Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук

**Бакалаврська дипломна робота на тему:**

WEB-ресурс для надання рекомендацій в настільних іграх

Виконав: студент 4 курсу,групи 3КН-20б
спеціальності 122 Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Колісник С.М.
(прізвище та ініціали)

Керівник: к.т.н., доцент каф. КН
Сілагін О. В.
(прізвище та ініціали)
« 07 » 06 2024 р.

Рецензент: к.т.н., доцент каф. САІТ
Жуков С. О.
(прізвище та ініціали)
« 07 » 06 2024 р.

Допущено до захисту
Зав. кафедри Яровий А.А.
« 07 » 06 2024 р.

Вінниця ВНТУ – 2024 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук
Рівень вищої освіти перший (бакалаврський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 122 Комп'ютерні науки
Освітньо-професійна програма – Комп'ютерні науки

**ЗАТВЕРДЖУЮ**
Завідувач кафедри КН
проф., д.т.н. Яровий А. А.
«07» 03 2024 р

**ЗАВДАННЯ**
**НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ**
Колісникy Сергію Миколайовичу

1. Тема роботи: WEB-ресурс для надання рекомендацій в настільних іграх.
Керівник БДР: Сілагін Олексій Віталійович, к.т.н., доц.
Затверджені наказом вищого навчального закладу "11" 03 2024 року №80
2. Термін подання студентом роботи 27.05.2024 р.
3. Вихідні дані до роботи:
- Дизайн інтерфейс в ціаново-оранжевих тонах, інтуїтивно зрозумілий, шрифт Space Grotesk;
- Функціональні вимоги: реєстрація та авторизація користувачів; перегляд та фільтрація настільних ігор; отримання рекомендацій в настільних іграх;
- Нефункціональні вимоги: збереження паролів у вигляді хешу; загальний час відгуку не перевищує 0.5 секунди, час відгуку при отриманні рекомендацій не перевищує 3 секунди ;
- Реляційна база даних;
- Відповідність стандарту ODBC.
4. Зміст бакалаврської дипломної роботи:
– Обгрунтування доцільності розробки WEB-ресурсу для надання рекомендацій в настільних іграх
– Аналіз відомих програмних рішень для надання рекомендацій в настільних іграх
– Проектування WEB-ресурсу для надання рекомендацій в настільних іграх
– Реалізація WEB-ресурсу для надання рекомендацій в настільних іграх
– Тестування WEB-ресурсу для надання рекомендацій в настільних іграх
5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень):
– схема алгоритму роботи WEB-ресурсу для надання рекомендацій в настільних іграх;

– діаграми варіантів використання WEB-ресурсу;
– діаграма класів рекомендаційного сервісу;
– приклад роботи програми.

6. Консультанти розділів роботи

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| 1-3 | Сілагін О. В., к.т.н., доц. каф. КН | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

7. Дата видачі завдання 07.03.2024

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва та зміст етапу | Термін виконання | | Примітка |
|---|---|---|---|---|
| | | початок | закінчення | |
| 1 | Обґрунтування доцільності розробки WEB-ресурсу для надання рекомендацій в настільних іграх | 06.05.24 | 13.05.24 | |
| 2 | Аналіз відомих програмних рішень для надання рекомендацій в настільних іграх | 13.05.24 | 20.05.24 | |
| 3 | Проектування WEB-ресурсу для надання рекомендацій в настільних іграх | 20.05.24 | 27.05.24 | |
| 4 | Реалізація WEB-ресурсу для надання рекомендацій в настільних іграх | 27.05.24 | 03.06.24 | |
| 5 | Тестування WEB-ресурсу для надання рекомендацій в настільних іграх | 03.06.24 | 06.06.24 | |

**Студент** _____ (підпис) **Колісник С.М.** (прізвище та ініціали)

**Керівник БДР** _____ **Сілагін О. В.**

# АНОТАЦІЯ

Бакалаврська кваліфікаційна робота складається з 87 сторінок формату А4, на яких є 27 рисунків, список використаних джерел містить 25 найменувань.

Бакалаврська дипломна робота присвячена створенню WEB-ресурсу для надання рекомендацій в настільних іграх. В роботі проаналізовано ресурси аналоги та наведено їх переваги та недоліки. Обґрунтовано вибір архітектури WEB-ресурсу, а також розглянуті його основні функції. Розроблено алгоритм роботи даного WEB-ресурсу. WEB-ресурс розроблено в середовищі Visual Studio Code та використано такі мови програмування, як JavaScript та Python. Також використано фреймворки: Express та React. Реалізовано мікросервіс рекомендацій, який використовує модель Two Tower.

Ключові слова: WEB-ресурс, настільні ігри, рекомендації, модель Two

## ABSTRACT

The bachelor's thesis is devoted to the creation of a web resource for providing recommendations in board games. The paper analyzes analogous resources and presents their advantages and disadvantages. The choice of the architecture of the WEB-resource is substantiated, and its main functions are considered. An algorithm for the operation of this WEB-resource has been developed. The WEB-resource is developed in the Visual Studio Code environment and uses such programming languages as JavaScript and Python. Also used frameworks: Express and React. A recommendation microservice using the Two Tower model has been implemented.



Keywords: WEB-resource, board games, recommendations, Two Tower model.

1 ОБҐРУНТУВАННЯ ДОЦІЛЬНОСТІ РОЗРОБКИ WEB-РЕСУРСУ ДЛЯ НАДАННЯ РЕКОМЕНДАЦІЙ
1.2 Аналіз відомих програмних рішень щодо надання рекомендацій в
2.1 Обґрунтування вибору моделі Two Tower для надання рекомендацій
2.2 Обґрунтування вибору архітектури WEB-ресурсу для надання
2.2 Обґрунтування вибору бібліотек та фреймворків для розробки WEB-
.....
.....
.....

# ВСТУП

**Актуальність досліджень.** Сучасний Інтернет стрімко розвивається, залучаючи мільярди користувачів по всьому світу. Однак з ростом обсягів даних та кількості користувачів мережа стикається з проблемою перенасичення інформацією. Це призводить до того, що дані часто усереднюються для того, щоб залучити якомога ширшу аудиторію. Внаслідок цього, знайти дійсно персоналізовану інформацію, яка відповідає індивідуальним потребам користувачів, стає дедалі складніше. Користувачі змушені задовольнятися рекомендаціями, які орієнтовані на загальні інтереси маси, а не на їхні конкретні вподобання та потреби.

Паралельно з цим активно розвивається галузь настільних ігор, які стають все більш популярними та поширеними. В умовах великого потоку інформації, користувачі часто шукають ефективні способи отримання персоналізованих рекомендацій щодо вибору настільних ігор. Це особливо важливо, оскільки сучасні настільні ігри є досить дорогими, і користувачі прагнуть знайти гру, яка відповідатиме їхнім конкретним вподобанням та інтересам, а не загальним смакам. Популярність настільних ігор зростає і разом з нею зростає потреба у точних та персоналізованих рекомендаціях.

Таким чином, проблема надання персоналізованих рекомендацій у настільних іграх є вкрай актуальною. Це питання потребує подальшого дослідження, щоб користувачі могли отримувати рекомендації, які максимально відповідають їхнім індивідуальним потребам та уподобанням. Завдяки цьому, вони зможуть більш ефективно витрачати свій час і кошти, знаходячи ідеальні для себе настільні ігри, що принесе їм максимальне задоволення та користь.

**Метою дослідження** є розширення функціональних можливостей WEB-ресурсу для надання рекомендацій в настільних іграх.

**Об’єктом дослідження** є процеси створення рекомендаційних систем у вигляді WEB-ресурсів.

**Предметом** дослідження є алгоритми та програмне забезпечення, для створення WEB-ресурсів для надання рекомендацій в настільних іграх.

**Задачі дослідження:**

1. Обґрунтування доцільності розробки WEB-ресурсу для надання рекомендацій в настільних іграх.
2. Аналіз відомих програмних рішень для надання рекомендацій в настільних іграх.
3. Проектування WEB-ресурсу для надання рекомендацій в настільних іграх.
4. Реалізація WEB-ресурсу для надання рекомендацій в настільних іграх.
5. Тестування WEB-ресурсу для надання рекомендацій в настільних іграх.

**Практична цінність.** Розроблені алгоритми та методи можуть бути використані при створенні подібних WEB-ресурсів.

**Апробація роботи.**

Робота апробувалась на наступних конференціях:

– LIII Науково-технічна конференція факультету інтелектуальних інформаційних технологій та автоматизації (2024), доповідь «Розробка WEB-ресурсу для надання рекомендацій в настільних іграх»;

– «Молодь в науці: дослідження, проблеми, перспективи (МН-2024), 2024р.», доповідь «Рекомендаційна система на основі моделі Two Tower».

**Публікації:** електронні тези LIII Науково-технічної конференція факультету інтелектуальних інформаційних технологій та автоматизації (2024) [1]; електронні тези конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2024), 2024р.» [2].

**Впровадження**. Результати, одержані в ході виконання бакалаврської кваліфікаційної роботи, а саме алгоритми та програмні засоби, плануються до впровадження розробки ТОВ «Намір Ентерпрайз», про що є відповідна довідка в додатку Д.

# 1 ОБҐРУНТУВАННЯ ДОЦІЛЬНОСТІ РОЗРОБКИ WEB-РЕСУРСУ ДЛЯ НАДАННЯ РЕКОМЕНДАЦІЙ В НАСТІЛЬНИХ ІГРАХ

## 1.1 Аналіз актуальності розробки WEB-ресурсу для надання рекомендацій в настільних іграх

Настільна гра - це гра, в якій учасники взаємодіють з ігровими компонентами, такими як фігурки, карти чи кубики, розміщеними на плоскій поверхні, найчастіше столі. [3].

У 20 столітті настільні ігри почали динамічно розвиватися в різних напрямках. Двома найбільшими ринками на той час були Сполучені Штати (США) та Німеччина.

З тих часів настільні ігри розвивались, створювались нові жанри, такі, як Знедавна, багато ігор почали комбінувати жанри.

Обсяг світового ринку настільних ігор у 2022 році оцінювався в 11 мільярда доларів США і очікується, що він постійно зростатиме. Прогнозується, що ринок настільних ігор досягне 30 мільярдів доларів США до 2026 року через зростання популярності та визнання серед людей, незважаючи на конкуренцію з боку джерел цифрових розваг [4].

Живучи в епоху Інтернету, з'явилося багато блогів про настільні ігри, форумів, творців контенту, що сприяє зростанню ринку настільних ігор.

Враховуючи велику зростаючу спільноту настільних ігор, дедалі більше людей шукають якісні способи знайти ідеальну настільну гру. Новітні, сучасні настільні ігри коштують недешево, оскільки зростає попит на чудово пророблений продукт та захопливий ігровий досвід. Завжди розумно провести певні дослідження щодо конкретної настільної гри перед купівлею. Зазвичай люди використовують такі форуми, як BoardGameGeek та Reddit, щоб шукати рекомендації щодо ігор. Деякі звертаються до YouTube-каналів, створених такими групами, як Dice Tower та Watch It Played, щоб отримати відеоогляди настільних ігор. Однак найчастіше ці ресурси пропонували одні

й ті самі, популярні в спільноті, настільні ігри і вони практично ніколи не були персоналізовані для кожної людини.

Тому є актуальним завдання по розробці WEB-ресурсу для надання рекомендацій в настільних іграх, який міг би враховувати індивідуальні потреби та уподобання користувачів.

**Аналіз відомих програмних рішень щодо надання рекомендацій в настільних іграх**

Останніми роками стрімкий розвиток інтернету спричиняє інформаційне перенавантаження, через що користувачеві стає складніше своєчасно отримати цікаву інформацію. Тому створюються WEB-ресурси, які надають рекомендації. Огляд наявних WEB-ресурсів з надання рекомендацій в настільних іграх є ключовим моментом для розуміння подальшого плану розробки нового WEB-ресурсу для надання рекомендацій.

Сервісів, які дозволяють отримати рекомендації в настільних іграх виявилось дуже мало, але наведемо деякі з них:

1. Board Game Recommendation Engine – веб ресурс для надання рекомендацій на основі трьох ігор, що подобаються.

Його створила «Quantic Foundry» — це компанія, що займається дослідженнями ринку і зосереджена на мотивації геймерів. За їх описом вони поєднують соціальні науки з наукою про дані, щоб зрозуміти, що рухає геймерами. Вони зібрали дані понад 1,65 мільйона геймерів. Вони використали відомі психометричні методи, щоб розробити емпіричну модель мотивації гравців [5].

В основному вони займаються наданням рекомендацій для геймерів, тобто для людей, які грають у відеоігри, але в них є розділ для надання рекомендацій і в настільних іграх.

Програми Quantic Foundry – це набір профілів гравців та інструментів, заснованих на дослідницькому проекті, проведеному давніми дослідниками ігор Ніком Йі та Ніколя Дюшено.

Їх система складається з трьох полів для уведення даних. Є інструкція, введіть три гри, які вам подобаються, і ми створимо рекомендації щодо ігор на основі даних понад 500 000 гравців. Наведемо як виглядає дана система на рисунку 1.1.

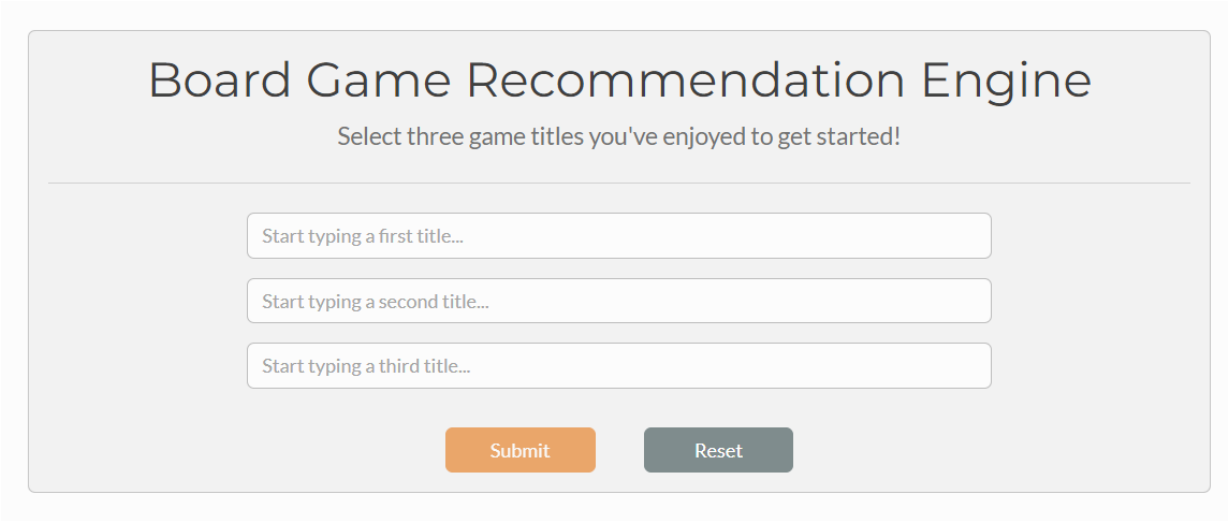



Рисунок 1.1 – Board Game Recommendation Engine

Основні переваги:

- можливість варіювати тип рекомендацій нішевий, збалансований, популярний;
- можливість фільтрувати рекомендації за роками;
- функціонал безкоштовний.

Недоліки:

- дуже довго відбувається пошук гри по назві;
- обов’язково потрібно шукати саме по трьом іграм, увести назву тільки однієї не можна;
- список рекомендацій містить тільки картинку гри та її опис, інші дані, як кількість гравців і тому подібні, не надаються;
- відсутність фільтрів по іншим важливим даним, як-от кількість гравців;
- немає можливості зберегти отримані рекомендації;
- при оновлення сторінки усі уведенні дані пропадають і процес отримання рекомендацій потрібно повторювати знову.

Trythesegames – ресурс, який зазначає, що його функціонал в наданні рекомендацій схожий, як попросити своїх знайомих порадити вам гру. Ось так розробники описують свій проєкт: «Уявіть, що ви щойно закінчили грати в настільну гру, яка вам дуже сподобалася, і запитали оточуючих, чи можуть вони порадити іншу гру. Рекомендації, імовірно, будуть надійнішими, якщо вони надходять від когось, кому також подобається гра, в яку ви щойно грали і він має широкі знання про інші настільні ігри. Якби ви могли запитати тисячі людей, які грали в гру, і порівняти їхні відповіді, враховуючи, сподобалася їм гра чи ні, можна було б розробити корисний алгоритм рекомендацій. Ця програма робить це, використовуючи оцінки, надані користувачами» [6].

