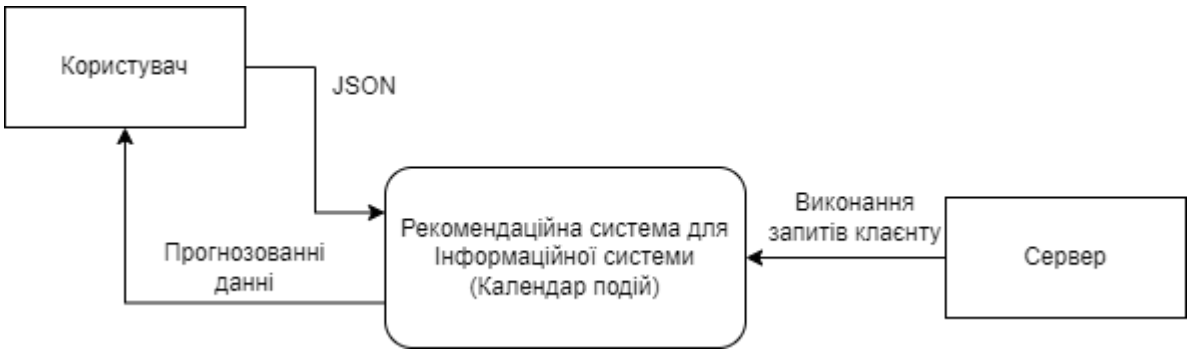


Рисунок 2.2 – Контекстна діаграма потоків даних

Загальна діаграма на Рисунку 2.2 повністю відображає процес передачі даних у системі. Вона демонструє всі етапи взаємодії між користувачем та системою, включно зі збором і введенням даних, їх обробкою та аналізом, а

також поданням результатів користувачу.

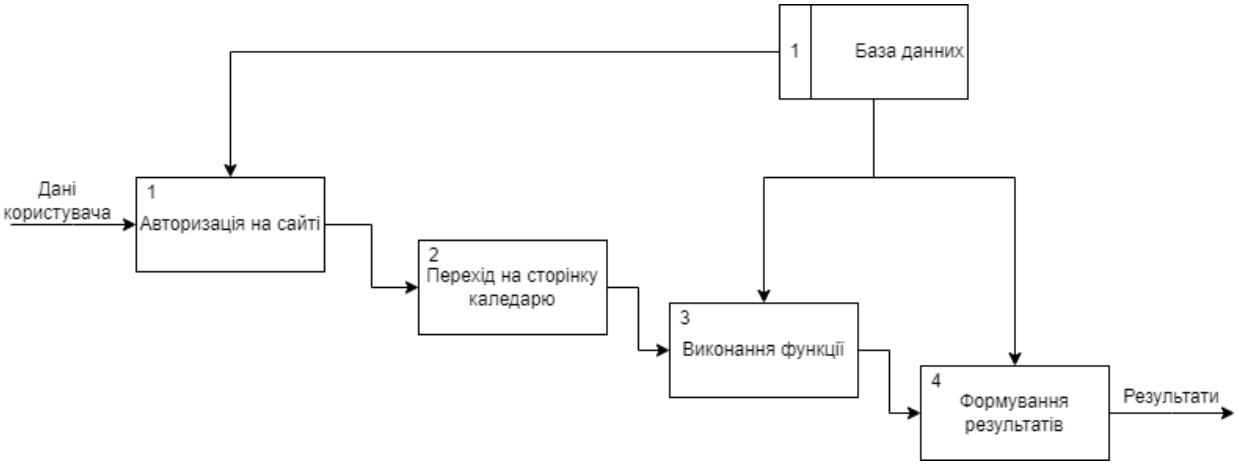


Рисунок 2.3 – Загальна діаграма потоків даних

Рисунок 2.3 ілюструє діаграму потоків даних, де показано:

1. База даних: система використовує базу даних, для зберігання інформації про події та користувачів.
2. Авторизація на сайті: користувач авторизується на сайті за допомогою логіну та паролю.
3. Перехід на сторінку календарю: перед заходом на сторінку зі сторону серверу загружаються дані про події користувача та його рекомендації.
4. Виконання функції: функція обробляє дані користувача на сервері.
5. Формування результатів: система генерує результати та надсилає їх клієнту.
6. Результати: дані відображаються на клієнтському пристрої і демонструються користувачу.

Рисунок 2.4 – Детальна діаграма потоків даних

Детальніше пункт 3 попередньої діаграми розкрито на Рисунку 2.4. На цьому етапі здійснюється безпосередня обробка даних, що може включати перевірку даних, прогнозування, навчання моделі та інші дії залежно від функціональних потреб системи. Діаграма обробки даних складається з таких кроків:

- Перевірка даних: оцінка їх відповідності заданій структурі.
- Обробка даних: виконання операцій над даними, таких як прогнозування або тренування моделі.
- Повернення помилки: формування повідомлення про помилку та відображення його на стороні клієнта.

Функціональна діаграма є засобом візуалізації функціональності системи або програмного продукту, а також встановлення взаємозв'язків між окремими функціями. Вона дозволяє краще зрозуміти принцип роботи системи та спектр виконуваних нею завдань.

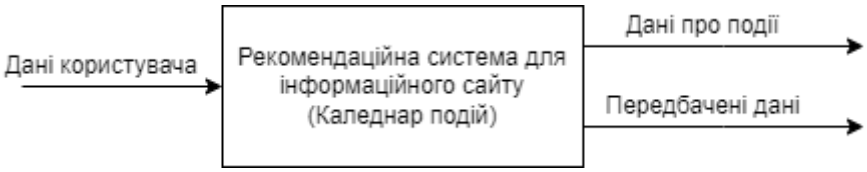


Рисунок 2.5 – Загальна функціональна діаграма

На Рисунку 2.5 показано загальну функціональну модель обробки та аналізу даних у системі. Вхідні дані можуть надходити у форматі JSON або CSV. Система обробляє ці дані та надає користувачу різні результати, такі як розклад автобусів, прогнозні дані щодо розкладу та графіки.

Такий підхід дозволяє перетворювати вхідну інформацію на корисні результати, забезпечуючи зручність і ефективність для користувачів.

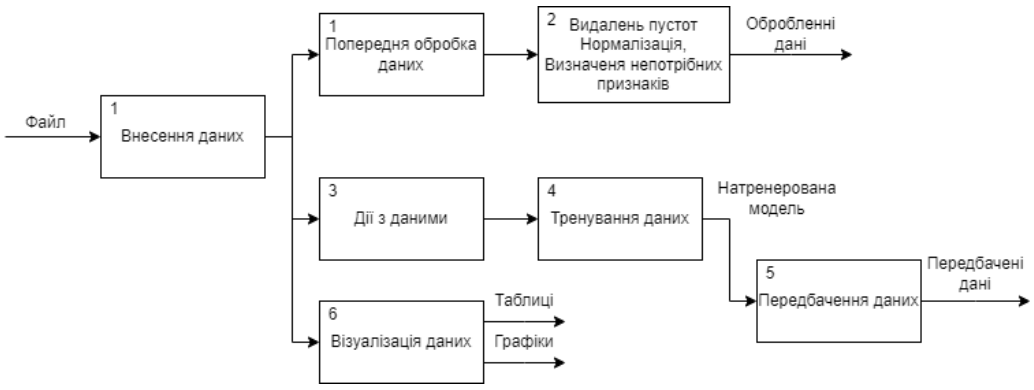


Рисунок 2.6 – Функціональна діаграма

Рисунок 2.6 демонструє детальнішу функціональну модель тренувальної програми. Діаграма відображає ключові етапи процесу навчання, починаючи з підготовки даних, їх перевірки та обробки, включаючи навчання моделі, прогнозування, і завершується візуалізацією результатів для користувача.