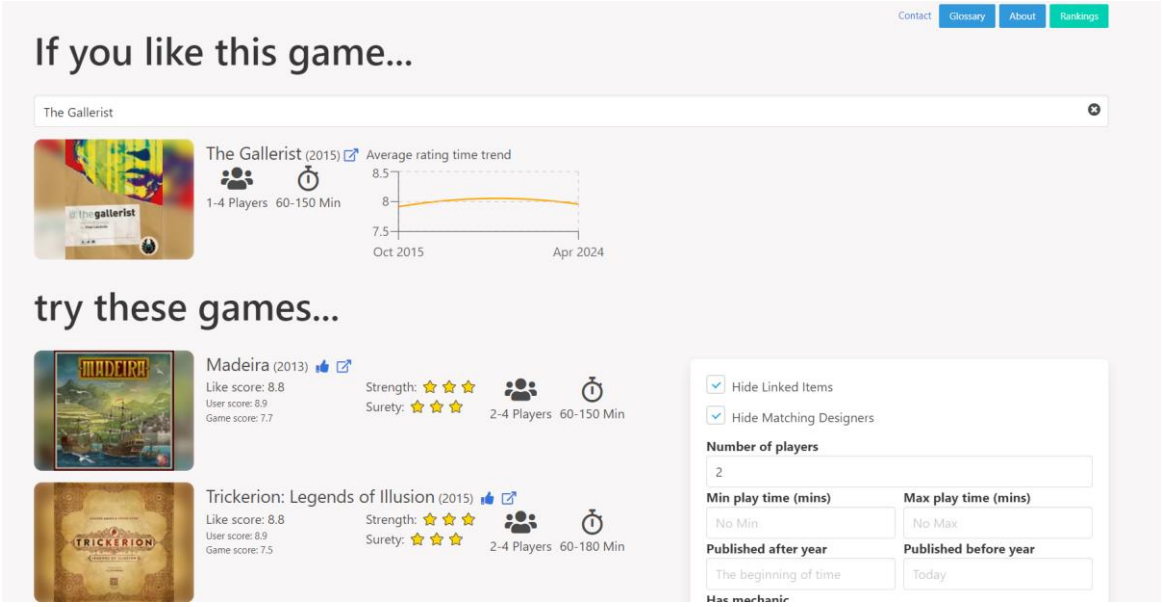



Рисунок 1.2 – Trythesegames

Основні переваги:

- виведені рекомендації містять основну інформацію про гру;
- можливість фільтрувати рекомендації з широким набором налаштувань;
- функціонал безкоштовний;
- є графік де показано зміни оцінок з часом;
- зберігає отримані рекомендації при оновленні сторінки.

Недоліки:

- рекомендації відбуваються тільки по схожості до однієї гри, більше вибрати не можна;
- у фільтрації відсутні пункти ,як-от ,наприклад, вік з якої можна грати в цю гру, або з якою кількістю гравців ця гра розкривається найкраще, є тільки можливість вибрати чи в загальному можливо грати в цю гру з такою кількістю;
- немає можливості зберегти отримані рекомендації;
- використовує колаборативну фільтрацію для отримання рекомендацій.

Отже, проаналізовані WEB-ресурси не можуть задовольнити усі потреби користувачів у певному функціоналі. Наприклад: замало параметрів для фільтрації, немає можливості зберегти отримані рекомендації та отримувати їх за більшим списком ігор, рекомендації надаються тільки на основі даних про інших користувачів, дані про характеристики ігор не враховуються. Це показує доцільність розробки WEB- ресурсу для надання рекомендацій в настільних іграх

**Формулювання вимог та постановка задачі**

Визначимо задачі і вимоги до WEB-ресурсу  для надання рекомендацій в настільних іграх. Задача: розширення функціоналу WEB-ресурсу для надання рекомендацій в настільних іграх, забезпечення користувачів місцем де вони зможуть отримувати персональні рекомендації в настільних іграх.

Перелік основних функцій WEB-ресурсу:

вторизація користувачів: Користувачі будуть мати можливість, за їх бажанням, зареєструвати профіль на даному WEB-ресурсі для зберігання ігор в свій профіль, щоб в подальшому мати змогу переглянути  список збережених ігор, коли він знадобиться і додатково отримати можливі рекомендації, які ігри їм можуть бути цікавими.

ерегляд та сортування ігор: Користувачі зможуть перегляди більше 100 тисяч ігор,  та фільтрувати їх за великою кількістю категорій, також якщо вони авторизовані вони можуть зберегти ігри собі в збережені.

тримати рекомендації в настільних іграх: Користувачі, можуть використати загальний список ігор та добавити їх в ті, що сподобались для отримання рекомендацій, або використати пошук та вибрати потрібну гру по назві, далі вони можуть отримати рекомендований список ігор, та відфільтрувати його за потреби. Авторизовані користувачі, також будуть мати можливість добавити собі в збережені те, що їх зацікавить.

Розроблений WEB-ресурс має працювати у всіх веб браузерах. Усі наявні функції повинні працювати без будь-яких помилок. Початком розробки WEB-ресурсу для надання рекомендацій в настільних іграх є аналіз відомих програмних рішень. Далі – вибір архітектури WEB-ресурсу, моделювання WEB-ресурсу з використанням мови UML та розробка узагальненого алгоритму роботи, обґрунтування вибору гібридної системи надання рекомендацій. Наступним етапом є вибір методів розробки та безпосередньо сама розробка WEB-ресурсу. Фінальним етапом є тестування розробленого WEB-ресурсу.

**Висновки**

У даному розділі було обґрунтовано доцільність розробки WEB-ресурсу для надання рекомендацій в настільних іграх. Розпочато було з аналізу актуальності даної теми, було розглянуто, як люди шукають та отримуються для себе рекомендації в іграх, який вплив на суспільство може мати створення даного ресурсу. В результаті аналізу було чітко зрозуміло, що потреба в даному WEB-ресурсів присутня.

При огляді існуючих WEB-ресурсів було проаналізовано два WEB-ресурси, які уже надають рекомендації та знайшли своїх користувачів. Були розглянуті WEB-ресурси «Board Game Recommendation Engine», «Trythesegames». Кожен з ресурсів має свої переваги та недоліки.

Було виконано постановку задачі та сформовано функціонал розроблюваного WEB-ресурсу.

# 2 ПРОЕКТУВАННЯ WEB-РЕСУРСУ ДЛЯ НАДАННЯ РЕКОМЕНДАЦІЙ В НАСТІЛЬНИХ ІГРАХ

## Обґрунтування вибору моделі Two Tower для надання рекомендацій в настільних іграх

Система рекомендацій може бути персоналізованою або неперсоналізованою. Неперсоналізована система може бути простішою, але персоналізована система працює краще, оскільки вона задовольняє потреби кожного окремого користувача

Сучасні системи рекомендацій в більшості є персоналізованими і їх можна умовно поділити на три категорії:

1. Колаборативна фільтрація(Collaborative Filtering) - рекомендує елементи на основі показників подібності між користувачами або елементами. Основне припущення алгоритму полягає в тому, що користувачі зі схожими інтересами мають спільні переваги. Існує два основних типи колаборативної фільтрації [7]:

- Основана на користувачах: ми шукаємо користувачів, які однаково оцінили різні елементи, а потім визначаємо оцінку відсутнього елемента за допомогою їхніх оцінок.
- Основана на предметах: ми аналізуємо зв’язки між парами предметів (користувач, який купив предмет А, також купив предмет Б) і визначаємо відсутню оцінку на основі оцінок, наданих іншими користувачами.

2. Контент-базована фільтрація (Contend-based Filtering): Працює на схожості між продуктами. Спочатку ми повинні створити вектор, що представляє всі характеристики продукту. Потім ми обчислюємо схожість між цими векторами за допомогою таких методів, як [8]:

- евклідова відстань;
- манхеттенська відстань;
- відстань Жаккара;
- косинусна відстань (або косинусна схожість).

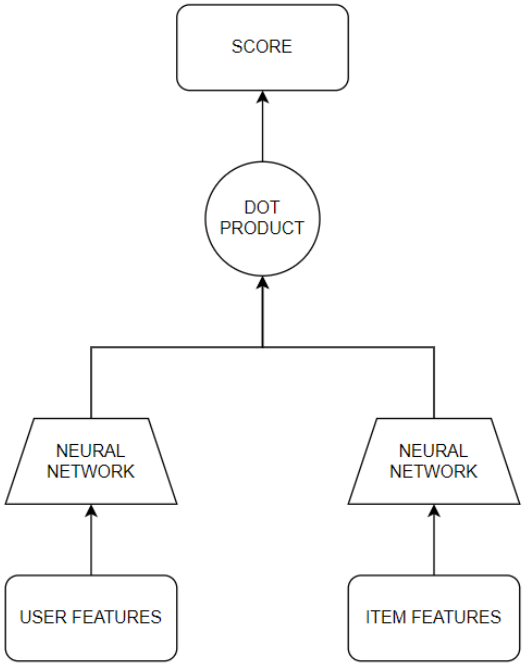
Гібридні системи

В більшості своїй новітні гібридні системи є кращими, бо вирішують низку проблем, які наявні в колаборативних або контент-базованих системах, такі як ефект «холодного старту» і тд. Все це відбувається завдяки використанню нейронних мереж.

Існує кілька архітектур рекомендацій на основі нейронних мереж. Однак, якщо контекстна інформація про наших користувачів доступна лише під час запиту. Це означає, що ми повинні обчислювати рекомендації в реальному часі без великих затримок. Отже, нам потрібна дуже ефективна система.

На щастя, є одна архітектура, яка відповідає всім цим вимогам – Two Tower.

Ідея архітектури Two Tower полягає в тому, що є дві окремі нейронні мережі, які називаються вежами. Можемо інтерпретувати ці дві вежі як окремі моделі, одна з яких представляє користувачів, відома як вежа запитів, а інша - предмети, відома як вежа кандидатів. Під час навчання кожна вежа вчиться перетворювати довільний набір вхідних ознак у вектори [9]. Архітектура моделі зображена на рисунку 2.1.




Рисисунок 2.1 – Архітектура моделі Two Tower

На рисунку 2.1 показано, що інформація про користувача та об'єкти взаємодіє тільки на фінальному етапі роботи системи. Це дозволяє ефективно навчати модель

та робить архітектуру Two Tower ідеальним вибором для отримання рекомендацій у реальному часі.

Під час виведення рекомендацій можна використати факт, що вектори вбудовування об'єктів (embeddings) є фіксованими. Після навчання моделі ці векторні представлення об'єктів не змінюються. Їх обчислює «вежа об'єктів» на етапі тренування і потім фіксує. Завдяки цьому під час виведення рекомендацій для користувача можна попередньо обчислити всі вектори вбудовування об'єктів відразу після навчання та завантажити їх у пам'ять, уникаючи необхідності запускати нейронну мережу для обчислення цих векторів.

Однак вектори вбудовування користувача залежать від контексту, тому їх потрібно обчислювати динамічно. Проте це необхідно робити лише для одного користувача за раз, що відбувається швидко. Для підвищення продуктивності можна розраховувати їх з певною періодичністю, але це означатиме пожертвування урахуванням історії та контексту в реальному часі.

Нарешті, обчислюється скалярний добуток між завантаженими векторами вбудовування об'єктів і обчисленими вбудовуваннями користувача. Результати впорядковуються за спаданням на основі отриманих оцінок - дуже ефективна та швидка операція. Для отримання фінального результату можливі й інші математичні операції чи прості нейронні мережі замість скалярного добутку.

Можемо зробити висновок, що використання Two Tower моделі для надання рекомендацій є доцільним та ефективним підходом, що розширяє можливості надання рекомендацій за рахунок нейронних мереж та дозволяє враховувати інтереси користувачів та надавати їм рекомендації в реальному часі.

**Обґрунтування вибору архітектури WEB-ресурсу для надання рекомендацій в настільних іграх**

Вибір архітектури є вирішальним кроком у розробці WEB-ресурсу. Архітектура WEB-ресурсу визначає взаємодію між різними користувацькими інтерфейсами, ба-

зами даних, компонентами та проміжним програмним забезпеченням. Вона забезпечує каркасну структуру, яка визначає логічні взаємозв'язки між різними кінцевими точками. Також описує стандартний потік даних між клієнтом і сервером. Архітектура WEB-ресурсу надає набір правил для передачі інформації з одного кінця в інший.

На сьогоднішній день існує декілька поширених архітектур. Серед архітекрур для розробки WEB-ресурсів можна виділити монолітну архітектуру, мікросервісну архітектуру, трирівневу архітектуру.

Монолітна архітектура є традиційним підходом до розробки програмного забезпечення, коли весь додаток будується як єдиний цілісний компонент. Назва «моноліт» походить від грецького слова «monolithos», що означає «єдиний камінь», що ілюструє саму суть такої архітектури - єдиний репозиторій коду, який містить всю бізнес-логіку. Для внесення змін у додаток з монолітною архітектурою необхідно оновлювати весь стек: отримувати доступ до бази коду, вносити зміни та розгортати оновлену версію інтерфейсу на стороні сервісу (рис. 2.2). [10].

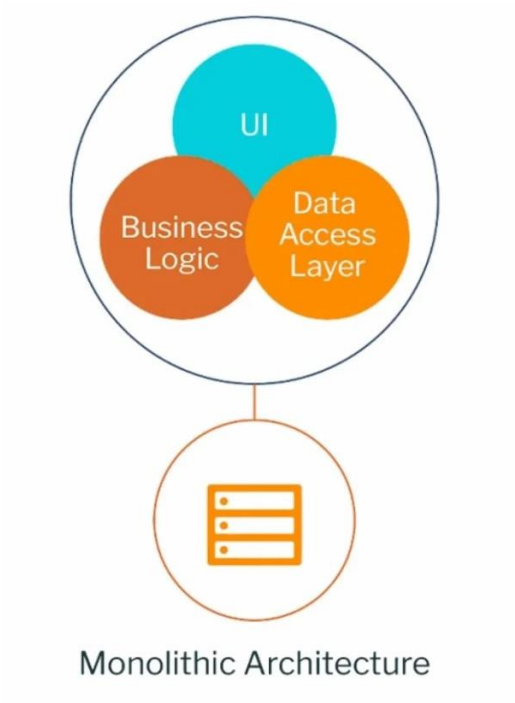



Рисунок 2.2 – Монолітна архітектура

До переваг монолітної архітектури можна віднести:

егке розгортання – розгорнути один файл простіше ніж декілька.

озробка – коли проект маленький і кодова база одна, нею легше оперувати і розробляти.

родуктивність - у централізованій базі коду та репозиторії один API часто може виконувати ту саму функцію, яку виконують декілька API з мікросервісами.

прощене тестування - оскільки монолітний додаток є єдиною централізованою одиницею, його простіше тестувати аніж розподілені системи.

егке налагодження - оскільки весь код знаходиться в одному місці, простіше відстежити і вирішити проблему.

До недоліків моноліту можна віднести:

овільніша швидкість розробки - великий монолітний додаток робить розробку складнішою і повільнішою, ускладнюється розуміння зробленого.

асштабованість – неможливо масштабувати окремі компоненти.

адійність – помилка в будь якому компоненті призводить до краху всієї системи.

ар'єр для впровадження технологій - будь-які зміни у фреймворку або мові впливають на весь додаток, що робить зміни часто дорогими та трудомісткими.

ідсутність гнучкості - моноліт обмежений технологіями, які вже були вибрані і використовуються в ньому.

озгортання - невелика зміна в монолітному додатку вимагає перерозгортання всього моноліту.

Мікросервісна архітектура - це підхід, згідно з яким застосунок розбивається на набір невеликих незалежних сервісів, кожен з яких працює у власному процесі. Ці сервіси взаємодіють між собою за допомогою легких та швидких протоколів передачі даних, зазвичай HTTP. Вони будуються навколо конкретних бізнес-вимог і розгортаються автономно в повністю автоматизованому середовищі з мінімальним централізованим керуванням. Мікросервіси можуть бути реалізовані з використанням різних мов програмування та технологій зберігання даних (рис. 2.3) [11].

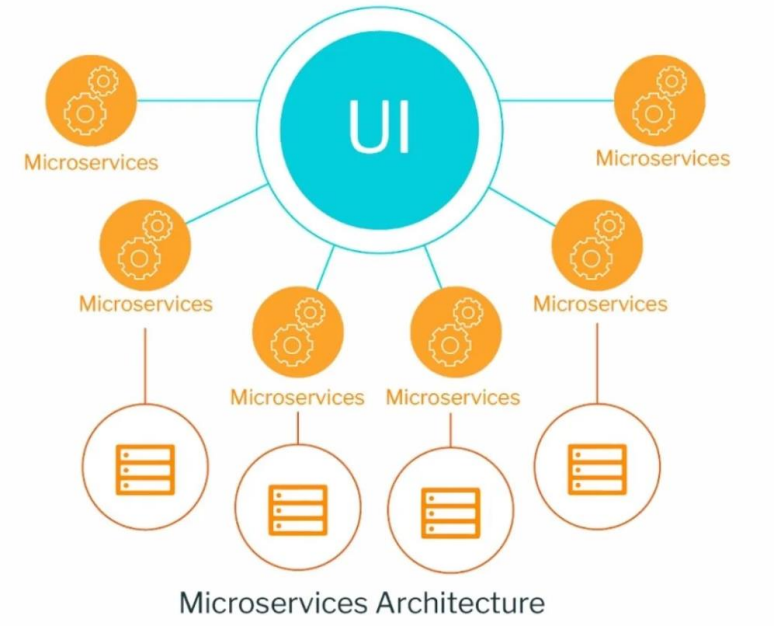



Рисунок 2.3 – Мікросервісна архітектура

Переваги мікросервісів:

нучке масштабування - якщо мікросервіс досягає свого пікового навантаження, нові екземпляри цього сервісу можуть бути швидко розгорнуті.

исокий рівень підтримки та тестування - команди можуть експериментувати з новими функціями та відкотити зміни назад, якщо щось не працює. Це полегшує оновлення коду і прискорює вихід нових функцій. Крім того, легко ізолювати та виправити несправності та помилки в окремих сервісах.

езалежне розгортання - оскільки мікросервіси є окремими блоками, вони дозволяють швидко і легко розгортати окремі функції незалежно одна від одної.

нучкість технології - мікросервісні архітектури дозволяють командам вільно обирати інструменти, які вони бажають.

До недоліків мікросервісів можна віднести:

озподіл розробки - мікросервіси додають більше складності порівняно з монолітною архітектурою, оскільки існує більше сервісів у різних місцях, створених різними командами. Без належного управління це може призвести до уповільнення розробки та погіршення операційної продуктивності.

кспоненціальне зростання витрат на інфраструктуру - кожен новий мікросервіс

може мати свої витрати на тести, сценарії розгортання, хостинг, інструменти моніторингу тощо.

ідсутність стандартизації - без спільної платформи може виникнути розмаїття мов, стандартів ведення журналів та моніторингу.

Трирівнева архітектура - це архітектура програмних додатків, яка організовує додатки на трьох логічних і фізичних обчислювальних рівнях: рівень представлення, або інтерфейс користувача; рівень програми, де обробляються дані; і рівень даних, де зберігаються і управляються дані програми [12].

WEB-ресурс матиме ті ж рівні, лише з тією різницею, що вони матимуть різні назви (рис. 2.4).

Веб-сервер: веб-сервер - це рівень презентації трирівневої архітектури, який обслуговує інтерфейс користувача (UI). Цей вміст може бути статичним або динамічним залежно від вимог.

Сервер додатків: рівень додатків - це середній рівень, який складається з повної логіки, необхідної для бізнесу, який буде обробляти вхідні дані користувачів тощо. Він виконує логіку.

Сервер бази даних: це внутрішній рівень веб-додатку, який складається з усіх даних, тому на цьому рівні здебільшого будуть бази даних, такі як MySQL, Oracle, DB2 або PostgreSQL.

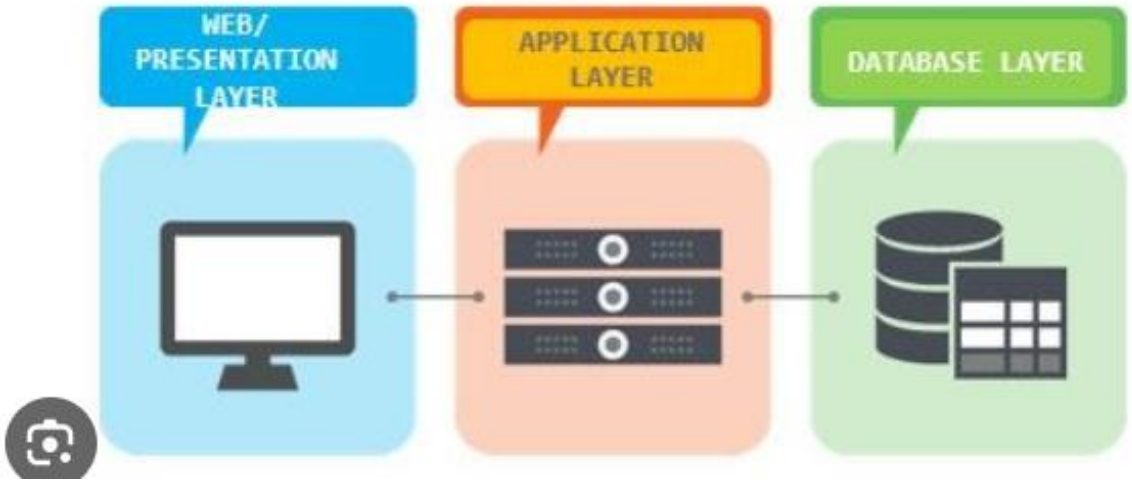



Рисунок 2.4 – Трирівнева архітектура

Переваги трирівневої архітектури

одульність: кожен рівень обробляє певний аспект програми, що робить систему більш модульною. Ця модульність полегшує розробку, тестування та обслуговування, оскільки кожен рівень можна опрацювати незалежно.

езалежне масштабування: кожен рівень можна масштабувати незалежно. Наприклад, рівень бази даних можна оптимізувати окремо від програмного рівня, що забезпечує більш ефективне управління ресурсами.

зольовані рівні: конфіденційні операції, такі як запити до бази даних, обробляються на програмному рівні. Це розділення допомагає запроваджувати найкращі методи безпеки, як-от запобігання прямому доступу до бази даних.

Недоліки трирівневої архітектури:

итрати на розробку: налаштування та підтримка трьох окремих рівнів ускладнює процес розробки. Розробникам необхідно керувати взаємодією та потоком даних між шарами.

атримка зв'язку: зв'язок між рівнями (наприклад, клієнт-сервер і сервер-база даних) може призвести до затримки. Мережні виклики між цими рівнями можуть уповільнити загальну продуктивність програми.

3. Витрати на інфраструктуру: обслуговування окремих серверів для кожного рівня може бути дорогим. Це вимагає більше апаратних ресурсів і може збільшити експлуатаційні витрати.

В результаті аналізу трьох архітектур, врахувавши їх переваги та недоліки, було вирішено надати перевагу мікросервісній архітектурі, тому що вона надасть такі можливості та переваги:

1. Незалежне розгортання: додавання рекомендаційної частини, як окремого мікросервісу, дозволить його незалежно від основної програми розробляти та розгортати.
2. Це надасть можливість використовувати потрібні технології для кожної з задач.
3. Якщо в майбутньому виникне потреба в розширенні, з мікросервісною архітектурою це не складе проблем.
4. В майбутньому це дозволяє залучити спеціалістів різних категорій до розробки.

5. Помилки ізольовані, тобто якщо зламається сервіс з надання рекомендацій, користувачі зможуть зайти в свої профілі і переглянути, що вони зберігали без проблем.

В загальному структурна схема при використанні мікросервісів для WEB-ресурсу для надання рекомендацій в настільних іграх буде виглядати як наведено на рисунку 2.5.

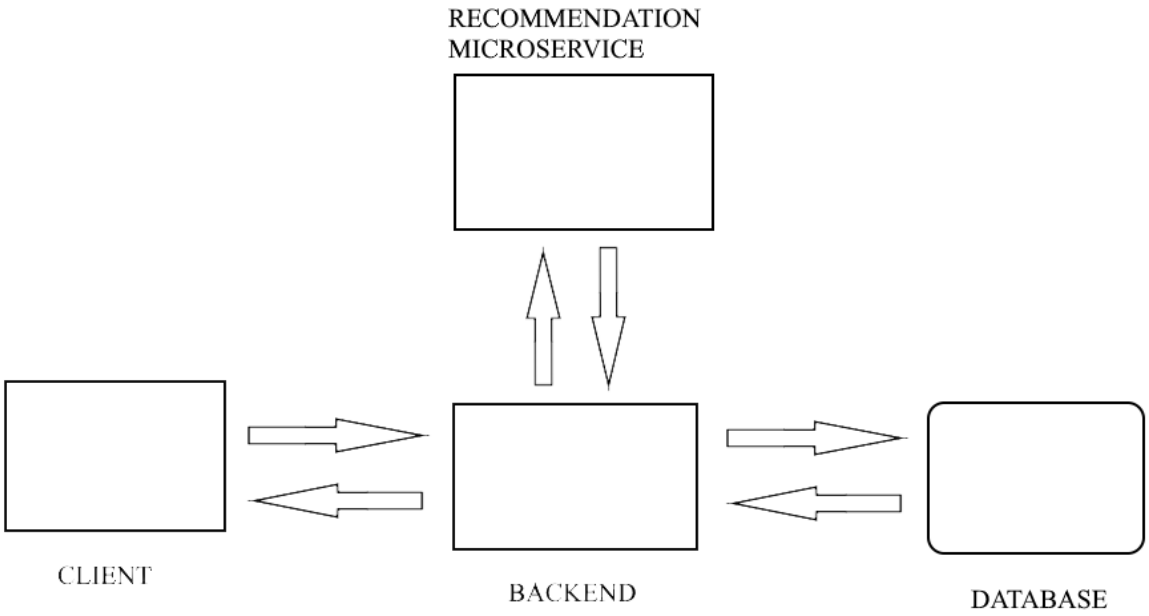



Рисунок 2.5– Схема архітектури WEB-ресурсу

**U**

**M**

**L** Для розробки WEB-ресурсу є важливим попереднє його проектування. Для

**проектування WEB-ресурсу для надання рекомендацій в настільних іграх**

того, в більшості WEB-ресурсів використовується UML. UML це уніфікована мова моделювання, що передбачає стандартизовану візуалізацію дизайну системи або його компонентів. Уніфікованість мови дозволяє всім, хто знайомий з її правилами читати і розуміти зображені діаграми.

В загальному UML діаграми поділяються на такі типи:

Структурні діаграми – діаграма класів(відображає загальну структуру та архітектуру класів) , компонентів (показує розподіл програми на структурні компоненти і

зв’язки) і об’єктів (демонструє окремі об’єкти у системі та їх взаємодії в фіксований момент часу).

Поведінкові – діаграма випадків використання (описує взаємодію між акторами і прецендентами), активностей (діаграма на якій показано дії, стани яких зображено на діаграмі станів), станів (зміна в часі станів об’єкта) [13].

Для даного проєкту буде розроблено 2 типи діаграм. З поведінкового типу буде розроблена діаграма використання, з структурного типу буде розроблена діаграма класів.

Давайте розглянемо діаграму варіантів використання більш детально. Ця діаграма, також відома як діаграма прецедентів або сценаріїв використання, дозволяє візуалізувати різні ролі та їх взаємодію з системою. Однак вона не відображає послідовність виконання кроків. Діаграма варіантів використання ілюструє функціональні вимоги системи з точки зору користувача, тобто що система може робити. [14]. Основні елементи нотації для діаграми варіантів використання зображені на рисунку 2.6.

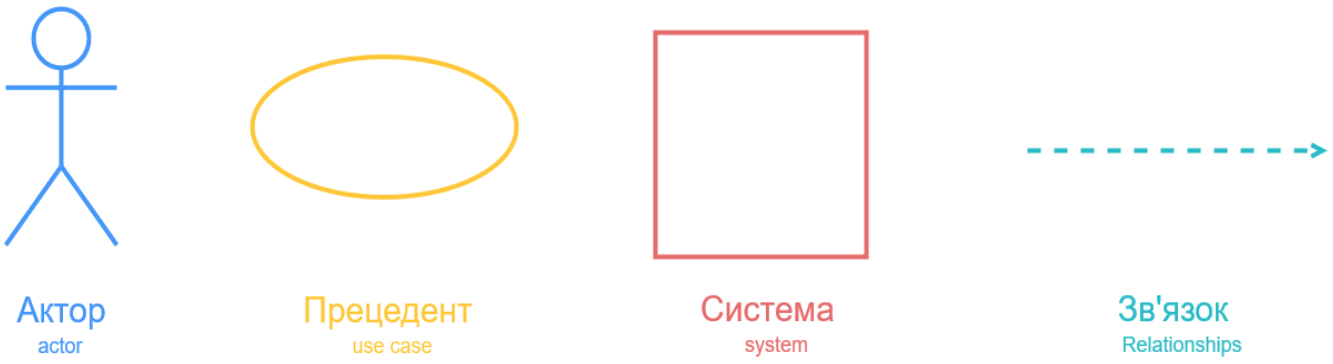



Рисунок 2.6 – Основні позначення діаграми варіантів використання

Діаграми варіантів використання містять чотири головні компоненти: акторів, варіанти використання (прецеденти), систему та зв'язки між ними.

Актор - це об'єкт, який взаємодіє із системою, проте не входить до її складу. Зазвичай актори зображуються у вигляді стилізованих фігурок або прямокутників з написом «актор».

Варіант використання (прецедент) визначає очікувану поведінку системи та відображає її функції, дії чи завдання. Він зображується у вигляді еліпса з назвою дії (дієсловом) всередині. Варіант використання вказує на те, що має статися, але не пояснює, як саме це відбудеться.

Система - це те, що моделюється, наприклад веб-сайт, мобільний додаток або модуль програмного забезпечення. Вона зображується прямокутником з назвою системи у верхній частині.

До діаграми можуть бути додані коментарі для розширення функціональності, надання підказок та умов.

Зв'язки (відносини) ілюструють взаємодію між акторами та варіантами використання. Усі актори повинні мати зв'язок з варіантами використання, але не всі варіанти використання обов'язково пов'язані з акторами. Зазвичай для зображення зв'язків використовують суцільну лінію.

Діаграма варіантів використання має чотири основні типи зв'язків:

1. Асоціація - це стандартний зв'язок між актором і варіантом використання. Позначається простою суцільною лінією. Кожен актор повинен мати хоча б один зв'язок з варіантом використання, а один варіант використання може бути пов'язаний з кількома акторами.
2. Розширення - вказує на додаткову функціональність або можливий необов'язковий сценарій поведінки системи. Позначається пунктирною лінією зі стрілкою, спрямованою на базовий варіант використання, та стереотипом <<extend>>. Базовий варіант використання має сенс сам по собі і може існувати без розширення. Розширений варіант активується лише за певних умов.
3. Включення - показує, що поведінка одного варіанту використання є частиною іншого. Це означає, що базовий варіант використання використовує включений для виконання певної операції. На відміну від розширення, включений варіант є обов'язковим для базового. Позначається пунктирною лінією зі стрілкою та стереотипом <<include>>, який вказує на включений варіант використання. Використовується для уникнення дублювання функцій.

4. Генералізація - відносини типу «батько-дитина» або успадкування між варіантами використання. Позначається суцільною лінією з незаповненим трикутником, який вказує на батьківський варіант використання.

Опишемо нашу майбутню діаграму. Акторами в нашому випадку будуть гості (не авторизовані користувачі), та авторизовані користувачі. Кожен актор має свої можливості при взаємодії з системою. У гостя будуть обмежені функціональні можливості. Але, гість зможе увійти в систему, зареєструватись, що відкриє йому додатковий функціонал у вигляді профіля, в якому він може зберігати ігри, які його зацікавили та отримувати за бажанням рекомендації.

Розроблені дві UML-діаграми варіантів використання для гостя і авторизованого користувача, зображені на рисунках 2.7, 2.8. Вони відображають можливості обох акторів та дозволяють зрозуміти їх взаємодії з системою.