На рисунку 2.6 зображено

- Попередня обробка даних: підготовка даних до аналізу, включаючи видалення пропусків, нормалізацію та виключення непотрібних ознак.
- Внесення даних: завантаження підготовлених даних у систему.
- Дії з даними: трансформація або додаткове очищення даних перед тренуванням моделі.
- Тренування даних: навчання моделі машинного навчання на підготовлених даних для виявлення закономірностей та прогнозування або прийняття рішень.
- Прогнозування даних: використання натренованої моделі для передбачення нових значень.
- Візуалізація результатів: представлення результатів роботи моделі у вигляді таблиць або графіків для наочного сприйняття.

2.2 Проектування потоків даних рекомендаційної системи для інформаційного сайту

Функціональні вимоги визначають конкретні завдання та операції, які система повинна забезпечувати, а також встановлюють механізми її взаємодії з користувачами та іншими інформаційними системами. Вони формують основу для моделювання логіки роботи програмного забезпечення та визначають його функціональні межі.

До таких вимог належать обробка та аналіз даних, взаємодія з базами даних, формування звітної документації, а також організація комунікації користувача з системою через інтерфейс або інтеграцію з зовнішніми сервісами.

Таблиця 2.1 – Функціональні вимоги до системи.

№	Вимога
1	Користувачі повинні мати можливість створити обліковий запис, увійти в систему та вийти з неї.
2	Користувачі можуть переглядати події, представлені у вигляді інтерактивного календаря. Інтерфейс повинен підтримувати перегляд за днями, тижнями або місяцями.
3	Користувачам повинні автоматично пропонуватись події на основі їхніх інтересів, переглядів, уподобань або минулої активності. Рекомендації можуть бути реалізовані за допомогою моделей машинного навчання, таких як KNN, логістична регресія або нейромережі.
4	Інтерфейс має бути інтуїтивно зрозумілим, адаптивним до мобільних пристроїв та забезпечувати легкий доступ до всіх основних функцій: перегляду календаря, фільтрації подій, перегляду деталей тощо.

В той же час нефункціональні вимоги описують не конкретні функції системи, а її загальні характеристики та якості, які вона повинна мати, щоб бути ефективною, надійною та придатною для користувачів. Ці вимоги фокусуються на тому, як система повинна виконувати свої функції, а не що вона повинна робити.

Ці вимоги охоплюють такі показники, як час відновлення системи після збоїв, швидкість реагування на дії користувача, рівень захисту даних від несанкціонованого доступу, а також інші характеристики, що визначають загальну продуктивність та надійність роботи системи.

Таблиця 2.2 – Нефункціональні вимоги до системи.

№	Вимога
1	Інтерфейс має бути інтуїтивно зрозумілим, підтримувати адаптивний дизайн для різних пристроїв (ПК, планшети, смартфони) та відповідати принципам UX/UI.
2	Робота системи має бути стабільною та надійною, з акцентом на мінімізацію будь-яких можливих збоїв чи помилок.
3	Архітектура системи має бути побудована таким чином, щоб у разі збільшення кількості користувачів або подій, вона могла масштабуватись без суттєвого зниження продуктивності.
4	Система повинна забезпечувати захист даних користувачів, включно з автентифікацією, шифруванням паролів. Доступ до панелі адміністратора має бути обмеженим.
5	У разі збою система повинна мати механізми для збереження даних та автоматичного відновлення. Регулярне резервне копіювання даних має бути обов’язковим.
6	Кодова база та структура системи повинні бути організовані таким чином, щоб їх можна було легко підтримувати, оновлювати або розширювати новими функціями.

2.3 Технології та інструменти для реалізації

Під час розробки системи з алгоритмом адаптивного формування тарифів для пасажирських перевезень передбачено використання мови JavaScript, клієнт-серверної архітектури, а також відповідних фреймворків і бібліотек.

Таблиця 2.3 – Використані при розробці технології

Технологія/Інструмент	**Опис**
Клієнт Бібліотеки	React – бібліотека JavaScript для розробки динамічних інтерфейсів користувача, яка дозволяє створювати інтерактивні та швидко оновлювані веб-сторінки.
	Redux – бібліотека для управління станом інтерфейсу користувача. Вона використовується для централізованого зберігання даних та синхронного оновлення інформації, що відображається у компонентах інтерфейсу.
	Axios – це JavaScript бібліотека для виконання HTTP-запитів з браузера. Вона працює на основі обіцянок (Promises) і дозволяє легко надсилати GET, POST, PUT, DELETE та інші типи запитів до серверу.

Продовження Таблиці 2.3

Технологія/Інструмент	**Опис**
Сервер Фреймворк	Flask - це мікрофреймворк для веб-розробки на Python, який дозволяє швидко створювати веб-додатки. Flask підтримує розширення, такі як робота з базами даних, авторизація, шаблони, та інші необхідні функції для створення веб-сервісу.
Сервер Бібліотеки	TensorFlow – бібліотека з відкритим кодом для машинного навчання, що дозволяє створювати, тренувати та застосовувати моделі машинного навчання для різних задач.
Система контролю версій	Git — інструмент версійного контролю, що дозволяє відслідковувати історію змін у вихідному коді, керувати версіями проекту й організовувати командну розробку.
Інтегроване середовище	Visual Studio Code – середовище розробки з відкритим кодом, яке підтримує процес програмування та пропонує інструменти для зручного редагування, налагодження та контролю версій коду.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕСПЕЧЕННЯ

3.1 Вибір інструментів розробника

Під час вибору технологій для розробки програмного забезпечення було враховано низку факторів, що підтверджують доцільність використання мови програмування TypeScript та середовища розробки Visual Studio Code. TypeScript, будучи надбудовою над JavaScript, забезпечує додаткову безпеку та зручність у розробці, зберігаючи всі переваги екосистеми JavaScript. Завдяки строгій типізації та покращеній підтримці інтегрованих середовищ розробки, TypeScript дозволяє створювати масштабовані та надійні веб-додатки.

Серед плюсів та мінусів якого можна виділити:

TypeScript додає до JavaScript можливість статичної типізації, що дозволяє визначати типи змінних, функцій, об'єктів та інших структур ще до запуску програми. Це дає змогу виявляти логічні та синтаксичні помилки вже на етапі компіляції, значно знижуючи ймовірність появи несподіваних помилок під час виконання. Завдяки строгій типізації розробники отримують надійніший і передбачуваніший код, що особливо важливо у великих командах і проєктах.

Завдяки поєднанню статичної типізації, чітких правил структуризації коду та інтеграції з редакторами коду (наприклад, VS Code), TypeScript забезпечує високий рівень підтримки для середніх і великих проєктів. Це дозволяє багатьом розробникам працювати над одним кодовим базисом без втрати контролю над якістю або складністю коду. TypeScript допомагає в навігації по коду, автодоповненні, рефакторингу та тестуванні, що загалом покращує продуктивність команди та надійність програмного продукту.

Однією з основних переваг TypeScript є його повна сумісність із JavaScript. Це забезпечує можливість використовувати будь-який існуючий код JavaScript без необхідності внесення змін для роботи в середовищі TypeScript. Крім того, компілятор TypeScript перетворює TypeScript-код у чистий JavaScript, який можна запускати в будь-якому середовищі, де підтримується JavaScript —

браузерах, Node.js тощо. Це робить перехід на TypeScript плавним і безболісним, особливо для проєктів, які вже частково написані на JavaScript.

Одним із недоліків TypeScript є необхідність попередньої компіляції коду перед його виконанням. Оскільки браузери та середовища на зразок Node.js не розуміють TypeScript безпосередньо, код потрібно компілювати у JavaScript за допомогою спеціального інструменту (наприклад, tsc).