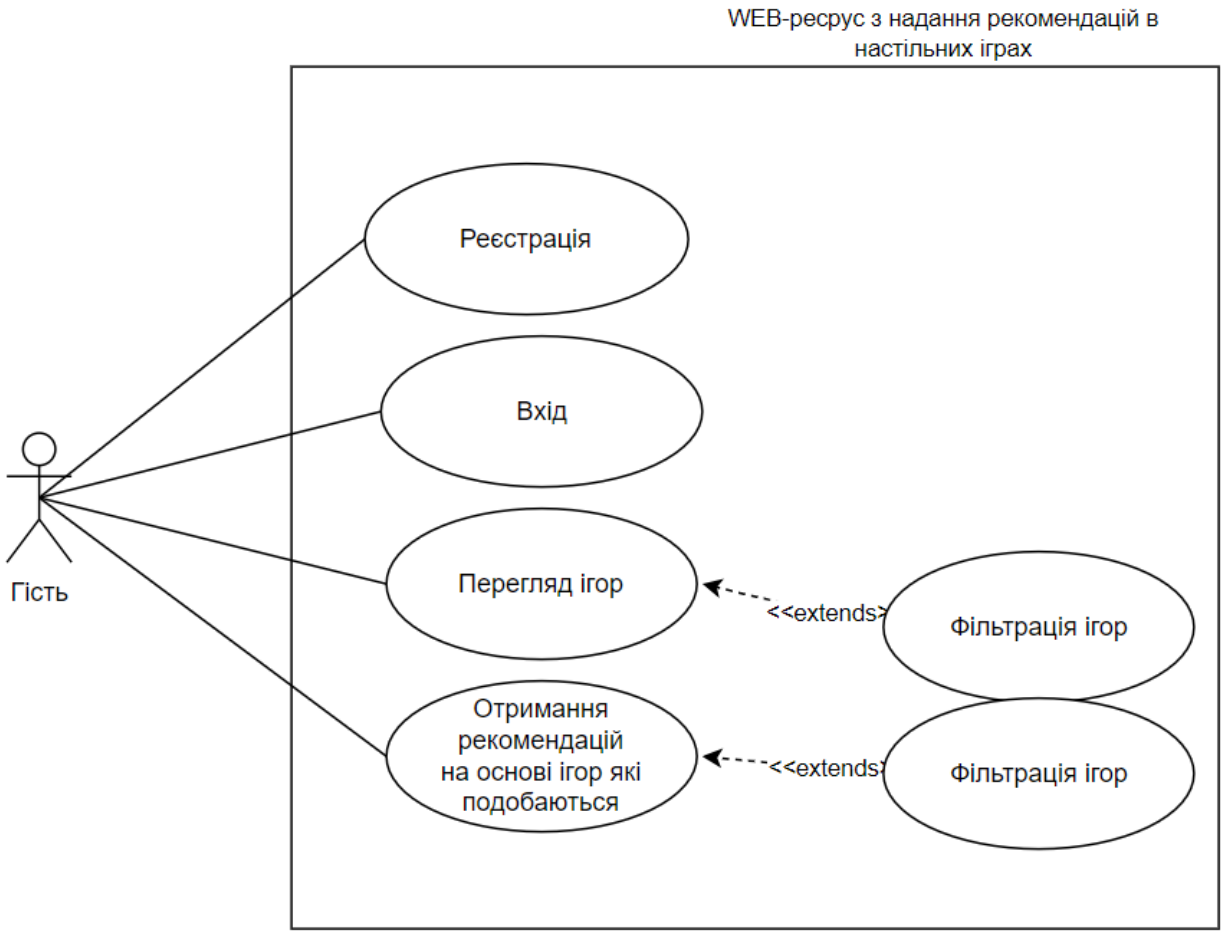



Рисунок 2.7 – Діаграма варіантів використання системи Гість

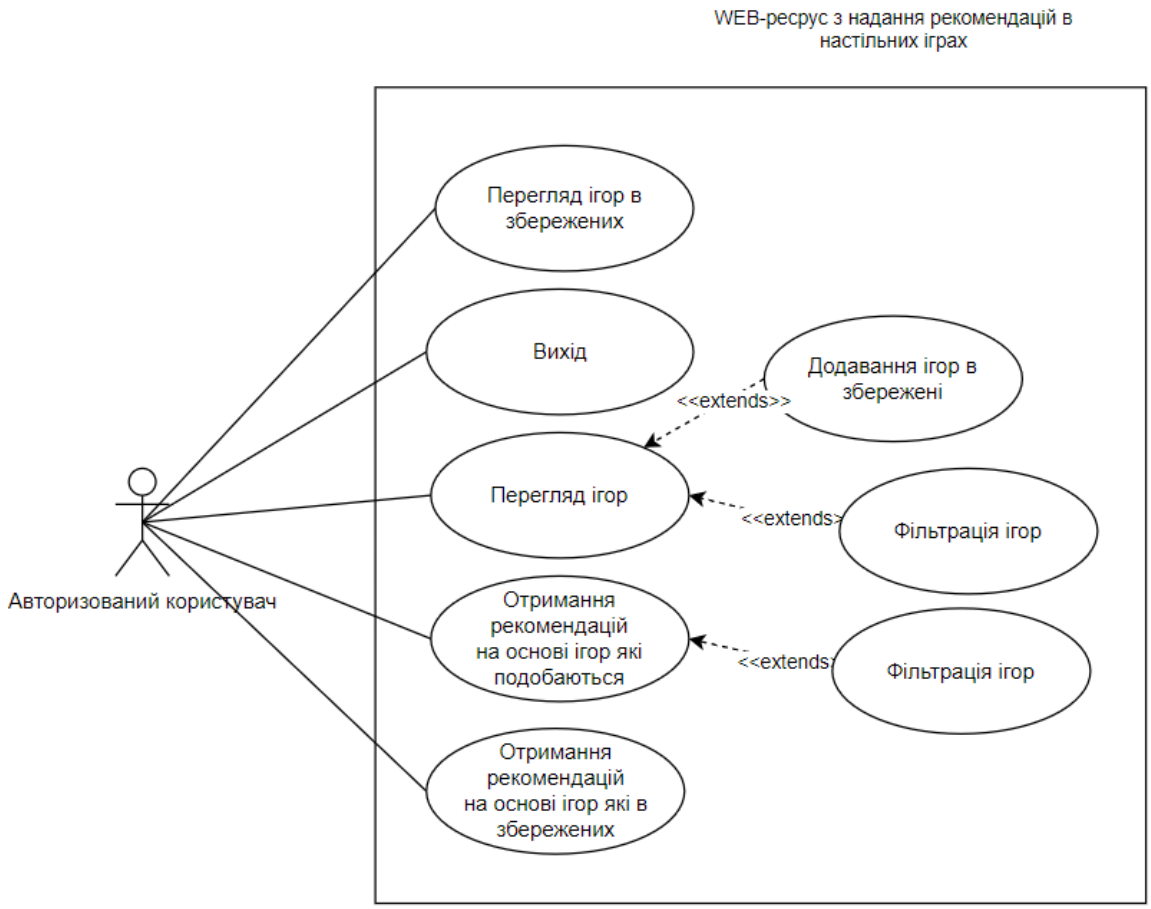



Рисунок 2.8 – Діаграма варіантів використання системи Авторизований користувач

Також буде розглянуто і розроблено діаграму класів для зображення структури мікросервісу з надання рекомендацій.

Діаграма класів описує типи класів у системі та статичні зв'язки між ними. Вона також показує атрибути та методи класів, а також обмеження, накладені на зв'язки між класами. Зовнішній вигляд і інтерпретація діаграми класів залежать від точки зору та рівня абстракції: на етапі аналізу класи можуть відображати об'єкти предметної області, тоді як на етапах проектування та реалізації вони можуть представляти компоненти програмної системи [15].

Розроблена діаграма класів зорбажена на рисунку 2.9.

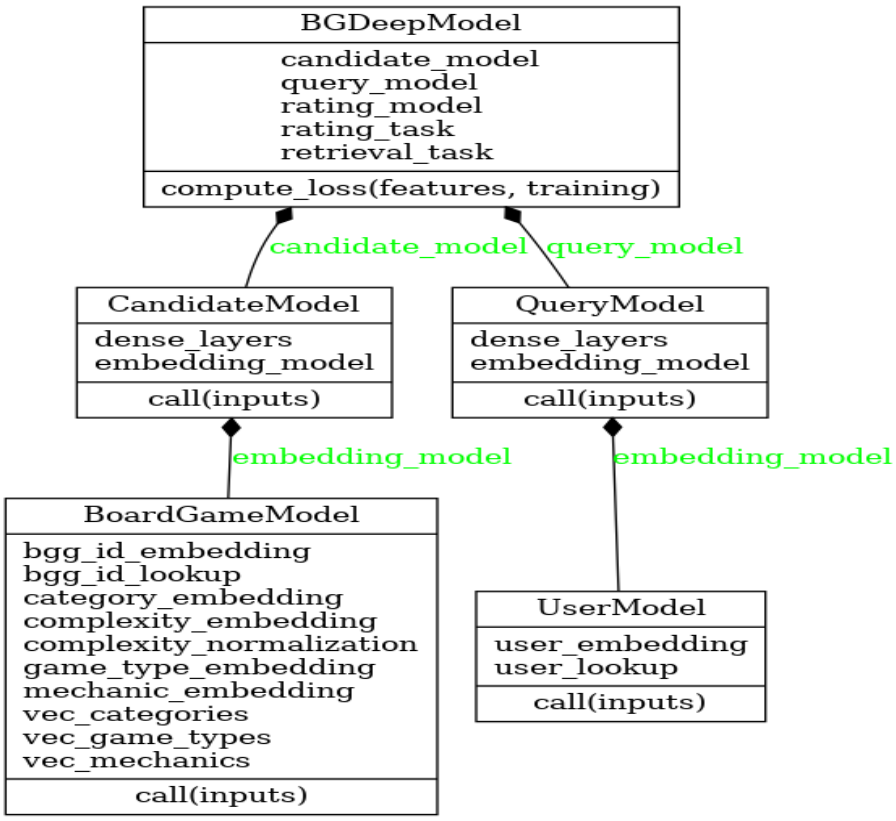



Рисунок 2.9 – Діаграма класів

Опишемо наведені класи

- це головний клас, який представляє глибоку нейронну мережу для рекомендацій ігор. Він містить наступні основні компоненти:

CandidateModel - це клас, який представляє модель для кандидатів (ігор). Він містить embedding_model (BoardGameModel) для вбудованого представлення ознак ігор та dense_layers для об'єднання цих вбудованих представлень.

QueryModel - це клас, який представляє модель для запиту користувача. Він містить embedding_model (UserModel) для вбудованого представлення імені користувача та dense_layers для обробки цього вбудованого представлення.

BoardGameModel - це клас, який представляє вбудовані шари для різних ознак ігор, таких як ідентифікатор гри, складність, типи ігор, категорії та механіки.

UserModel - це клас, який представляє вбудований шар для імені користувача.

Ці класи взаємодіють між собою для створення глибокої нейронної мережі, для рекомендацій ігор на основі вподобань користувачів та ознак ігор.

## 2.4 Розробка схеми алгоритму роботи WEB-ресурсу для надання рекомендацій в настільних іграх

Алгоритм - це структурована послідовність кроків, що за обмежену кількість дій призводить до розв'язання завдання.

Блок-схема – це графічне зображення процесу або алгоритму, що складається з набору стандартних символів для позначення різних етапів, рішень і дій, пов'язаних із цим процесом.

Опишемо алгоритм роботи WEB-ресурсу для надання рекомендацій в настільних іграх та зобразимо блок-схему на рисунку 2.10.

**І випадок**

Алгоритм у випадку натиснення кнопки «Реєстрація» складається з таких кроків:

Крок 1. Завантаження головної сторінки.

Крок 2. Користувач вибирає на яку сторінку перейти далі.

Крок 3. Користувач переходить на сторінку Авторизації.

Крок 4. Користувач натискає кнопку «Реєстрація».

Крок 5. Завантаження сторінки реєстрації .

Крок 6. Введення вхідних даних.

Крок 7. Якщо данні вірні перейти до кроку 6, якщо ні до кроку 3.

Крок 8. Відкриття сторінки Вхід.

**ІІ випадок**

Алгоритм у випадку натиснення кнопки «Вхід» складається з таких кроків:

Крок 1. Завантаження головної сторінки.

Крок 2. Користувач вибирає на яку сторінку перейти далі.

Крок 3. Користувач переходить на сторінку авторизації.

Крок 4. Користувач натискає кнопку «Вхід».

Крок 5. Завантаження сторінки входу.

Крок 6. Введення вхідних даних.

Крок 7. Якщо данні вірні перейти до кроку 8, якщо ні до кроку 5.

Крок 8. Відкриття сторінки профілю.

**III випадок**

Алгоритм у випадку, якщо вибрано сторінку «Рекомендації» складається з таких кроків:

Крок 1. Завантаження головної сторінки.

Крок 2. Користувач вибирає на яку сторінку перейти далі.

Крок 3. Користувач переходить на сторінку Рекомендацій.

Крок 4. Користувач вибирає ігри та додає їх в список ігор, які йому сподобались.

Крок 5. Користувач вибирає чи хоче він фільтрувати ігри, якщо так то крок 6 якщо ні то крок 7.

Крок 6. Користувач переглядає відфільтровані ігри.

Крок 7. Перевірка чи авторизувався користувач, якщо так то крок 8, якщо ні крок 9.

Крок 8. Користувач може добавити ігри в збережені.

Крок 9. Користувач вирішує чи бажає він перейти до іншої сторінки, якщо так то крок 2 якщо ні то крок 10.

Крок 10. Користувач вирішує чи бажає він завершити роботу з WEB-ресурсом, якщо так то крок 11 якщо ні то крок 3

Крок 11. Завершення роботи з WEB-ресурсом.

**IV випадок**

Алгоритм у випадку, якщо вибрано сторінку «Рекомендації» складається з таких кроків:

Крок 1. Завантаження головної сторінки.

Крок 2. Користувач вибирає на яку сторінку перейти далі.

Крок 3. Користувач переходить на сторінку Настільні ігри.

Крок 4. Користувач переглядає список ігор.

Крок 5. Користувач вибирає чи хоче він фільтрувати ігри, якщо так то крок 6, якщо ні то крок 7.

Крок 6. Користувач переглядає відфільтровані ігри.

Крок 7. Перевірка чи авторизувався користувач, якщо так то крок 8, якщо ні крок 9.

Крок 8. Користувач може добавити ігри в збережені.

Крок 9. Користувач вирішує чи бажає він перейти до іншої сторінки, якщо так то крок 2, якщо ні то крок 10.

Крок 10. Користувач вирішує чи бажає він завершити роботу з WEB-ресурсом, якщо так то крок 11, якщо ні то крок 3

Крок 11. Завершення роботи з WEB-ресурсом.

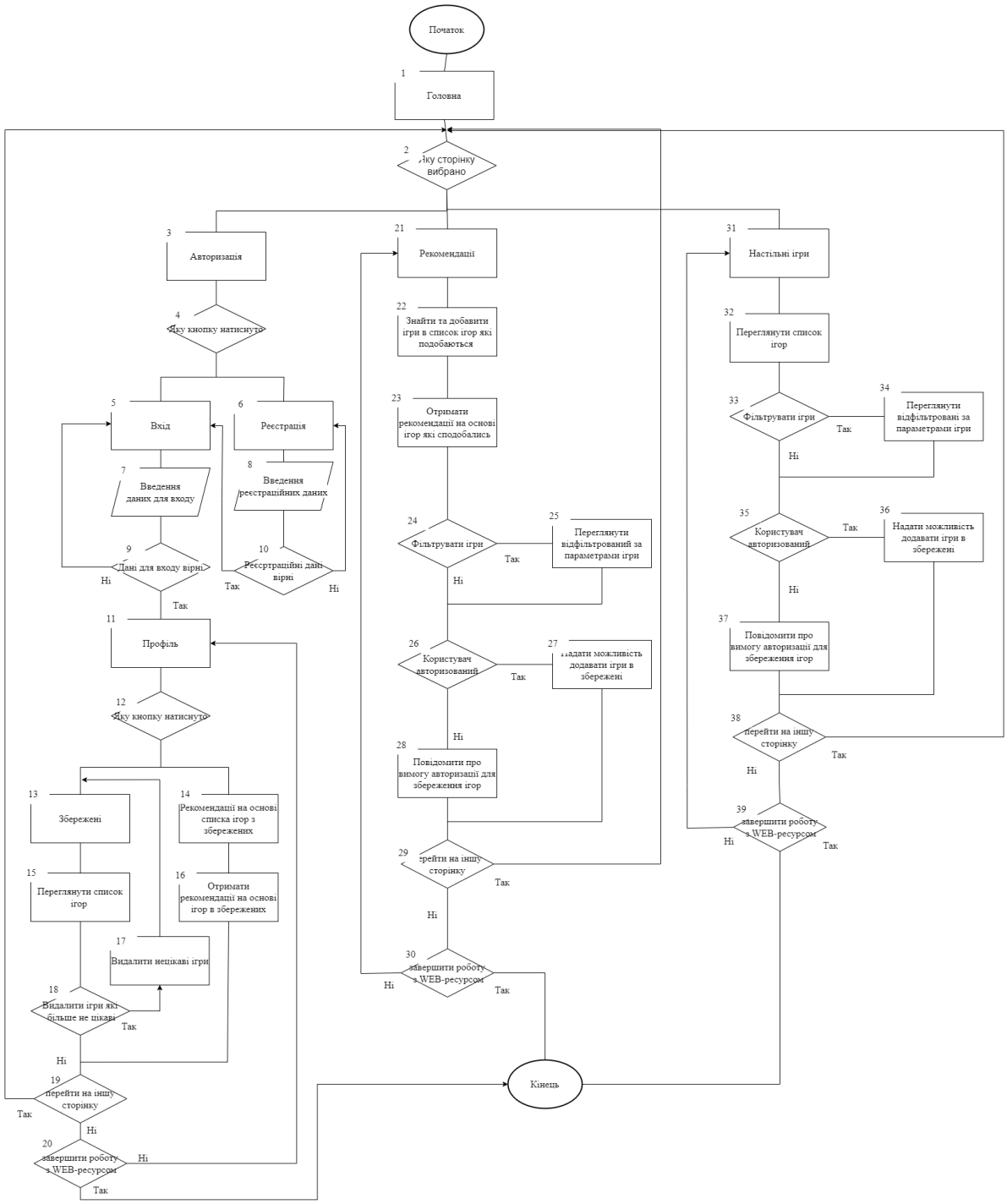



Рисунок 2.10 – Схема алгоритму роботи WEB-ресурсу для надання рекомендацій в настільних іграх

**V випадок**

Алгоритм у випадку натиснення кнопки «Збережені» складається з таких кроків:

Крок 1. Завантаження головної сторінки.

Крок 2. Користувач вибирає на яку сторінку перейти далі.

Крок 3. Користувач переходить на сторінку авторизації.

Крок 4. Користувач натискає кнопку «Вхід».

Крок 5. Завантаження сторінки входу.

Крок 6. Введення вхідних даних.

Крок 7. Якщо дані вірні перейти до кроку 8, якщо ні до кроку 5.

Крок 8. Відкриття сторінки профілю.

Крок 9. Натиснення кнопки збережені.

Крок 10. Перегляд списку ігор в закладках.

Крок 11 Чи хоче користувач видалити деякі ігри, якщо так то крок 12, якщо ні то крок 13.

Крок 12. Видалити ігри.

Крок 13. Перейти до кроку 8.

Крок 14. Користувач вирішує чи бажає він перейти до іншої сторінки якщо так то крок 2, якщо ні то крок 15.

Крок 15. Користувач вирішує чи бажає він завершити роботу з WEB-ресурсом, якщо так то крок 16, якщо ні то крок 8.

Крок 16. Завершення роботи з WEB-ресурсом.

**VI випадок**

Алгоритм у випадку натиснення кнопки «рекомендації на основі закладок» складається з таких кроків:

Крок 1. Завантаження головної сторінки.