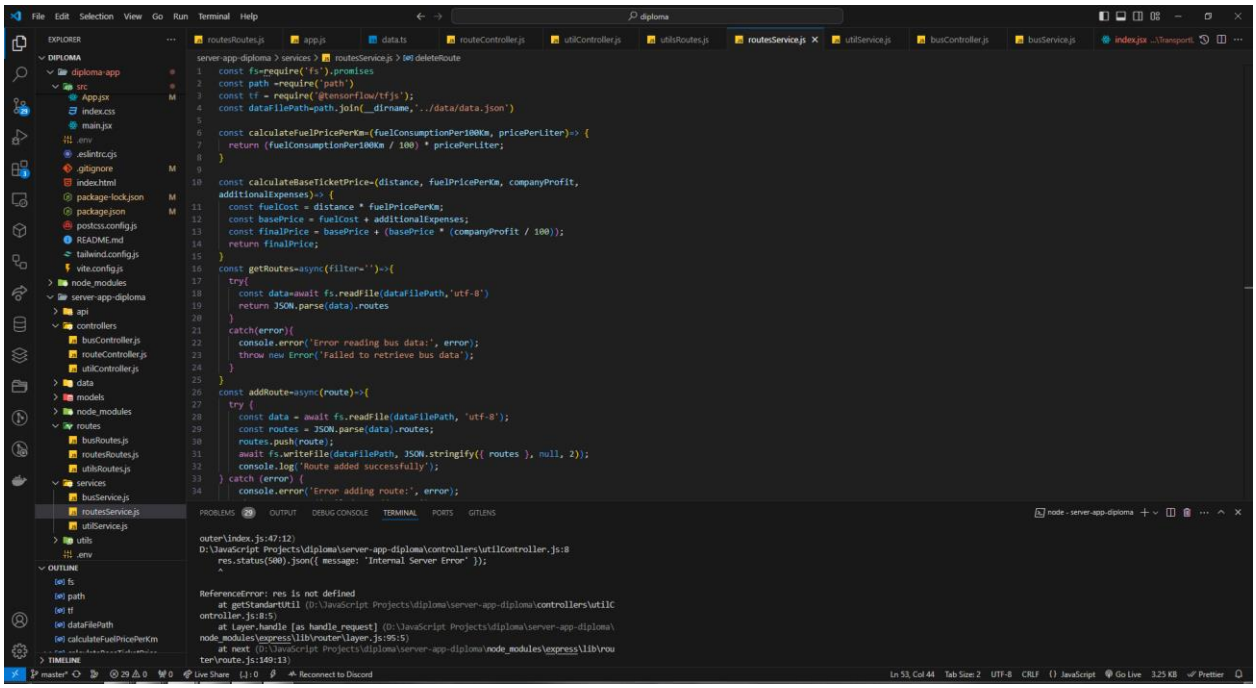


Рисунок 3.1 – Visual Studio Code (IDE)

Visual Studio Code є інтегрованим середовищем розробки (IDE), створеним компанією Microsoft. Воно слугує потужним інструментом для розробки різноманітних програмних продуктів, включаючи веб- та мобільні додатки, десктопні програми, ігри, сервіси та інші рішення.

IDE надає широкий спектр функцій, що підтримують розробників на всіх етапах створення програмного забезпечення — від написання коду до його тестування та налагодження. Основна частина середовища відображає редактор коду (наприклад, на JavaScript) із можливістю імпорту модулів, визначення функцій та конфігурації викликів API.

Зліва розташована бічна панель з вкладками, такими як Explorer, Search,

Source Control та Extensions. Вкладка Explorer дозволяє переглядати структуру проекту, включаючи папки на кшталт node_modules, public та файли, такі як package.json.

У нижньому лівому куті IDE знаходяться піктограми для управління обліковим записом та налаштуваннями середовища, а зверху розташоване меню з опціями File, Edit та View для керування проектом і робочим простором.

3.2 Створення сховища даних

Під час створення цього сховища даних використовувалася база данних SQLite.

Дані про події представлені у вигляді таблиці, де кожен таблиці представляє окрему подію, а стовпці містять ключові характеристики кожної з них, такі як назва, опис, дата проведення, тип, популярність, локація та тривалість.

	title	description	date	type	popularity	popularit...	location	duration
1	Event 1	Description for Event 1	2025-03-01 00:00:00.000000	Festival	Big	1001	Park	8
2	Event 2	Description for Event 2	2025-03-03 00:00:00.000000	Weekend	Medium	500	Beach	48
3	Event 3	Description for Event 3	2025-03-06 00:00:00.000000	Festival	Big	1501	City Center	12
4	Event 4	Description for Event 4	2025-03-06 00:00:00.000000	Weekend	Small	200	Mountain	48
5	Event 5	Description for Event 5	2025-03-18 00:00:00.000000	Festival	Medium	800	Stadium	10
6	Event 6	Description for Event 6	2025-03-12 00:00:00.000000	Weekend	Small	300	Lake	48
7	Event 7	Description for Event 7	2025-03-12 00:00:00.000000	Festival	Big	2000	Downtown	15
8								

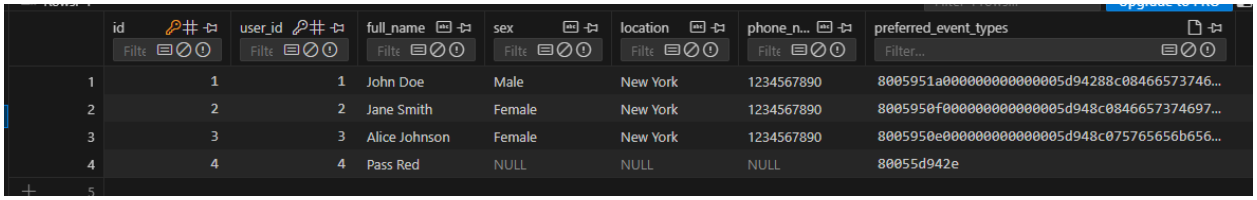
Рисунок 3.2 – Приклад подій

Таблиця 3.1 – Таблиця подій

Назва строки	**Опис**
id	Ідентифікатор події
title	Назва події
description	Описання події
date	Дата та час події
type	Тип події
popularity	Популярність події

popularity_count	Кількість відвідувачів події
location	Місце події
duration	Тривалість події (в минутах)

Дані про деталі користувачів представлені у вигляді таблиці, де кожен рядок відповідає окремій особі, а стовпці містять інформацію про ID, повне ім'я, стать, місцезнаходження, номер телефону та вподобання щодо типів подій.



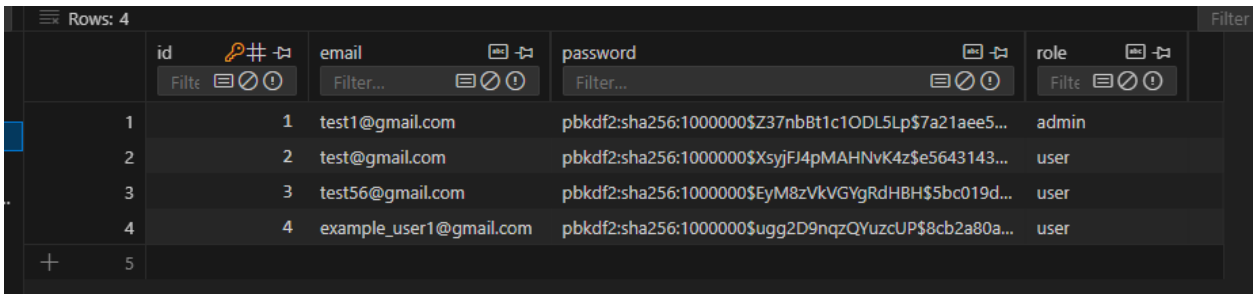
	id	user_id	full_name	sex	location	phone_n...	preferred_event_types
1	1	1	John Doe	Male	New York	1234567890	8005951a000000000000005d94288c08466573746...
2	2	2	Jane Smith	Female	New York	1234567890	8005950f000000000000005d948c0846657374697...
3	3	3	Alice Johnson	Female	New York	1234567890	8005950e000000000000005d948c075765656b656...
4	4	4	Pass Red	NULL	NULL	NULL	80055d942e

Рисунок 3.3 – Приклад деталей про користувачів

Таблиця 3.2 – Таблиця деталей

Технологія/Інструмент	**Опис**
id	Ідентифікатор деталі користувача
User_id	Ідентифікатор користувача
Full_name	Прізвище та ім’я користувача
Sex	Стать користувача
Location	Локація користувача
Phone_number	Номер телефону користувача
Preferred_event_types	Бажані типи подій користувача

Дані про деталі користувачів представлені у вигляді таблиці, де кожен рядок представляє окремого користувача з унікальним ідентифікатором, адресою електронної пошти, захешованим паролем (алгоритм pbkdf2:sha256) та роллю ("admin" або "user").



Rows: 4

	id	email	password	role
1	1	test1@gmail.com	pbkdf2:sha256:1000000$Z37nbBt1c1ODL5Lp$7a21aee5...	admin
2	2	test@gmail.com	pbkdf2:sha256:1000000$XsyjFJ4pMAHNvK4z$e5643143...	user
3	3	test56@gmail.com	pbkdf2:sha256:1000000$EyM8zVkVGYgRdHBH$5bc019d...	user
4	4	example_user1@gmail.com	pbkdf2:sha256:1000000$ugg2D9nqzQYuzcUP$8cb2a80a...	user
5				

Рисунок 3.4 – Приклад користувачів

Таблиця 3.3 – Таблиця користувачів

Технологія/Інструмент	**Опис**
id	Ідентифікатор користувача
Email	Електрона пошта користувача
Password	Пароль користувача
role	Роль користувача

3.3 Програмна реалізація рекомендаційної системи

Програмна архітектура дослідження організована як клієнт-серверний веб-сайт, де частина відповідальна за передбачення зосереджена на серверному боці. Ця частина коду розділена на три основні функції:

```
def prepare_data(self):
    """
    Prepare training data from UserDetails and Events.
    """
    users = UserDetails.query.all()
    events = Event.query.all()

    user_ids = [user.id for user in users]
    event_ids = [event.id for event in events]

    user_event_matrix = np.zeros((len(user_ids), len(event_ids)))

    for user in users:
        for event in user.events:
            user_event_matrix[user_ids.index(user.id)][event_ids.index
            (event.id)] = 1

    return user_ids, event_ids, user_event_matrix
```

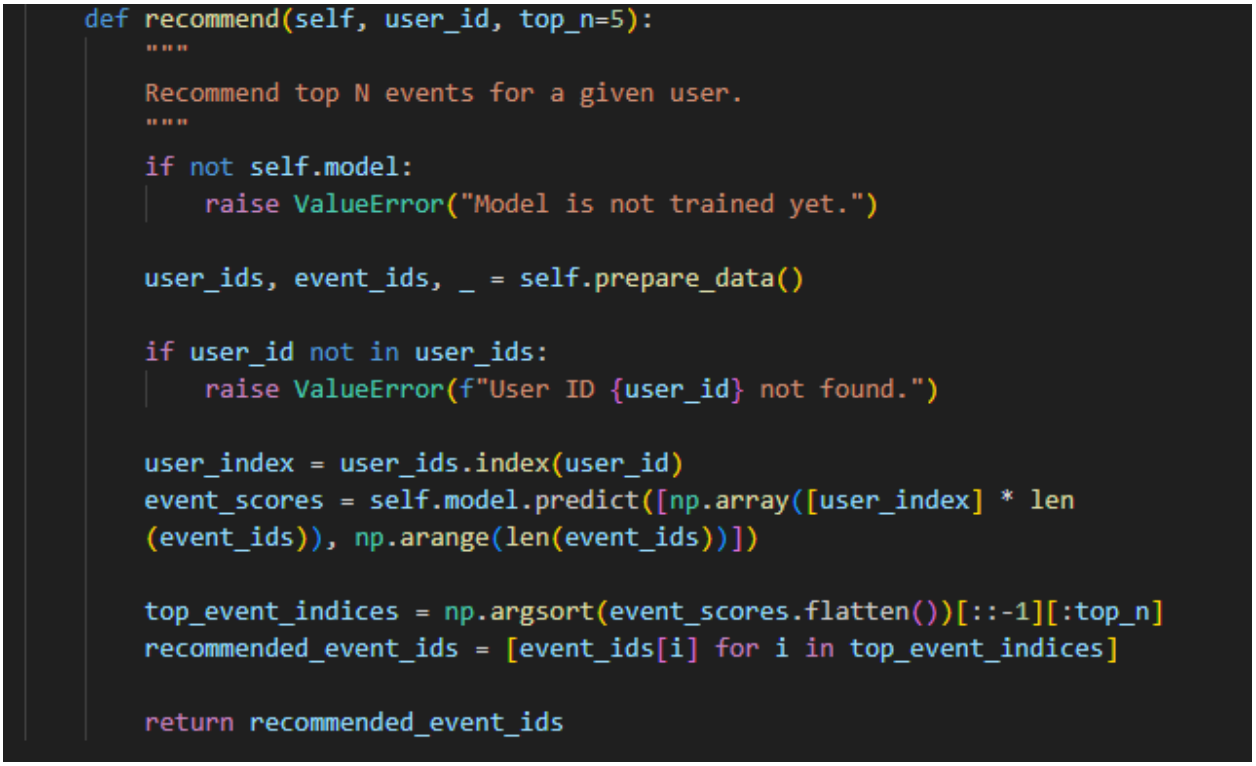
Рисунок 3.5 – Код prepare_data

preprocess_data: відповідає за підготовку навчальних даних для моделі рекомендацій. Він отримує всі записи з таблиць *UserDetails* та *Event*, витягує унікальні ідентифікатори користувачів і подій, а потім створює матрицю взаємодій розміром кількість користувачів * кількість подій. Кожен елемент цієї матриці встановлюється в 1, якщо користувач був пов'язаний з певною подією (тобто переглядав або відвідував її), і в 0 в іншому випадку. Ця матриця зберігає інформацію про зв'язки між користувачами та подіями, яка буде використана для навчання моделі.

```
def train(self):
    """
    Train the recommendation model.
    """
    user_ids, event_ids, user_event_matrix = self.prepare_data()

    num_users = len(user_ids)
    num_events = len(event_ids)

    user_input = tf.keras.layers.Input(shape=(1,))
    event_input = tf.keras.layers.Input(shape=(1,))

    user_embedding = tf.keras.layers.Embedding(num_users, 50)(user_input)
    event_embedding = tf.keras.layers.Embedding(num_events, 50)(event_input)

    dot_product = tf.keras.layers.Dot(axes=2)([user_embedding,
    event_embedding])
    output = tf.keras.layers.Flatten()(dot_product)

    self.model = tf.keras.models.Model(inputs=[user_input, event_input],
    outputs=output)
    self.model.compile(optimizer='adam', loss='binary_crossentropy',
    metrics=['accuracy'])

    user_indices, event_indices = np.where(user_event_matrix > 0)
    labels = user_event_matrix[user_indices, event_indices]

    self.model.fit(
        [user_indices, event_indices],
        labels,
        epochs=10,
        batch_size=32
    )
```