Крок 2. Користувач вибирає на яку сторінку перейти далі.

Крок 3. Користувач переходить на сторінку авторизації.

Крок 4. Користувач натискає кнопку «Вхід».

Крок 5.Завантаження сторінки входу.

Крок 6. Введення вхідних даних.

Крок 7. Якщо дані вірні, перейти до кроку 8, якщо ні до кроку 5.

Крок 8. Відкриття сторінки профілю.

Крок 9. Натиснення рекомендації на основі закладок.

Крок 10. Переглянути список рекомендацій.

Крок 11. Користувач вирішує чи бажає він перейти до іншої сторінки, якщо так то крок 2, якщо ні то крок 12.

Крок 12. Користувач вирішує чи бажає він завершити роботу з WEB-ресурсом, якщо так то крок 13, якщо ні то крок 8

Крок 13. Завершення роботи з WEB-ресурсом.

## 2.5 Висновок

У цьому розділі були проведені дослідження та проектування з метою створення новітнього та функціонального WEB-ресурсу, що надаватиме рекомендації в настільних іграх для користувачів.

Було обґрунтовано вибір моделі Two Tower для надання рекомендацій у настільних іграх

Далі, було обґрунтовано вибір мікросервісної архітектури для WEB-ресурсу. Ця архітектура визначається своєю масштабованістю, можливістю використання різних технологій.

Під час проектування було розроблено UML-діаграми варіантів використання , що показують як користувачі взаємодіють з WEB-ресурсом та діаграму класів, щоб показати взаємодію та роботу рекомендаційної системи. Ці діаграми дозволяють краще зрозуміти функціональні можливості ресурсу та основні сценарії його використання.

Крім того, створена схема алгоритму функціонування WEB-ресурсу. Схема алгоритму надає детальний опис процесу роботи програми. Це допомагає кращому розумінню поставленої задачі та уникненню помилок під час реалізації функціональності.

Все це допомогає кращому розумінню завдання та зменшує шанс на помилки під час втілення функціоналу.

# ЕАЛІЗАЦІЯ WEB-РЕСУРСУ ДЛЯ НАДАННЯ РЕКОМЕНДАЦІЙ В НА-СТІЛЬНИХ ІГРАХ

## Обґрунтування вибору мов програмування

У даному розділі порівняємо переваги та недоліки 5 мов програмування: Python, C++, Java, C# та JavaScript.

- це високорівнева мова програмування загального призначення, яка широко використовується в таких галузях, Data Science, штучний інтелект та веб-розробка. Однією з головних переваг Python є її простота та зрозумілість. Вона також має велику та активну спільноту, а це означає, що розробники можуть користуватися великою кількістю ресурсів та бібліотек. Однак Python може працювати повільніше, ніж деякі інші мови, що робить її менш ідеальною для певних типів додатків, які вимагають великої обчислювальної потужності [16].

Переваги використання Python:

инамічна типізація

ython не залежить від платформи, і один і той самий код без проблем відображається на різних ОС (Windows, Linux, Mac).

она має широкий спектр бібліотек, що містять безліч модулів, пакетів і шаблонів.

ає велику підтримку спільноти.

ає багато фреймворків та бібліотек для нейронних мереж та машинного навчання.

Мінуси використання Python:

ython працює повільніше порівняно з C або C++.

ython погано підходить для розробки мобільних додатків.

ін споживає багато пам'яті.

івень доступу до баз даних у Python трохи недорозвинений і примітивний.

C++ - це високопродуктивна мова зі статичною типізацією, яка широко використовується в системному програмуванні, розробці ігор та багатьох інших сферах. Її перевага полягає в тому, що вона надзвичайно швидка та ефективна, що робить її гарним вибором для завдань, які вимагають великої обчислювальної потужності. Вона також вимагає більшої уваги до управління пам'яттю і може бути схильна до вразливостей безпеки при неправильному використанні [16].

Переваги C++:

можна використовувати для створення кодів системного рівня, а також кодів для вбудованих пристроїв.

ожливо легко додавати бібліотеки C++.

ля мережевих, ігрових та серверних додатків C++ є найкращим вибором.

Мінуси C/C++:

омилки не виявляються на початковому етапі. Вони з'являються лише після компіляції коду. Тому налагодження стає складним і виснажливим.

правління кодом є складним.

Java - це популярна об'єктно-орієнтована мова програмування, яка широко використовується для розробки Android-додатків та в багатьох інших галузях. Однією з головних переваг Java є її портативність, оскільки вона може працювати на будь-якому пристрої з встановленою віртуальною машиною Java. Вона також має велику та активну спільноту, що забезпечує широкий спектр доступних бібліотек та ресурсів. Однак, Java може бути повільнішою порівняно з деякими іншими мовами програмування і вимагає більше пам'яті для роботи. Крім того, вона може бути більш багатослівною, тобто потребувати більше рядків коду для виконання тих самих завдань, що й інші мови. [16].

Переваги використання Java:

ava є об'єктно-орієнтованою і дозволяє створювати багаторазові коди.

од на Java не залежить від стадії виконання і працює на будь-якому обчислювальному пристрої з попередньо встановленою JVM.

сновна перевага Java полягає в тому, що вона дуже безпечна.

ava розподіляє пам'ять на дві секції: оперативну пам'ять і стекову пам'ять. Отже,

щоразу, коли викликаючи змінну, JVM викликає її з кеш-пам'яті або зі стеку. Таким чином, зберігати та відновлювати інформацію в Java дуже просто.

отенціал багатопотоковості Java допомагає виконувати кілька завдань одночасно.

Мінуси використання Java:

ava працює повільно і вимагає великого обсягу пам'яті.

ava має багатослівні коди з довгими складними реченнями.

ерування пам'яттю за допомогою Java є дещо складним.

C# - це сучасна об'єктно-орієнтована мова, яка широко використовується для розробки додатків та ігор для Windows. Вона має багато можливостей, які роблять її простою у використанні та підтримці, наприклад, автоматичне керування пам'яттю та сильна типізація. C# також має велику та активну спільноту, з великою кількістю ресурсів та бібліотек. Однак C# переважно використовується на платформі Windows, тому вона може бути не такою портативною, як деякі інші мови. [16].

Переваги C#:

видка розробка додатків з чіткою структурою програми.

омпіляція та виконання коду відбувається швидко за допомогою C#.

# проста у вивченні та підтримує сучасні функціональні можливості.

порівнянні з процедурно-орієнтованими мовами, C# спрощує розробку коду та управління ним.

# підтримує автоматичне збирання сміття.

Недоліки C#:

# бракує гнучкості. Вам потрібен попередньо встановлений фреймворк .Net, оскільки C# повністю залежить від Microsoft.Net.

JavaScript - це популярна мова високого рівня, яка широко використовується у веб-розробці. Однією з головних переваг JavaScript є її розповсюдженість, оскільки вона підтримується всіма сучасними веб-браузерами. Це також динамічно типізована мова, що означає, що вона більш гнучка і легша у вивченні, ніж деякі інші мови. Однак JavaScript може працювати повільніше, ніж деякі інші мови, і може бути не найкра-

щим вибором для певних типів додатків, які потребують великої обчислювальної потужності. Вона також може бути схильною до вразливостей безпеки при неправильному використанні [16].

Переваги використання JavaScript:

S працює порівняно швидко і плавно.

а допомогою JS можна створювати багаті інтерфейси.

S  швидко рендериться в браузері на стороні клієнта.

и можете створити повністю функціонуючий додаток від початку до кінця, використовуючи лише JS.

Мінуси використання JS:

дна помилка в JS-коді може призвести до зупинки рендерингу всього веб-сайту.

ажко створювати великі додатки на JS.

ізні браузери інтерпретують JS по-різному. Таким чином, стає важко читати і писати кросбраузерний код на JS.

Щоб подолати недоліки JS, можна використовувати TypeScript.  Це надбудова , яка додає до мови статичну типізацію. Він призначений для створення великих додатків, які трансполюються на JS.

Для розробки користувацької і серверної частини було вирішено використовувати JavaScript. Ця мова програмування є невід'ємною частиною веб-розробки і підтримується усіма сучасними браузерами. Завдяки величезній спільноті інженерів, існує багата екосистема бібліотек та фреймворків для JavaScript, що полегшує створення як користувацьких інтерфейсів, так і серверної логіки. Крім того, динамічна типізація робить JavaScript гнучкою. А можливість писати повнофункціональні додатки лише на одній мові від фронтенду до бекенду може спростити розробку всього проекту.

Для реалізації мікросервісу з надання рекомендацій було вирішено взяти Python. Ця мова широко застосовується в, машинному навчанні та обробці даних завдяки потужним бібліотекам та фреймворкам. Оскільки система рекомендацій базується на алгоритмах машинного навчання, Python став логічним вибором для цього мікросервісу.

**Обґрунтування вибору бібліотек та фреймворків для розробки WEB-ресурсу для надання рекомендацій в настільних іграх**

В попередньому розділі було вирішено, що для розробки серверної та клієнтської частини буде використано мову програмування JavaScript, а для мікросервісу з надання рекомендацій Python, в даному розділі ми розглянемо фреймворки і бібліотеки доступні на даних мовах для розробки нашого WEB-ресурсу.

Фреймворки є важливим інструментом для фронтенд-розробників, оскільки вони надають структурований, організований та узгоджений код для розробки веб-додатків від планування до розгортання. Всі вони являють собою заздалегідь написаний код, до якого фронтенд-розробники можуть легко отримати доступ і без проблем вставити його у власні проєкти. Деякі загальні риси включають:

Компонентний дизайн: фреймворки надають розробникам різноманітні можливості, а завдяки модульним компонентам, розробники можуть створювати свої сторінки з нуля. Компоненти, які надають фреймворки, - це багаторазові пакети коду з різними функціями.

Послідовний крос-платформний код: у фреймворку код є послідовним і стандартизованим, що дає веб-розробнику фронтенду перевагу. Це значно полегшує роботу на різних платформах і між командами, оскільки вони можуть зосередитися на інших аспектах, окрім узгодженості.

Маніпуляції з DOM: DOM, або модель об'єктів даних, необхідна розробникам для використання у своїх проєктах, і фреймворки іноді мають інструмент маніпулювання DOM, який дозволяє оптимізувати процес рендерингу та підвищити загальну швидкість реакції користувацького інтерфейсу [17].

Це лише деякі з основних функцій, які можна знайти у фронтенд-фреймворках; однак, існує ще більше доступних функцій, які роблять кожен фреймворк унікальним.

Давайте розглянемо плюси і мінуси двох різних популярних фреймворків React та Angular.

– Розроблений і підтримуваний компанією Meta, React - це фреймворк написаний на мові програмування Javascript з відкритим вихідним кодом, який пропонує фронтенд-

розробникам безліч варіацій компонентів, що будують користувацький інтерфейс, і є одним з найбільш широко використовуваних фреймворків для веб-розробників [17].

Переваги React

рхітектура на основі компонентів: React може похвалитися багаторазовими компонентами, які не впливають на інші частини дизайну та користувацького інтерфейсу, і завдяки цьому веб-розробники можуть легко керувати та маніпулювати своїми кодами та дизайном.

іртуальний DOM: Хоча деякі фреймворки безпосередньо маніпулюють DOM, React має функцію, яка дозволяє веб-розробникам рендерити свої DOM більш ефективно.

Мінуси React

остійні зміни: Оскільки код на React постійно змінюється і розвивається, фронтенд веб-розробникам також потрібно постійно бути в курсі подій і використовувати найсучасніший та найцікавіший код.

– Ще один популярний фреймворк з великою кількістю функцій і підтримується компанією Google. Хоча це фреймворк на Javascript, як і React, він написаний на TypeScript і в основному використовується для створення односторінкових додатків

Переваги Angular

агатофункціональність: Angular випереджає інші фронтенд-фреймворки своїми можливостями. Він пропонує комплексну архітектуру MVC, двостороннє зв'язування даних, використання TypeScript. Розробники Angular мають у своєму розпорядженні всі необхідні інструменти в рамках одного фреймворку.

еревага TypeScript: TypeScript - це мова програмування, заснована на JavaScript, яка покращує та розширює її можливості. Оскільки TypeScript є синтаксичною надмножиною JavaScript, він дозволяє використовувати статичну типізацію, підвищуючи якість коду та полегшуючи виявлення помилок для розробників.

Мінуси Angular:

кладність: Оскільки цей фреймворк багатослівний і має багато функцій та інструментів, Angular більше підходить для складних додатків, ніж для простих.

Переглянувши всі мінуси та плюси фреймворків, було вибрано саме React. Він є найпопулярнішим фреймворком для розробки WEB-ресурсів з чого випливає що він має більшу підтримку спільноти, бібліотек і надбудов. На відмінну від Angular його можна використовувати як для розробки великий проектів, так і для маленьких. Він використовує саме JS, а не TS.

Далі розглянемо бібліотеки, які знадобляться для розробки WEB-ресурсу крім фреймворка.

Для навігації буде викорстано react-router-dom. React Router DOM — це пакет npm, який дозволяє реалізувати динамічну маршрутизацію у WEB-ресурсі.

React Router Dom використовується для створення односторінкових програм, тобто програм, які мають багато сторінок або компонентів, але сторінка ніколи не оновлюється, а вміст динамічно вибирається на основі URL-адреси. Цей процес називається маршрутизацією, і він можливий за допомогою React Router Dom. Головна перевага react-router полягає в тому, що сторінку не потрібно оновлювати, наприклад, коли клацнуто посилання на іншу сторінку. Крім того, це швидко, дуже швидко порівняно з традиційною навігацією по сторінках. Це означає, що користувальницький досвід є кращим, а додаток має загальну кращу продуктивність [18].

Для стилізації додатку буде використано Material UI – це бібліотека React-компонентів з відкритим кодом, яка реалізує принципи Material Design від Google. Вона є повноцінним рішенням і може бути використана у проектах без додаткових налаштувань. [19].

Завдяки Material-UI WEB-ресурс матиме гарний зовнішній вигляд, зручний інтерфейс та відповідатиме сучасним стандартам веб-розробки.

На цьому вибір інструментів для створення клієнтської частини завершено, перейдемо до вибору інструментів для серверної частини.

Для написання сервеної частини буде використано середовище Node.js, одним з найпопулярніших фреймворків для розробки додатків на Node.js є Express, який є частиною сімейства Node.js.

Переваги Express:

овний спектр сторонніх пакетів, доступних в Express, готових до використання, є, мабуть, його найбільшою перевагою.

нучкість у написанні кастомної логіки, пристосованої до ваших конкретних потреб.

xpress майже такий самий старий, як і Node.js; отже, він вдосконалювався протягом багатьох років. Це призвело до його відносної стабільності у порівнянні з його альтернативами.

xpress має велику спільноту, що означає, що багато проблем було вирішено, а нові проблеми можуть бути швидко вирішені.

Мінуси Express:

ножинні вкладені зворотні виклики, відомі як «пекло зворотних викликів», є одним з недоліків Express.js.

Nest.js є одним з найбільш швидкозростаючих фреймворків Node.js завдяки своїм неймовірним можливостям. Це фреймворк з відкритим вихідним кодом, який дозволяє інженерам створювати складні системи.

Плюси:

est.js прогресивний! Він використовує сучасні функції JavaScript для реалізації, кращі шаблони дизайну для розробки веб-додатків.

ей фреймворк є розширюваним і гнучким, що дозволяє використовувати інші бібліотеки JavaScript.

ін універсальний. Він має адаптивну екосистему до всіх типів і розмірів серверних додатків [20].

Недоліки:

чимось схожий на фронтенд фреймворк Angular, також використовує TypeScript, і більше налаштований для розробки великих проєктів.

З перелічених фреймворків для нашого завдання найкраще перейде популярний і стабільний Express. Nest створений більше для складних систем і він достатньо новий, що може призвести до того, що при розробці будуть виникати якісь проблеми які ще не вирішено розробниками фреймворку.

Далі розглянемо бібліотеки, які знадобляться для розробки серверної частини WEB-ресурсу крім фреймворка.

Для звязку з базою даних буде використано Sequelize — це інструмент Node.js ORM на основі обіцянок для Postgres, MySQL, MariaDB, SQLite, Microsoft SQL Server, . Він має надійну підтримку транзакцій, відносини, активне та відкладене завантаження, реплікацію читання тощо [21].

ORM, або Object Relational Mapper, — це частина програмного забезпечення, призначена для перекладу між представленнями даних, які використовуються базами даних, і тими, які використовуються в об’єктно-орієнтованому програмуванні.

Для створення мікросервісу з рекомендаційною системою буде використана бібліотека Tensorflow. Tensorflow - це open-source бібліотека з відкритим вихідним кодом, яка розповсюджується безкоштовно. Вона призначена для задач машинного навчання та штучного інтелекту [22]. Хоча Tensorflow можна застосовувати для різних цілей, вона особливо потужна для навчання та роботи з глибокими нейронними мережами. Саме тому ця бібліотека була обрана для реалізації рекомендаційної системи в рамках цього проєкту.

Також у розробці буде задіяна бібліотека FastAPI, оскільки вона добре підходить для невеликих проектів завдяки своїй простоті, високій продуктивності та зручності використання. FastAPI дозволяє швидко створювати API з мінімальним обсягом коду. Крім того, ця бібліотека підтримує асинхронне програмування, що покращує швидкодію додатків. Вона також легко інтегрується з іншими популярними бібліотеками та фреймворками на Python.

**Обґрунтування вибору середовища програмування**

В даному розділі буде проведений порівняльний аналіз трьох різних середовищ розробки: PyCharm, Webstorm, Visual Studio Code. Кожне з цих середовищ має свої переваги та недоліки, які будуть враховані при виборі найбільш зручного середовища розробки для даного WEB-ресурсу.

Visul studio code – Це IDE від Microsoft і використовується для розробки веб-сайтів, веб-додатків і мобільних додатків. Воно має широкий спектр функцій, що дозволяє розробляти майже все. Його можна використовувати як у Windows, так і в MAC

Плюси IDE:

езкоштовне і має відкритий вихідний код, що робить його легко доступним для будь-кого.

егке і швидке.

ає настроюваний інтерфейс, що дозволяє користувачам адаптувати його до своїх конкретних потреб.

істить вбудовану підтримку для налагодження, яка може бути дуже корисною для виявлення та виправлення помилок у коді.

Мінуси IDE:

еякі користувачі можуть вважати інтерфейс перевантаженим або захаращеним

ля деяких додаткових функцій може знадобитися встановлення розширень або плагінів.

е включає стільки функцій із коробки, скільки деякі інші IDE, такі як PyCharm або WebStorm.

уже мало вбудованих інструментів налагодження.

PyCharm — це IDE, розроблений JetBrains спеціально для Python. Він найбільш відомий завдяки інтелектуальному завершенню коду, перевірці коду, підсвічуванню помилок на льоту та дуже швидким виправленням. Він також пропонує чудову підтримку для використання фреймворків, таких як Django, Flask, тощо. Він інтегрується з IPython Notebook, і також можна використовувати такі наукові пакети, як Numpy та Matplotlib. Ця IDE підтримує ОС Windows, macOS і Linux [23].

Плюси IDE:

пеціально розроблений для розробки Python.

ключає такі функції, як налагодження, рефакторинг коду та підтримку систем контролю версій.

ає чистий та інтуїтивно зрозумілий інтерфейс.

оступно безкоштовно з обмеженими ресурсами.

Мінуси IDE:

рофесійний редактор є платним продуктом.

удово підходить для розробки Python, але може бути не таким ефективним для інших мов.

оже бути надмірним для деяких користувачів, яким потрібен лише базовий редактор коду.

еякі користувачі можуть віддати перевагу полегшеному редактору коду, наприклад Visual Studio Code.

JetBrains WebStorm є середовищем розробки програмного забезпечення, створеним компанією JetBrains. Воно призначене для роботи з JavaScript, HTML та CSS. WebStorm базується на платформі IntelliJ IDEA і є спеціалізованою версією іншого продукту від JetBrains – PhpStorm. WebStorm пропонує підмножину функціональних можливостей PhpStorm. WebStorm постачається з вбудованими плагінами, які розширюють можливості роботи з JavaScript, такими як підтримка Node.js. Ці плагіни також доступні безкоштовно для користувачів PhpStorm. [24].

Плюси IDE:

пеціально розроблений для веб-розробки з акцентом на JavaScript.

ає чистий та інтуїтивно зрозумілий інтерфейс.

е IDE переважно для проектів, пов’язаних із веб-розробкою.

егулярно оновлюється новими функціями та вдосконаленнями.

Мінуси IDE:

рофесійне видання є платним продуктом.

уже багато можливостей, не підходить для новачків Деякі користувачі можуть віддати перевагу полегшеному редактору коду, наприклад Visual Studio Code.

Отже, проаналізувавши всі плюси та мінуси доступних IDE було вирішено для розробки WEB-ресурсу для надання рекомендацій в настільних іграх використати Visual Studio Code. Хоча інші IDE, такі як PyCharm для Python або WebStorm для JavaScript, можуть бути більш спеціалізованими та мати додаткові функції з коробки,

Visual Studio Code є більш легким і гнучким рішенням, при установці додаткових розширень дозволить комфортно працювати з двома мовами одночасно. Також враховуючи те що нам потрібно одночасно підтримувати велику кількість технологій це є чудовим рішенням.

**Розробка компонентів WEB-ресурсу для надання рекомендацій в настільних іграх**

Для реалізації клієнтської частини, як уже було описано в попередніх розділах буде використано фреймворк React. Розробка інтерфейсу користувача складається з розробки різних функціональних компонентів. Функціональні компоненти повертають JSX код. В загальному наш інтерфейс буде складатись з таких компонентів <Auth n/> <Profile/>. Кожен з компонентів відповідає за відображення певної сторінки.

Коротко про кожен компонент:

Auth />: цей компонент відповідає за обробку загальних аспектів аутентифікації. Він може включати в себе логіку визначення, чи користувач вже аутентифікований, та перенаправлення на відповідні сторінки.

Login />: компонент для сторінки входу. Містить форму для введення логіну та паролю, а також обробники подій для відправки цих даних на сервер.

SignUp />: компонент для сторінки реєстрації. Містить форму для введення даних нового користувача та обробники подій для реєстрації на сервері.

Boardgames />: компонент для відображення списку настільних ігор. Виконує запити до сервера для отримання даних про ігри та їх відображення.

компонент для відображення рекомендацій щодо настільних ігор. Виконує запити до сервера для отримання персоналізованих рекомендацій.

Profile />: компонент для відображення профілю користувача.

За допомогою react-router-dom задається перенаправлення між всіма цими компонентави. Наведемо код роутера.

Для зв'язку даних між всіма цими функціональними компонентами використовуються стани. У React стани дозволяють зберігати та управляти динамічними даними.

Для обміну даними між клієнтською частиною та серверною частиною (бекендом) застосовується REST API (Репрезентативний Переносний Інтерфейс Стану). Для захисту даних і забезпечення безпеки використовується технологія JWT (JSON Web Token). JWT дозволяє здійснювати надійну автентифікацію (перевірку особи) та авторизацію (надання прав доступу) користувачів. Токен JWT зберігається на клієнтській стороні (наприклад, у веб-браузері) і передається разом з кожним запитом до захищених ресурсів на сервері, забезпечуючи безпечний доступ.

Бекенд який реалізовує Rest API буде написано за допомогою Node.js Express. Для взаємодії з базою даних, в нашому випадку PostgreSQL буде використано ORM equalize. Для цього на стороні бекенда будуть створюватись моделі для відображення таблиць з бази даних. Бекенд буде відповідати та реалізовувати всі потрібні ендпоїнти та логіку взаємодії з мікросервісом з отримання рекомендацій

Сам мікросервіс з надання рекомендацій буде використовувати Fast API для створення невеличкого REST API та Tensorflow для створення гібридної моделі для надання рекомендацій

В загальному модель для надання рекомендацій буде мати буде мати такий вигляд

Для урізноманітнення рекомендацій які надаються моделлю добавимо категоризація за допомогою LDA.

LDA (Latent Dirichlet Allocation) - це популярна тематична модель, яка використовується в обробці природної мови та аналізі текстових даних. Вона допомагає виявляти приховані тематики або теми в корпусі документів.

Основна ідея LDA полягає в тому, що кожен документ (в даному випадку опис настільної гри) являє собою суміш різних тем, і слова в документі генеруються цими прихованими темами. Модель LDA автоматично виявляє ці приховані теми та визначає, наскільки кожен документ відноситься до різних тем [25]. Для цього використаєм готову модель з бібліотеки Gensim.

Далі опишемо, як будуть зберігатись наші дані про користувачів в базі даних.

База даних невеличка і буде складатись всього з 2 таблиць, Гри та Користувача які будуть зв’язані між собою зв’язком Багато-Багато. Для реалізації зв’язку багато до багатьох використовують асоціативну табличку, зображено дану табличку і зв'язок на рисунку 3.1

Рисунок 3.1 – Асоціативна таблиця

Зобразимо дану структуру за допомогою ER моделі. Реалізація зв'язку «багато до багатьох» у програмуванні є складною задачею, оскільки реляційні бази даних в своїй основі підтримують лише зв'язки типу «один до багатьох». Щоб подолати цю проблему, розробники баз даних створюють додаткову проміжну сутність, яка слугує своєрідним з'єднувачем між двома основними сутностями, забезпечуючи можливість встановлення зв'язку «багато до багатьох» між ними. (рис. 3.2).

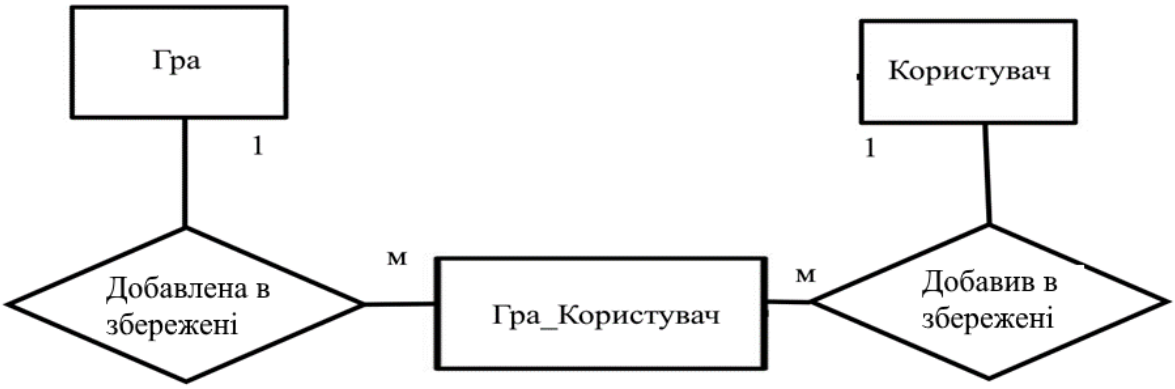



Рисунок 3.2 – ER модель бази даних

В даному випадку штучним типом сутності буде Гра_Користувач.

**Тестування розробленого WEB-ресурсу**

Для перевірки коректності роботи розробленого WEB-ресурсу буде проведено його тестування.

Для авторизації користувач переходить на сторінку авторизації, де в нього є вибір або зареєструватись або увійти, вигляд сторінки зображено на рисунку 3.3

Рисунок 3.3 – Сторінка авторизації

Якщо користувач натиснув кнопку Реєстрація, відкриється форма,  у якій, у відведених полях потрібно увести електронну адресу, ім’я, пароль та завантажити аватар. Форма реєстрації зображена на рисунку 3.4.

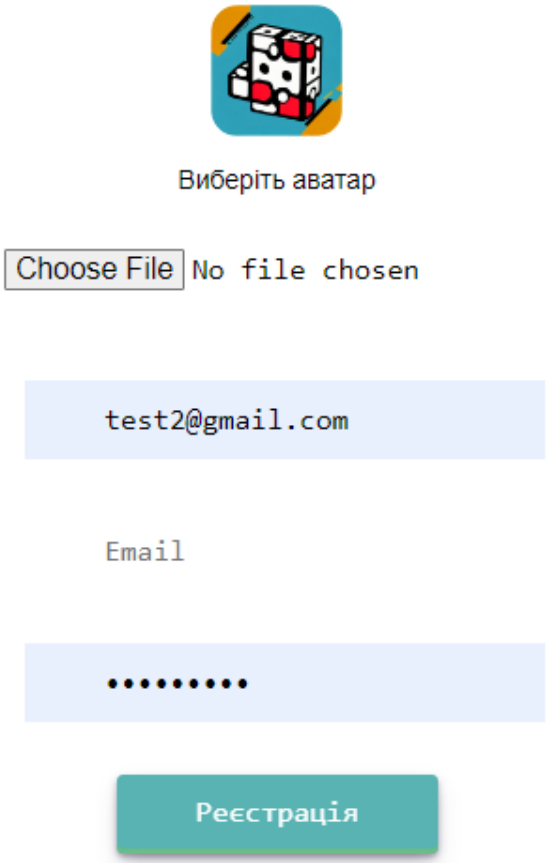



Рисунок 3.4 –  Форма реєстрації

Після реєстрації користувач може увійти до свого профілю за допомогою форми входу, використавши електронну пошту та пароль які він використав при реєстрації. Форма зображена на рисунку 3.5. В результаті входу на сервері генерується JWT токен який зберігається в кешові браузера і з кожним запитом він буде відправлятись для підтвердження авторизації. Токен має час дії, після закінчення цього часу, користувача автоматично викине з системи.

Рисунок 3.5 – Форма входу

Після авторизації відкривається сторінка з профілем користувача, можемо впевнитись, що на відкритій сторінці відображається вибраний користувачем аватар та його ім’я, а також кількість ігор (рис. 3.6).

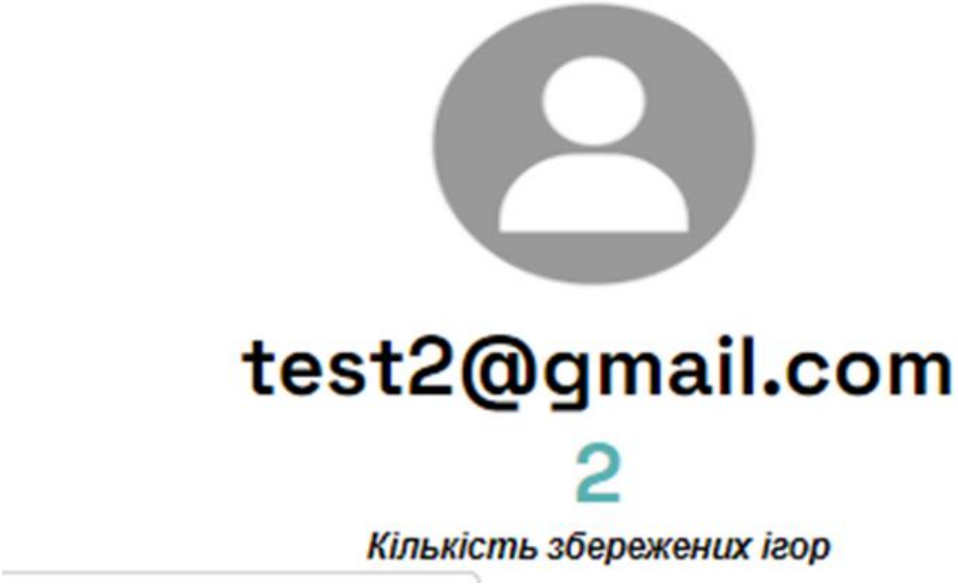



Рисунок 3.6 – Інформація про користувача в профілі

В самому профілі користувач може переглянути ігри які він зберіг (рис. 3.7).

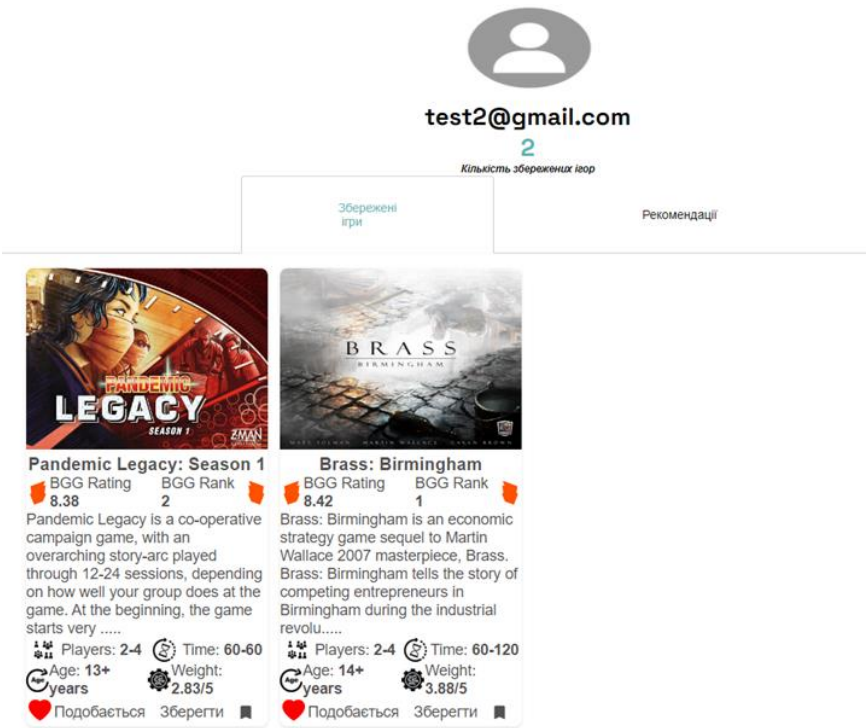



Рисунок 3.7 – Збережені ігри

Також в профілі користувач може отримати рекомендації на основі збережених ігор (рис. 3.8).