Рисунок 3.6 – Код train

train: навчає модель рекомендацій на основі даних, підготовлених методом *prepare_data*. Спочатку визначається кількість користувачів і подій, після чого створюються вхідні тензори для користувачів і подій. Для кожного з них використовується вбудований (embedding) шар, який перетворює індекси у вектори з 50 вимірами. Потім обчислюється скалярний добуток (dot product) між векторами користувачів і подій, результат якого розглядається як ймовірність інтересу. Модель компілюється з оптимізатором - *adam* і функцією втрат - *binary_crossentropy*, що підходить для бінарних завдань класифікації. Для тренування використовуються індекси пар користувач–подія, де була зафіксована взаємодія (тобто значення у матриці дорівнює 1).



```
def recommend(self, user_id, top_n=5):
    """
    Recommend top N events for a given user.
    """
    if not self.model:
        raise ValueError("Model is not trained yet.")

    user_ids, event_ids, _ = self.prepare_data()

    if user_id not in user_ids:
        raise ValueError(f"User ID {user_id} not found.")

    user_index = user_ids.index(user_id)
    event_scores = self.model.predict([np.array([user_index] * len
    (event_ids)), np.arange(len(event_ids))])

    top_event_indices = np.argsort(event_scores.flatten())[::-1][:top_n]
    recommended_event_ids = [event_ids[i] for i in top_event_indices]

    return recommended_event_ids
```

Рисунок 3.7 – Код predicts

recommend: реалізує механізм генерації персоналізованих рекомендацій для конкретного користувача. Спочатку перевіряється, чи модель була навчена. Далі знову викликається метод *prepare_data* для отримання ідентифікаторів користувачів та подій. Якщо заданий *user_id* не знайдено, викликається помилка. Інакше модель прогнозує "оцінки" (відповідні

ймовірності зацікавленості) для всіх подій для вказаного користувача. Події сортуються за спаданням оцінки, і повертається список *top_n* подій з найвищими прогнозними значеннями. Таким чином, користувач отримує рекомендації на основі передбаченої релевантності подій.

3.4 Зовнішній вигляд застосунку.

Фронт-енд сторона даного веб-сайту була розроблена щоб продемонструвати роботу рекомендаційної системи на живому прикладі. Тобто за допомогою даного веб-сайту можно побачити, як використання рекомендаційної системи на бек-енд частині інформаційного веб-сайту буде виглядати користувача даної системи.

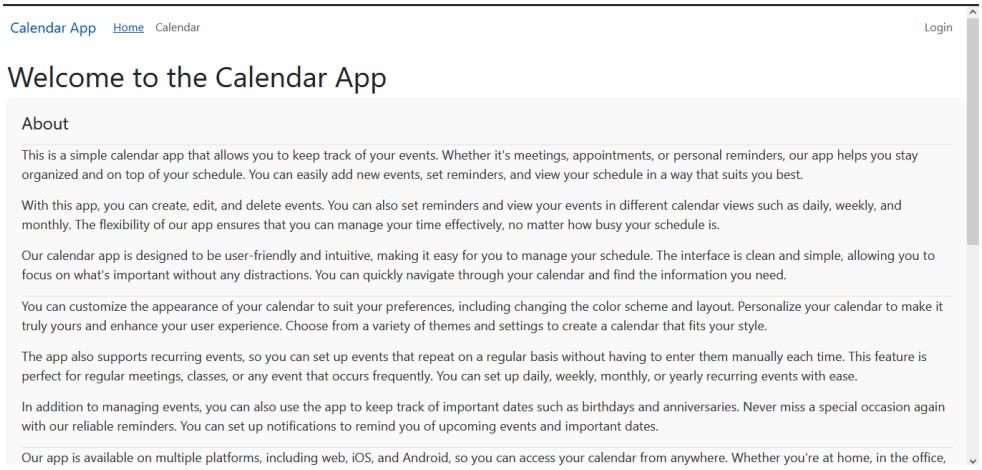


Рисунок 3.8 – Домашня сторінка веб-сайту

На рис 3.8 ми можемо бачити домашню сторінку данного веб сайту, де немає нічого особливого крім тексту та верхньої частини сторінки, де ми можемо зайти в свій аккаунт або перейти до календаря:

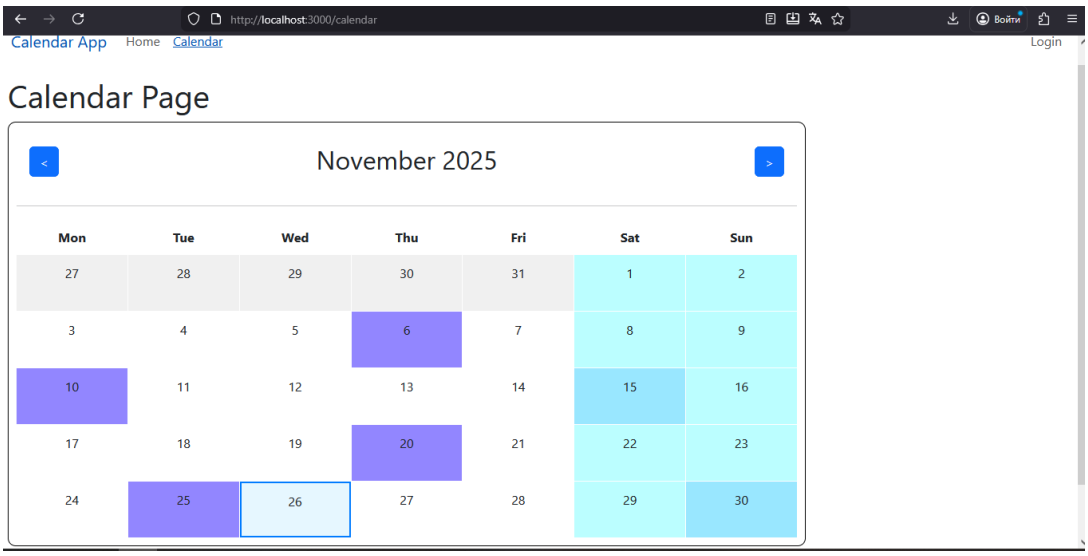


Рисунок 3.9 – Сторінка з календарем

На рис 3.9 ми можемо бачити сам календар та дати що мають звичайну версію(без подій) та дату з подією(обведена). Якщо дата з подією то зазвичай вона обведена фіолетовим кольором.

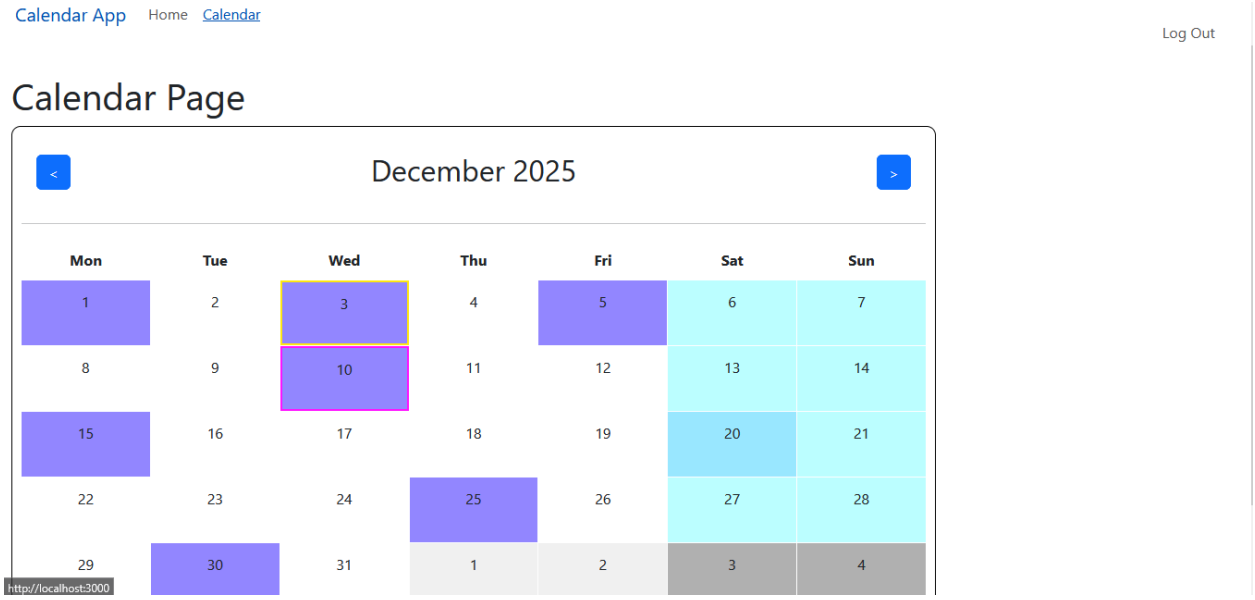


Рисунок 3.10 – Сторінка з календарем та авторизованим користувачем

Якщо ж користувач буде авторизованим то ми зможемо побачити на Рис 3.10, що користувач має змогу підписатися на подію, після чого рамка підписаної дати змінить свій колір на жовтий, а рекомендовані події будуть обведені червоним.