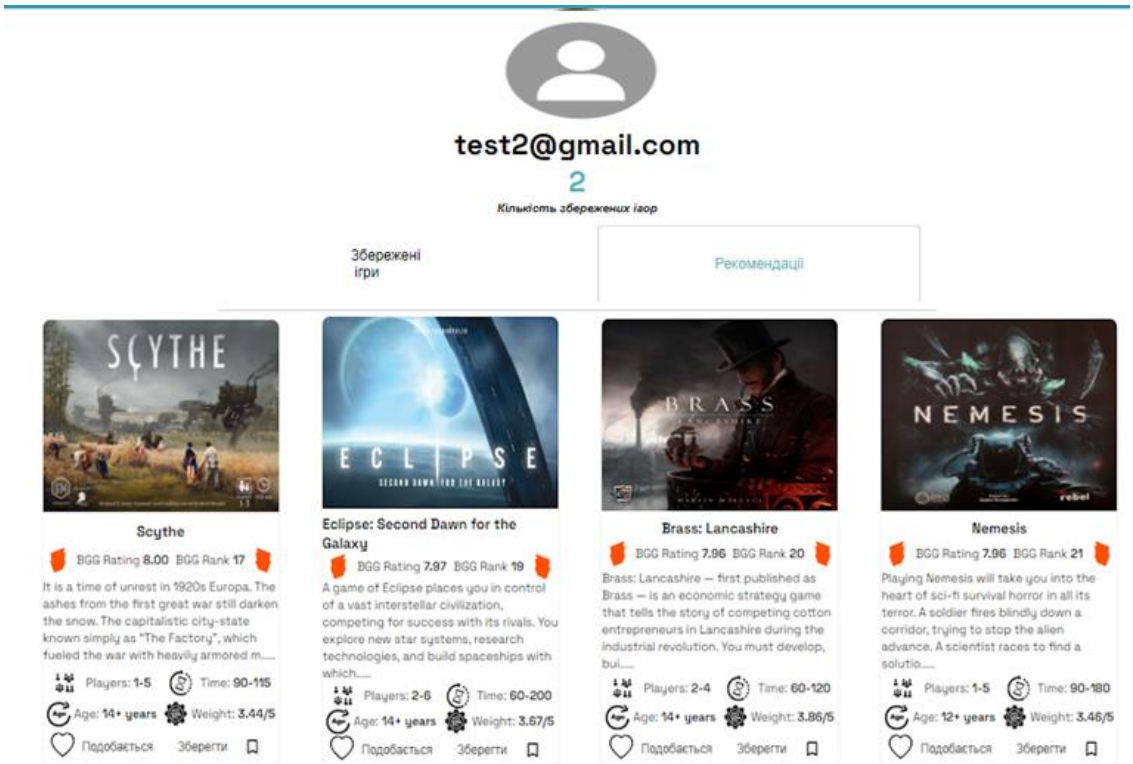



Рисунок 3.8 – Рекомендації на основі збережених ігор

Авторизований користувач може перейти до списку всіх ігор і побачити індикатори, які показують ігри, що він добавив в список, які йому подобаються, а які ігри добавив в список збережених. Він може добавляти і забирати ігри прямо в цьому списку (рис. 3.9).

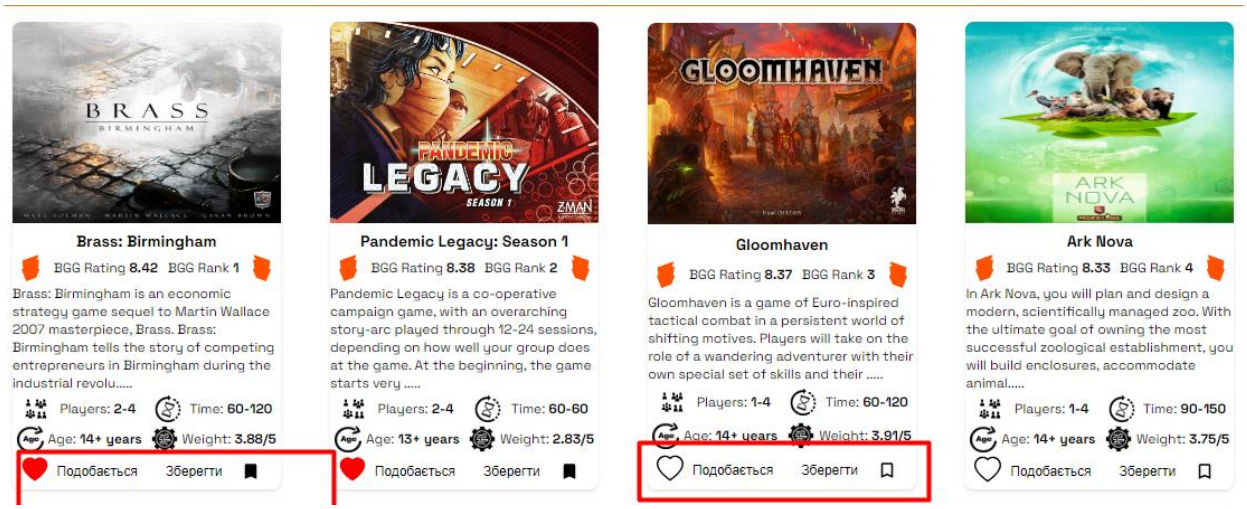



Рисунок 3.9 – Список ігор з кнопками подобається та зберегти

Також у користувачів, незалежно від того чи авторизувались вони є можливість добавити ігри в список ігор які їм сподобались і отримати на цій основі рекомендації, результат таких дій зображений на рисунку 3.10.

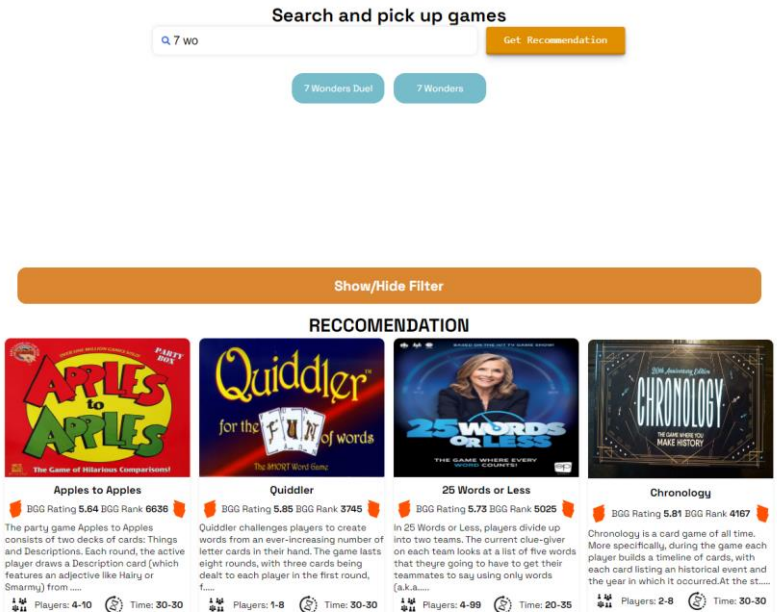



Рисунок 3.10 – сторінка рекомендацій

За бажанням користувач може відфільтрувати отримані результати, і результати всіх цих операцій зберігаються при перезавантаженні сторінки в кешові браузера. Фільтр зображено на рисунку 3.11.

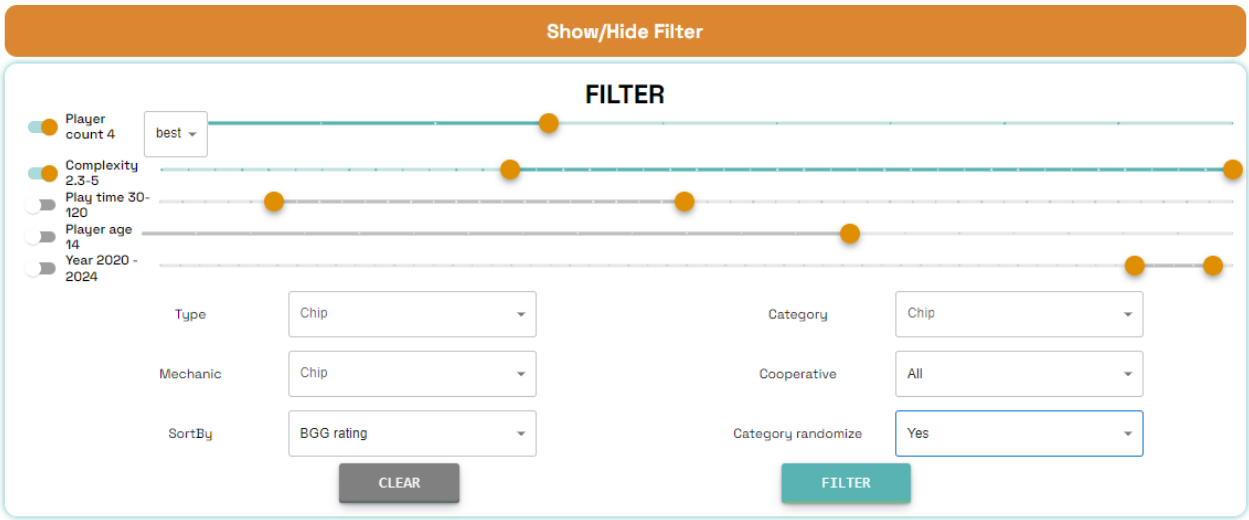



Рисунок 3.11 – Фільтр

В самому меню фільтрації для користувача доступні такі функції:

ожливість вмикати і вимикати потрібний параметр фільтрації (рис. 3.11).

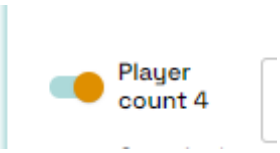



Рисунок 3.12 – Перемикач для параметру кількість гравців

два типи слайдерів для вибору значень, один який вибирає тільки одне значення, інший який вибирає діапазон значень (рис. 3.12).

Рисунок 3.13 – Слайдер для параметру кількість гравців

акож є списки з вибором багатьох варіантів для наприклад категорій (рис.

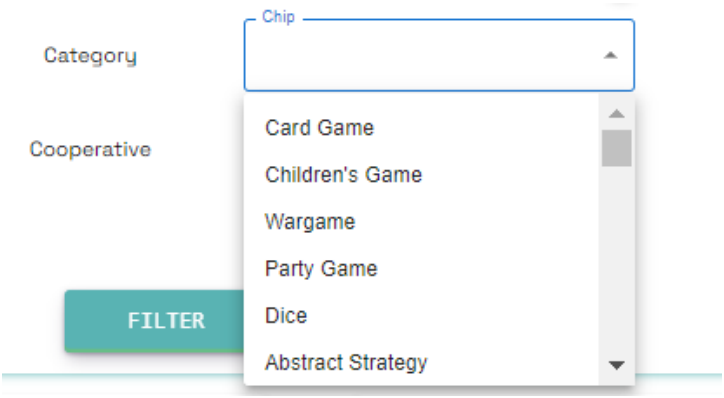



Рисунок 3.14 – Випадаючий список для вибору

випадаючий список для вибору певних характеристик, як от чи включати категорії згенеровані LDA ігри (рис. 3.15).

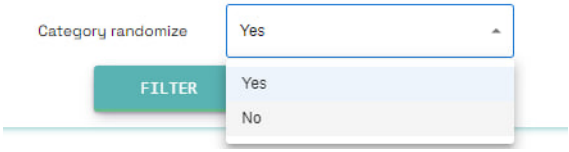



Рисунок 3.15 – Випадаючий список для вибору

Відповідно до проведених досліджень було проведено порівняння розробленого WEB-ресурсу з раніше розглянутими аналогами. В порівнянні з аналогами розроблений WEB-ресурс виводить більше інформації про ігри, дозволяє фільтрувати ігри з більшим спектром налаштувань фільтрації, дає можливість зберігати отримані рекомендації, не втрачає отриманий результат при перезавантаженні сторінки, використовує гібридні нейронні мережі для надання кращих рекомендацій, для отримання рекомендацій дозволяє вибирати в рази більше ігор.

Отже, проаналізувавши та протестувавши розроблений додаток, можна стверджувати, що поставлених цілей досягнуто.

### 3.6 Висновок

В цьому розділі проведено аналіз найчастіше використовуваних для розробки мов програмування. Визначено особливості, переваги та недоліки кожної з них, а також доцільність у використанні для поставленого завдання. В результаті аналізу для реалізації клієнтської і серверної частин було обрано мову програмування JavaScript, а для сервісу надання рекомендацій Python.

Після вибору мов було обґрунтовано вибір фреймворків та бібліотек. Обґрунтування включало ретельний аналіз переваг і недоліків кожного з варіантів. Для розробки клієнтської частини був обраний фреймворк React, для сервеної Express. Для мікросервісу з надання рекомендацій були вибрані бібліотеки FastApi та Tensorflow. В загальному додатково під час аналізу були вибрані такі бібліотеки: Sequalize, react-router-dom,  material ui. Для управління базою даних була вибрана безкоштовна система керування реляційною базою даних PostgreSql. Було розроблено ER – модель бази даних WEB-ресурсу для надання рекомендацій в настільних іграх.

Також обґрунтовано вибір IDE для розробки даного проєкту. Вибір пав на Visual Studio Code, як IDE яке зможе забезпечити гнучке управління всіма вище переліченими технологіями. Всі ці кроки є важливими для забезпечення успішного створення WEB-ресурсу.

Проведено тестування розробленого додатка та порівняння з аналогами.

Було розширено функціональні можливості WEB-ресурсу з надання рекомендацій в настільних іграх: зберігання рекомендацій, збільшення кількості вибору ігор, покращення рекомендаційної системи, зберігання даних при перезавантаженні, збільшенню можливості фільтрації. З цього можна зробити висновок, що мету дослідження було досягнуто.

# ВИСНОВКИ

У результаті виконання бакалаврської дипломної роботи було розроблено WEB-ресурс для надання рекомендацій в настільних іграх.

Всі завдання поставлені перед бакалаврською дипломною роботою виконані.

Було обґрунтовано доцільність розробки WEB-ресурсу для надання рекомендацій в настільних іграх та проаналізовано переваги та недоліки відомих програмних рішень, що дозволило сформулювати задачі для подальшого створення WEB-ресурсу.

Було проведено проектування, а саме обґрунтовано вибір архітектури для WEB-ресурсу. Визначено та описано основні функції системи та процес взаємодії користувачів з нею за допомогою UML діаграм. Обґрунтовано використання моделі Two Tower для надання рекомендацій. Розроблено та описано схему роботи самого WEB-ресурсу.

Також була здійснена реалізація. Для початку було обґрунтовано вибір інструментів, а саме: мов програмування, фреймворків, бібліотек та середовищ розробки. Проведено тестування розробленого WEB-ресурсу для перевірки його функціональності, коректності та взаємодії компонентів системи. Окрім того, було проведено порівняльну характеристику розробленого WEB-ресурсу з ресурсами-аналогами.

Мета дослідження у вигляді розширення функціональних можливостей WEB-ресурсу для надання рекомендацій в настільних іграх була досягнута за рахунок того, що у порівнянні з розглянутими виробами-аналогами, розроблений WEB-ресурс надає такі функціональні можливості: зберігання отриманих рекомендацій, збільшення кількості вибору ігор для отримання рекомендацій, покращення рекомендаційної системи за рахунок глибоких нейронних мереж, зберігання даних при перезавантаженні сторінки, збільшенню можливості фільтрації.

## СПИСОК ДЖЕРЕЛ


1. Колісник С. М. Сілагін О. В., « Розробка веб-ресурсу для надання рекомендацій в настільних іграх» в Матеріали конференції «LIII Науково-технічна конференція факультету інтелектуальних інформаційних технологій та автоматиза-

2. Колісник С. М. Сілагін О. В., « Рекомендаційна система на основі моделі перспективи (МН-2024)», Вінниця, 2024. [Електронний ресурс]. Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/mn/mn2024/paper/view/213

20. Н

[Електронний ресурс]. Режим доступу до ресурсу: https://trythesegames.com

7. Collaborative Filtering in Machine Learning [Електронний ресурс]. – Режим

8. Building a Content-Based Recommendation System [Електронний ресурс]. –

9. Video Recommendations at Joyn: Two Tower or Not to Tower, That Was Never

13. Для чого потрібні UML діаграми? [Електронний ресурс]. Режим доступу до ресурсу: https://foxminded.ua/uml-diagramy/

14. Як будувати UML-діаграми. [Електронний ресурс]. Режим доступу до ресурсу: https://dou.ua/forums/topic/40575/

15. Діаграма класів, що робить [Електронний ресурс]. Режим доступу до ре-

[
E9. M
[Електронний ресурс]. Режим доступу до ресурсу: https://sequelize.org/docs/v6/
22. T
24. W
25. A Beginner’s Guide to Latent Dirichlet Allocation(LDA) [Електронний ре-

# ДОДАТКИ



## ДОДАТОК А

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: WEB-ресурс для надання рекомендацій в настільних іграх

НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: WEB-ресурс для надання рекомендацій в настільних іграх

Тип роботи: бакалаврська дипломна робота
(БДР, МКР)

Підрозділ кафедра комп'ютерних наук, ФІІТА
(кафедра, факультет)

**Показники звіту подібності Unicheck**

Оригінальність 98,91% Схожість 1,09%

Аналіз звіту подібності (відмітити потрібне):

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

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

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

Особа, відповідальна за перевірку _____ Озеранський В.С.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Колісник С.М.

Керівник роботи _____ Сілагін О.В.