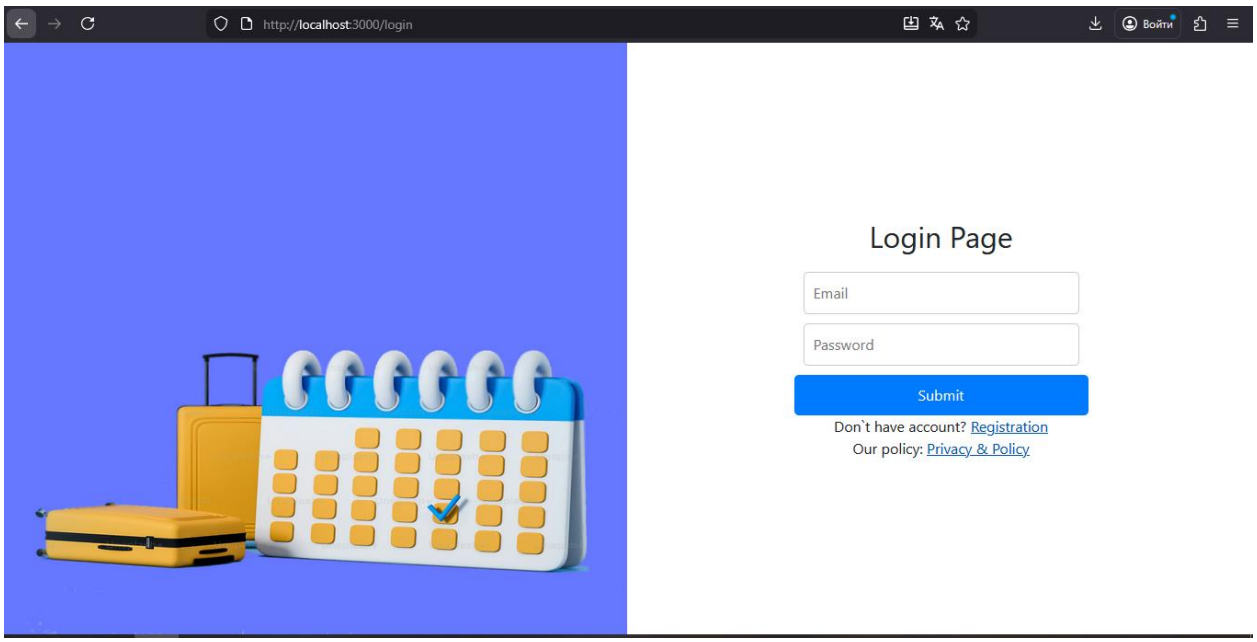


Рисунок 3.11 – Сторінка для автентифікації користувача

На рис 3.11 ми можемо бачити сторінку входу, де користувач вводить свій логін та пароль щоб зайти на сайт під своїм особистим профілем. Якщо його дані неправильно введено чи в системі сталася помилка, то висвічується повідомлення, яке показує що саме пішло не так.

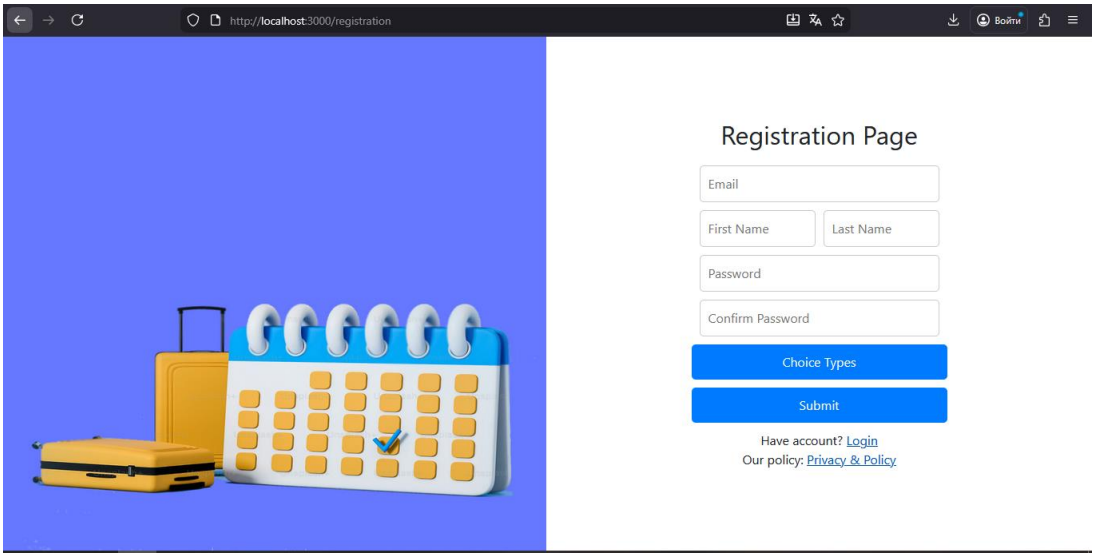


Рисунок 3.12 – Сторінка для реєстрації користувача

На рис 3.12 ми можемо бачити сторінку реєстрації, де користувач, якого ще немає в системі вводить свої данні – Логін, Електрону пошту, Пароль та Підтвердження паролю, далі ці дані перевіряються чи все введено вірно і якщо

все успішно переводить користувача на сайт. Якщо його дані неправильно введено чи в системі сталася помилка, то висвічується повідомлення, яке показує що саме пішло не так.

Також важливо уточнити що використовувати функції рекомендаційної системи можуть використовувати лише авторизовані користувачі, що допомагає не перевантажувати систему та дає змогу рекомендаційній системі працювати з більшою точністю за рахунок даних вказаних користувачами що авторизувалися

3.5 Оцінка якості створеної системи.

Тестування бекенду проводилося з метою перевірки коректності логіки роботи серверної частини додатку, зокрема, обробки запитів, взаємодії з базою даних та роботи з зовнішніми API (за наявності).

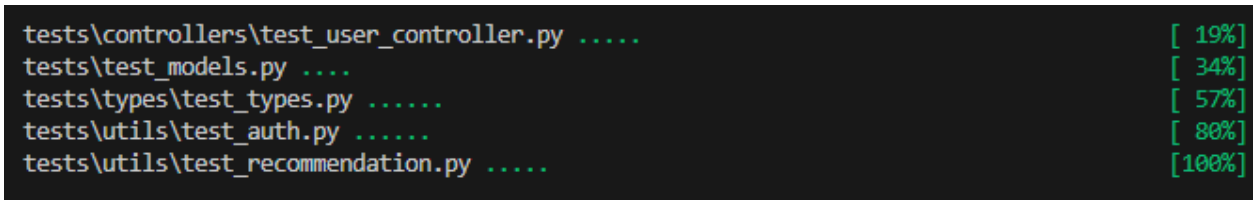


Рисунок 3.13 – Покриття компонентів backend тестами

Кожен тестовий файл супроводжується показником відсоткового тестового покриття, що відображає частину коду, яка була перевірена відповідними тестами.

Список на зображенні є важливим, оскільки він відображає рівень перевірки різних частин програмного забезпечення. Вищі показники тестового покриття означають, що певний модуль має кращу перевірку, тоді як нижчі значення можуть вказувати на те, що ще залишаються невипробувані фрагменти коду, які можуть бути схильні до помилок або непередбачуваної поведінки тому їх важко тестувати.

Result ▲	Test	Duration	Links
Passed	tests/controllers/test_user_controller.py::test_login_success	00:00:01	
Passed	tests/controllers/test_user_controller.py::test_login_invalid_password	00:00:01	
Passed	tests/controllers/test_user_controller.py::test_registration_duplicate_user	3 ms	
Passed	tests/controllers/test_user_controller.py::test_logout_user_valid	5 ms	
Passed	tests/controllers/test_user_controller.py::test_get_user_details_success	524 ms	
Passed	tests/test_models.py::test_event_instance_creation	1 ms	
Passed	tests/test_models.py::test_token_blacklist_repr	0 ms	
Passed	tests/test_models.py::test_user_instance_creation	0 ms	
Passed	tests/test_models.py::test_user_details_to_dict	1 ms	
Passed	tests/types/test_types.py::test_popularity_enum_members	0 ms	
Passed	tests/types/test_types.py::test_popularity_enum_values	0 ms	
Passed	tests/types/test_types.py::test_popularity_is_enum	0 ms	
Passed	tests/types/test_types.py::test_event_type_enum_members	0 ms	
Passed	tests/types/test_types.py::test_event_type_enum_values	0 ms	
Passed	tests/types/test_types.py::test_event_type_is_enum	1 ms	
Passed	tests/utils/test_auth.py::test_token_required_valid_token	5 ms	
Passed	tests/utils/test_auth.py::test_token_required_missing_token	2 ms	
Passed	tests/utils/test_auth.py::test_token_required_invalid_token	2 ms	
Passed	tests/utils/test_auth.py::test_token_required_expired_token	2 ms	
Passed	tests/utils/test_auth.py::test_token_required_blacklisted_token	2 ms	
Passed	tests/utils/test_auth.py::test_token_required_user_not_found	2 ms	
Passed	tests/utils/test_recommendation.py::test_prepare_data	5 ms	
Passed	tests/utils/test_recommendation.py::test_train_model	43 ms	
Passed	tests/utils/test_recommendation.py::test_recommend_returns_ids	4 ms	
Passed	tests/utils/test_recommendation.py::test_recommend_raises_if_not_trained	2 ms	
Passed	tests/utils/test_recommendation.py::test_recommend_raises_if_user_not_found	3 ms	

Рисунок 3.14 – Список результатів тестування backend

На зображенні представлений список результатів тестування різних функцій і модулів програмного забезпечення. Кожен рядок містить результат тесту, назву тесту, час його виконання та колонку з посиланнями, яка в даному зображенні порожня. Усі тести успішно пройдені, що підтверджується статусом "Passed" у колонці "Result".

Перелік тестів охоплює різні аспекти роботи програмного забезпечення, включаючи:

— Функції контролера користувача: перевірка коректності обробки запитів користувачів, включаючи створення, редагування та видалення даних.

— Функції моделей: тестування взаємодії між різними моделями даних, їх створення, оновлення та видалення.

— Функції типів: перевірка коректності роботи з різними типами даних, включаючи їх конвертацію та валідацію.

— Функції автентифікації: тестування процесу входу, реєстрації та управління сесіями користувачів.

— Функції рекомендацій: перевірка алгоритмів генерації рекомендацій на основі даних користувачів.

Час виконання тестів значно варіюється:

— Миттєві тести: завершуються за 0 мс, що свідчить про високу продуктивність.

— Тести середньої тривалості: виконуються за 100-300 мс, що є прийнятним показником для більшості функцій.

— Тривалі тести: можуть тривати до 524 мс, що вказує на складність або об'ємність перевірки.

Тестування бекенду показало, що серверна частина коректно обробляє запити, відповідає на них з правильними даними та коректно взаємодіє з базою даних. Виконання модульних та інтеграційних тестів забезпечило високий рівень покриття коду, що дозволило виявити потенційні проблеми на ранніх етапах розробки. Пройдені тести підтвердили правильність реалізації бізнес-логіки, а також ефективну обробку помилок і несподіваних ситуацій.

Тестування фронтенду проводилось з метою перевірки стабільності та коректності роботи користувацького інтерфейсу додатку. Основна увага була зосереджена на перевірці компонентів React, їхньої реакції на зміни пропсів, коректності рендерингу та обробки подій користувача.

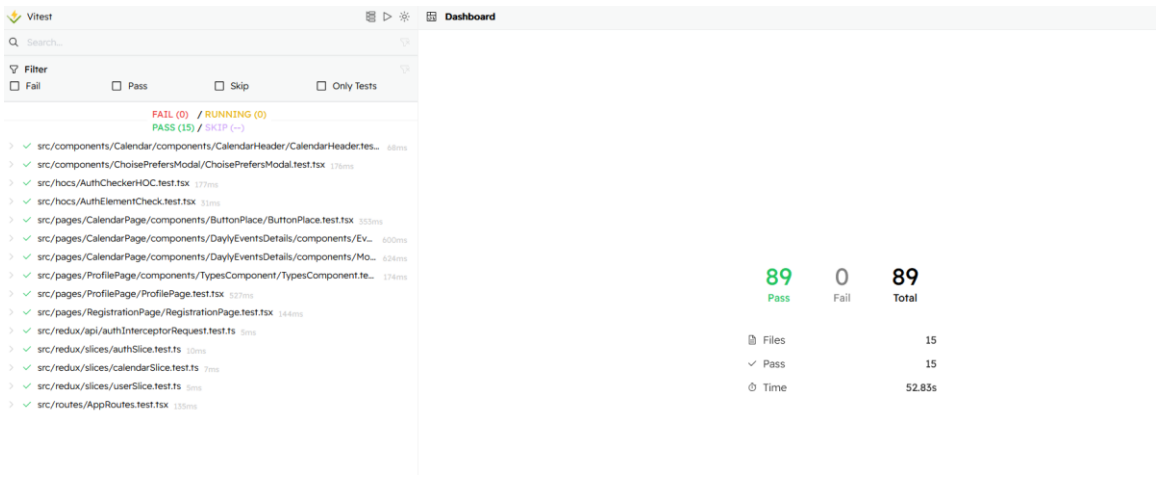


Рисунок 3.15 – Список результатів тестування frontend

Тестування фронтенду показало, що користувацький інтерфейс працює стабільно, а всі компоненти відображаються та функціонують відповідно до вимог. Модульні тести підтвердили, що зміни у вигляді або функціональності компонентів не порушують загальну стабільність системи. Тести також підтвердили правильність реакції інтерфейсу на різні сценарії взаємодії користувача, включаючи відсутність даних чи помилки API.

При аналізі тестування можна підтвердити високий рівень якості створеної системи, де як серверна, так і клієнтська частини функціонують згідно з вимогами та очікуваннями. Система успішно відпрацьовує приписанні їй модульні тести та має добре покриття тестами як backend, так і frontend частини сайту.

ВИСНОВОК

Розробка програмного забезпечення рекомендаційної системи для сайту інформування про події є цікавим досвідом для дослідження цієї теми та також для підвищення якості користувацького досвіду та ефективного управління інформацією. Така система дозволяє автоматизовано аналізувати переваги користувачів, забезпечуючи персоналізований підбір заходів, що відповідають їх інтересам. Крім того, рекомендаційна система сприяє оптимізації пошуку та навігації по сайту, що робить доступ до актуальної інформації швидким та зручним.

Використання TypeScript при розробці програмного забезпечення рекомендаційної системи для інформаційного сайту забезпечує високу надійність та масштабованість проекту. Статична типізація сприяє виявленню помилок на ранніх стадіях розробки, що значно підвищує якість кінцевого продукту, а також полегшує підтримку та розширення функціоналу системи. Завдяки інтуїтивно зрозумілій структурі та сучасним можливостям мови, TypeScript дозволяє ефективно інтегрувати різноманітні бібліотеки та фреймворки, що робить розробку адаптивних та високопродуктивних веб-додатків більш зручною та безпечною.

На основі зазначених технологій можна створити систему, що забезпечує ефективне управління та персоналізацію контенту для інформаційного сайту з календарем подій. Така система зможе ефективно аналізувати дані, формувати персоналізовані рекомендації та забезпечувати швидкий доступ до актуальної інформації, що в сукупності підвищує зручність користування сайтом та конкурентоспроможність платформи на ринку веб-послуг.

ПЕРЕЛІК ПОСИЛАНЬ

1. The Events Calendar | Calendar and tickets for WordPress. Режим доступу https://theeventscalendar.com
2. Francesco Ricci · Lior Rokach · Bracha Shapira · Paul B. Kantor. Recommender Systems Handbook. Режим доступу https://d1wqtxts1xzle7.cloudfront.net/32978074/Recommender_systems_handbook.pdf?1738107746=&response-content-disposition=inline%3B+filename%3DEditors.pdf&Expires=1747323696&Signature=cOzV2WKIZKj2oZvNqCoXBGC2qIN5DddpMiBbq3Bt6EdjjMVVIoWLYJwx1Hw6XbdPzevSs1RlFp2ztWiKuySGWc6BP~UiaLVDQE5DP9hahfIGyRZzbfws9qzPj4~7Kf9uBsHcazXEe6cOq6GAZWG~IvIEUJ8~eVH0t8AVH48epHgvWjzZKG3WOQeO4tjpiSB1GQiQe2aCCblATQpRc9h3W5nK8w9-supo0Es9CFuDQDXpcKuzJpXrH9f9jkZQp47a~Zam8lr1yHN4j7MsMSuISroB6P6jSnrwwO9tXR~QINuFogu80KHJFs~XrtoS5H6kiIz2QOnMuPdL8WdzTvGbUA__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA
3. unittest — Unit testing framework Режим доступу https://docs.python.org/uk/3.12/library/unittest.html
4. Sharvil Katariya, Joy Bose, M Vinod Reddy, Amritansh Sharma, Shambhu Tappashetty. A Personalized Health Recommendation System based on Smartphone Calendar Events. Режим доступу https://d1wqtxts1xzle7.cloudfront.net/56896352/ICOST2018_paper_44-libre.pdf?1530342645=&response-content-disposition=inline%3B+filename%3DA_Personalized_Health_Recommendation_Sys.pdf&Expires=1747324481&Signature=RLtFFKXGXkXBshMlkCJXabdll~VLBa-SmReRlvy9D8CMLFchqe-lJiHXhSgPl~~bUNT3kseJq8lP0zexStBMdCcvVi1DfdK8qTuLusNLK5FPSnhf17K2eiuIKGH-bcsFvnWDFY1mPPxppue3k5iaE1wabZJt2H668Go6AItGCpJuRh05yJTS1oXkQ~N0Z9DrIgY1IdMuzuMTonJFsP3eKNCBxU0jyp5EgqY-cEnAC2cuc2CiAI1YbA5ggye8NB9QQsE4LRPixGtSEKIQ8rzIP-CCuVneWSpHU96ZmSw-o-htNoNyZj2lhx4TJPMcCOmKSR8EiHqAoFI22cJ3I6tnTg__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA
5. Jinseok Nam, Jungi Kim, Eneldo Loza Menc´ıa, Iryna Gurevych, and Johannes F¨urnkranz. Large-Scale Multi-label Text Classification — Revisiting Neural Networks. Режим доступу https://link.springer.com/chapter/10.1007/978-3-662-44851-9_28

Додаток А

recomendation.py

```
import tensorflow as tf
import numpy as np
from app.models.user_details import UserDetails
from app.models.event import Event
from app.database import db

class RecommendationModel:
    def __init__(self):
        self.model = None
        self.user_ids=[]
        self.event_ids=[]
    def prepare_data(self):
        users = UserDetails.query.all()
        events = Event.query.all()
        if not users or not events:
            return [], [], np.array([])
        self.user_ids = [user.id for user in users]
        self.event_ids = [event.id for event in events]
        user_id_to_index = {uid: idx for idx, uid in enumerate(
self.user_ids)}
        event_id_to_index = {eid: idx for idx, eid in
enumerate(self.event_ids)}
        user_event_matrix = np.zeros((len( self.user_ids),
len(self.event_ids)), dtype=np.float32)
        for user in users:
            print(f"User {user.id} events:", [e.id for e in user.events])
        for user in users:
            u_idx = user_id_to_index.get(user.id)
            if u_idx is None:
                continue
            for event in user.events:
                e_idx = event_id_to_index.get(event.id)
                if e_idx is not None:
                    user_event_matrix[u_idx, e_idx] = 1.0
        return self.user_ids, self.event_ids, user_event_matrix
    def train(self,epoch=10, batch_size=32):
        user_ids, event_ids, user_event_matrix = self.prepare_data()
        if len(user_ids) == 0 or len(event_ids) == 0:
            raise ValueError("No users or events for training this model")
        user_indices, event_indices = np.where(user_event_matrix > 0)
        labels = user_event_matrix[user_indices,
event_indices].astype(np.float32)
        labels = np.clip(labels, 0.0, 1.0)
        if len(labels) == 0:
            print("No relates for training. Skip training")
            return
        num_users = len(user_ids)
        num_events = len(event_ids)
```

```
        user_input = tf.keras.layers.Input(shape=(1,))
        event_input = tf.keras.layers.Input(shape=(1,))
        user_embedding = tf.keras.layers.Embedding(num_users, 50,
embeddings_initializer='he_normal')(user_input)
        event_embedding = tf.keras.layers.Embedding(num_events, 50,
embeddings_initializer='he_normal')(event_input)
        dot_product = tf.keras.layers.Dot(axes=2)([user_embedding,
event_embedding])
        output = tf.keras.layers.Flatten()(dot_product)
        self.model = tf.keras.models.Model(inputs=[user_input,
event_input], outputs=output)
        self.model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])
        dataset = tf.data.Dataset.from_tensor_slices(((user_indices,
event_indices), labels))
        dataset = dataset.repeat().batch(batch_size)
        steps_per_epoch = max(1, len(labels) // batch_size)
        self.model.fit(dataset, epochs=epoch,
steps_per_epoch=steps_per_epoch, verbose=1)
    def recommend(self, user_id, top_n=5):
        if not self.model:
            raise ValueError("Model is not trained yet.")
        user_ids, event_ids, _ = self.prepare_data()
        if user_id not in user_ids:
            raise ValueError(f"User ID {user_id} not found.")
        user_index = user_ids.index(user_id)
        event_scores = self.model.predict([np.array([user_index] *
len(event_ids)), np.arange(len(event_ids))])
        top_event_indices = np.argsort(event_scores.flatten())[::-
1][:top_n]
        recommended_event_ids = [event_ids[i] for i in top_event_indices]
        return recommended_event_ids
recommendation_model = RecommendationModel()
def train_recommendation_model():
    recommendation_model.train()
    return recommendation_model
def recommend_events(user_id, model, top_n=5):
    return model.recommend(user_id, top_n)
```

Додаток Б

Посилання на backend частину веб-сайту: https://github.com/TiMur-know/Event-Calendar-with-Recomend-System-Back
Посилання на frontend частину веб-сайту: https://github.com/TiMur-know/Event-Calendar-with-Recomend-System-Frontend
Посилання на об’єднанний веб-сайт: https://github.com/TiMur-know/Event-Calendar-with-Recomend-System-Fullstack