# ДОДАТОК Б

# ІНСТРУКЦІЯ КОРИСТУВАЧА

Для авторизації користувачу необхідно перейти на сторінку авторизації, де в нього є вибір або зареєструватись, або увійти, вигляд сторінки зображено на рисунку Б.1

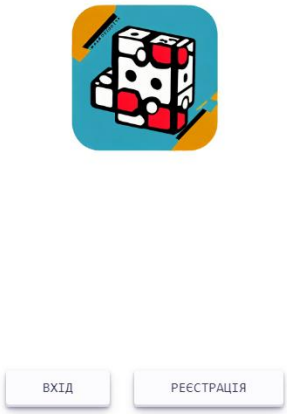



Рисунок Б.1 – Сторінка авторизації

Якщо користувач натиснув кнопку Реєстрація, відкриється форма, у якій, у відведених полях потрібно увести електронну адресу, ім’я, пароль та завантажити аватар. Форма реєстрації зображена на рисунку Б.2.

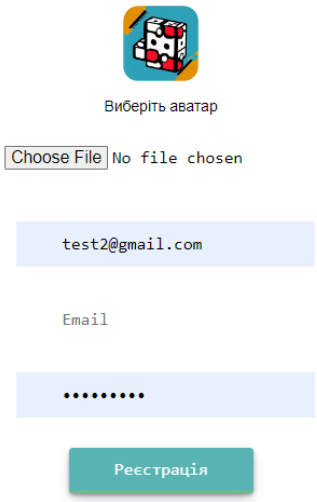



Рисунок Б.2 – Форма реєстрації

Після реєстрації користувач може увійти до свого профілю за допомогою форми входу, використавши електронну пошту та пароль, які він використав при реєстрації. Форма зображена на рисунку Б.3.

Рисунок Б.3 – Форма входу

В самому профілі користувач може за допомогою кнопок вибрати переглянути ігри, які він зберіг чи отримати рекомендації на їх основі(рис. Б.4).

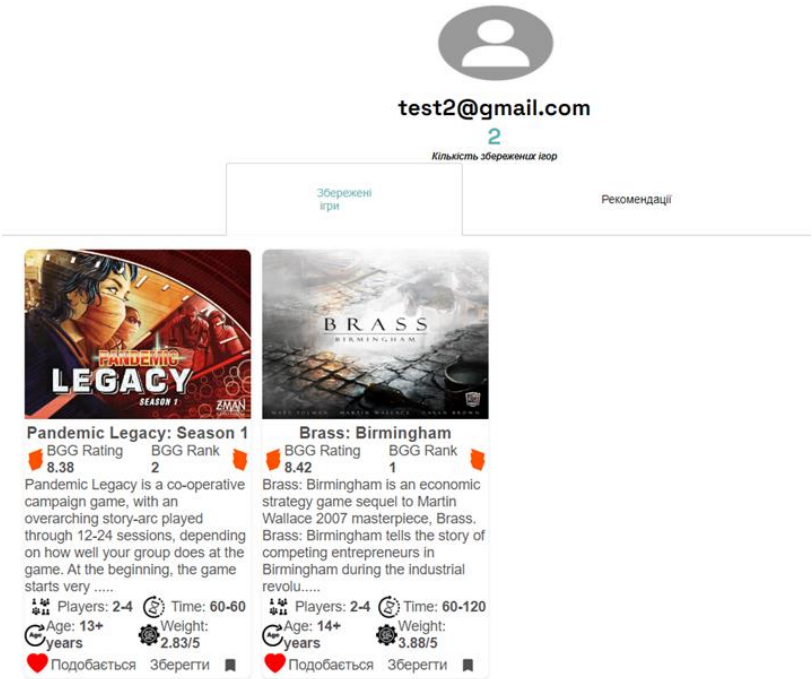



Рисунок Б.4 – Профіль

Для додавання ігор в цей список користувач повинен перейти на вкладку настільні ігри де він побачить індикатори, які ігри він добавив в список тих, що йому подобаються, а які ігри добавив в список збережених. Він може добавляти і забирати ігри прямо в цьому списку (рис. Б.5).

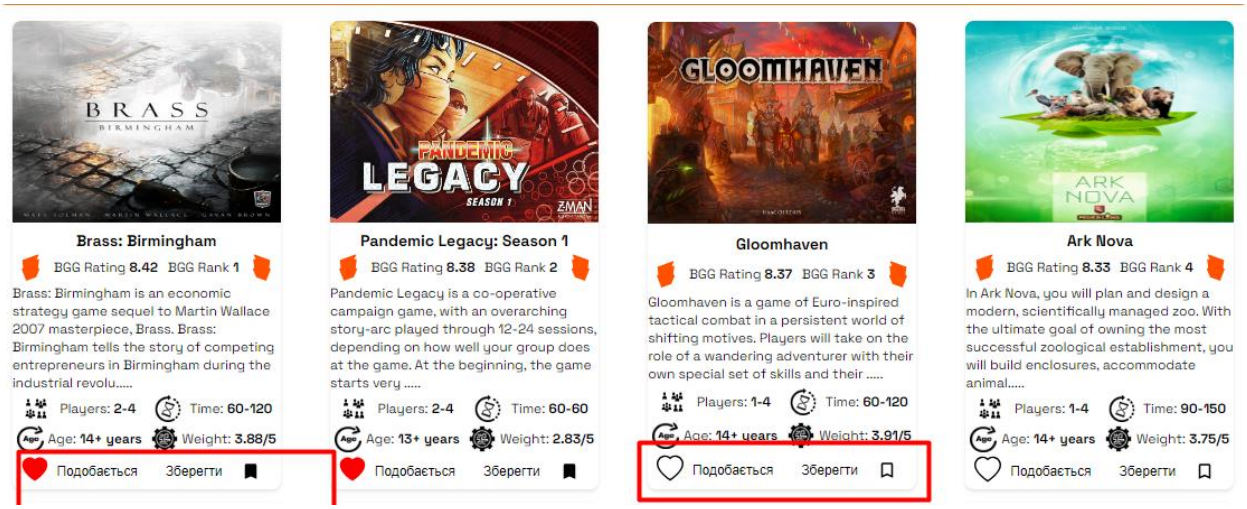



Рисунок Б.5 – Список ігор з кнопками подобається та зберегти

Також у користувачів, користувачі можуть перейти на сторінку рекомендацій, де для отримання рекомендацій вони мають використати пошук або вони до цього мали вибрати ігри з списку ігор, вигляд сторінки рекомендацій зображений на рисунку Б.6

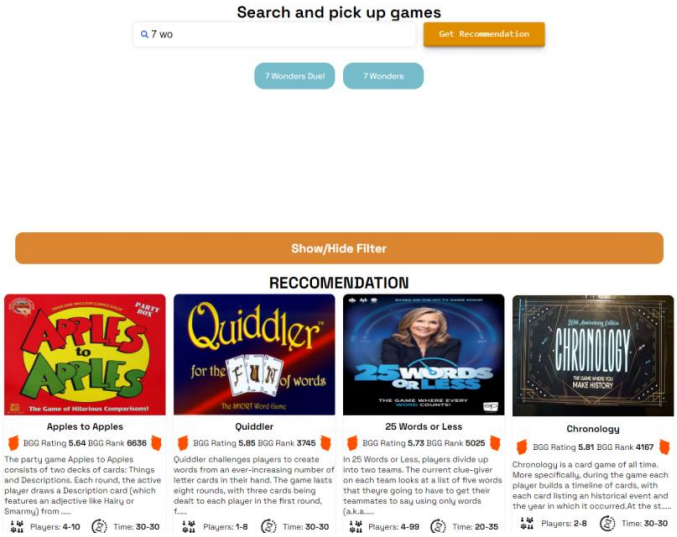



Рисунок Б.6 – Сторінка рекомендацій

За бажанням користувач може відкрити фільтр та відфільтрувати отримані результати.Вигляд фільтру зображено на рисунку Б.7

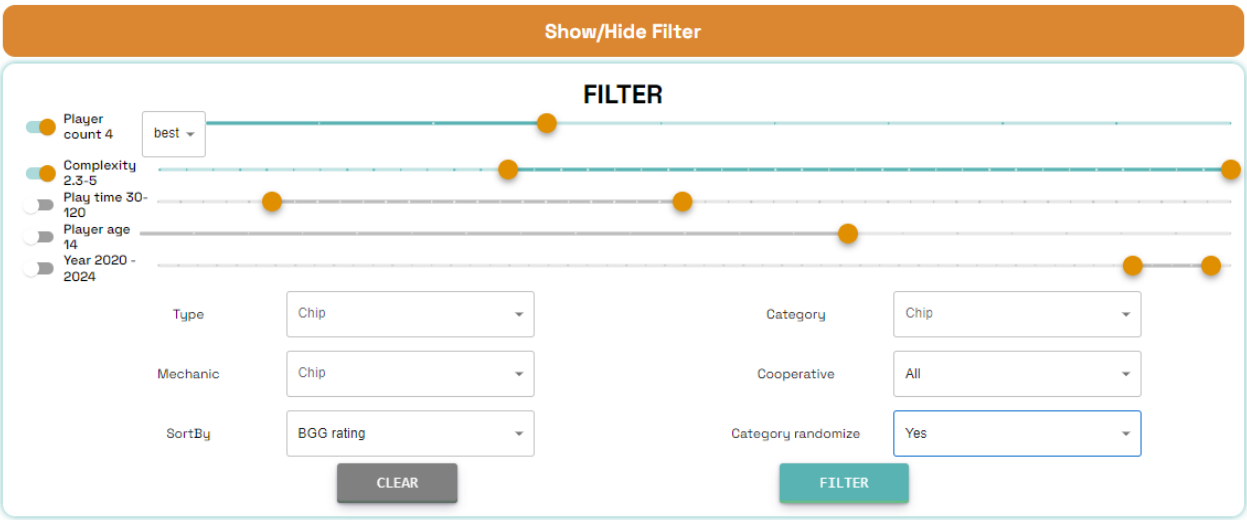



Рисунок Б.7 – Фільтр

# ДОДАТОК В

# ЛІСТИНГ ПРОГРАМИ

**Навігація**

```
const router = createHashRouter([
    {
      path: '/',
      element: (
        <Main
          isAuth={isAuth}
          onLogOut={logoutHandler}
          token={token}
        />
      ),
      errorElement: <ErrorPage />,
    },
    {
      path: '/recommendation',
      element: (
        <Recommendation
        selectedGames = {selectedGames}
        setSelectedGames = {setSelectedGames}
          isAuth={isAuth}
          onLogOut={logoutHandler}
```

```
            token={token}
        />
    ),
    errorElement: <ErrorPage />,
    loader: recommendationLoader,
},
{
    path: '/auth',
    element: <Auth />,
},
{
    path: '/auth/login',
    element: <Login />,
    action: handleLogin,
},
{
    path: '/auth/singup',
    element: <SingUp />,
    action: handleSingUp,
},
{
    path: '/profile',
    element: (
        <Profile
            isAuth={isAuth}
            onLogOut={logoutHandler}
            token={token}
            selectedGames = {selectedGames}
            setSelectedGames = {setSelectedGames}
        />
```

```
        ),
        loader: profileLoader,
        errorElement: <ErrorPage />,
    },
    {
        path: '/boardgames',
        element: (
            <Boardgames
                isAuth={isAuth}
                onLogOut={logoutHandler}
                token={token}
                selectedGames = {selectedGames}
                setSelectedGames = {setSelectedGames}
            />
        ),
        loader: boardgamesLoader,
    },
]);
return <RouterProvider router={router} />;
```

**Серверна частина**

**Рекомендаційний сервіс**

# ДОДАТОК Г

# ГРАФІЧНА ЧАСТИНА

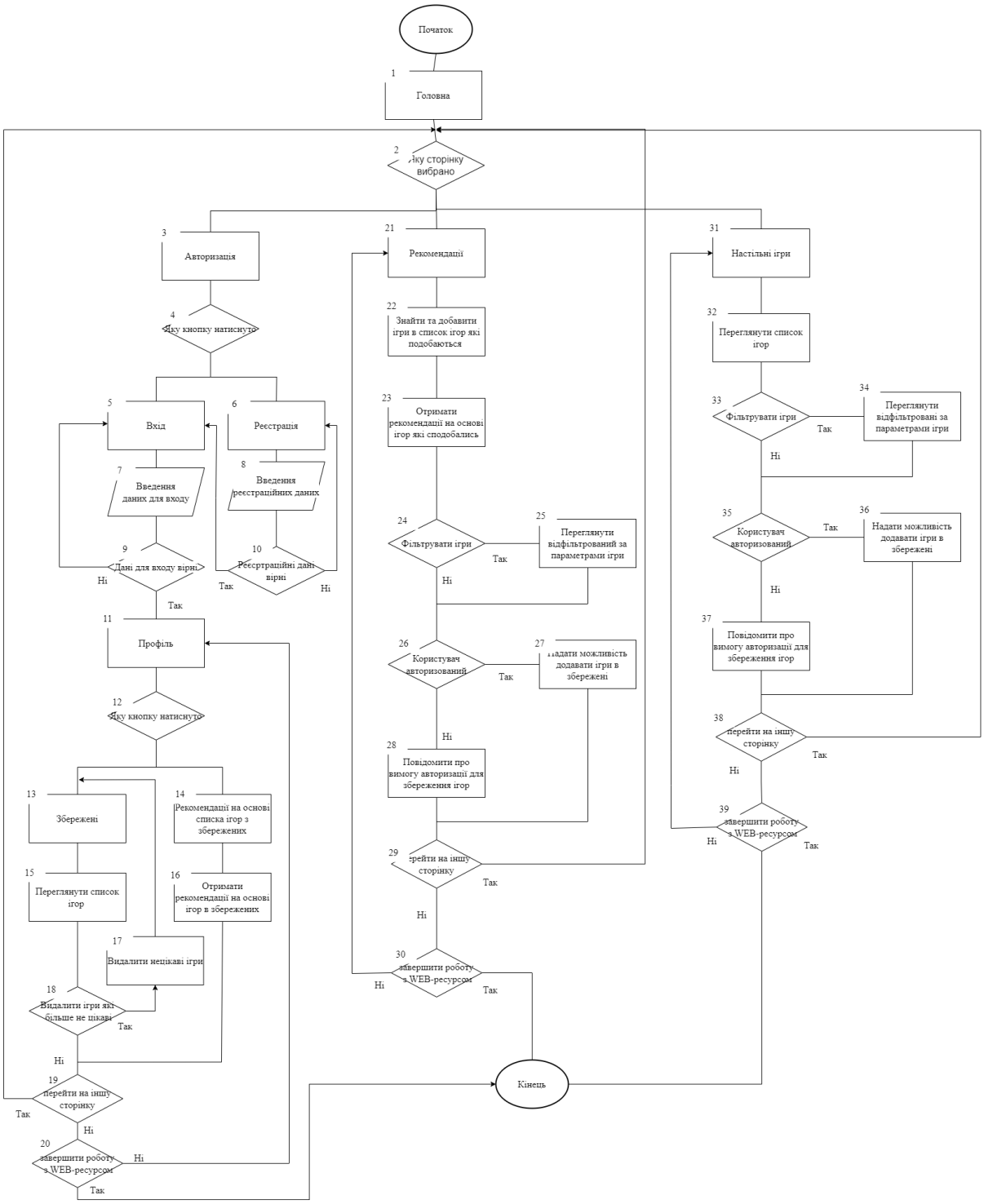



Рисунок Г.1 – Схема алгоритму роботи WEB-ресурсу з надання рекомендацій в настільних іграх

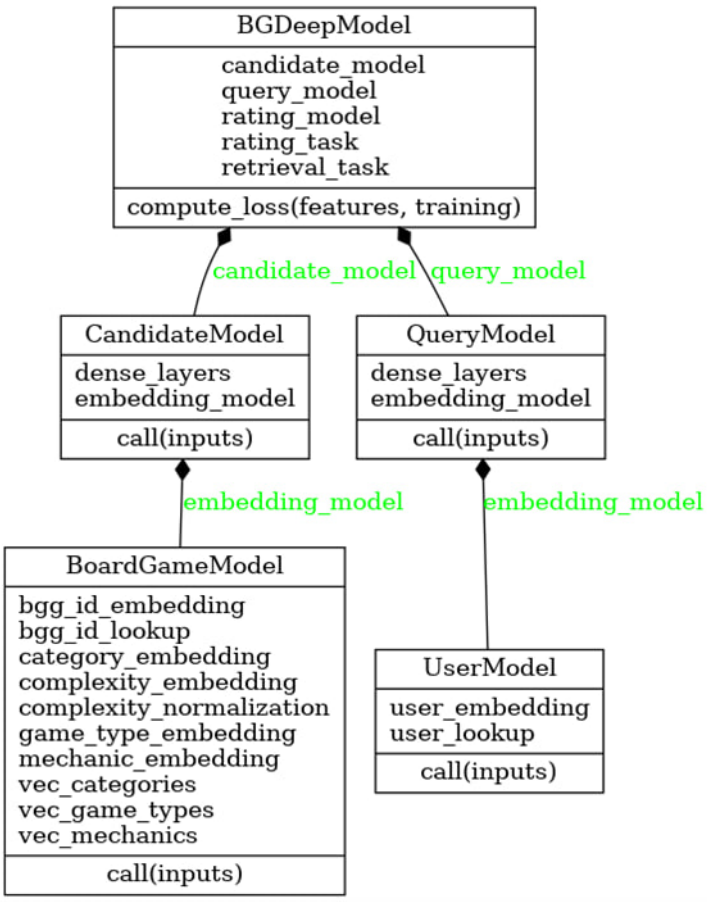



Рисунок Г.2 – Діаграма класів

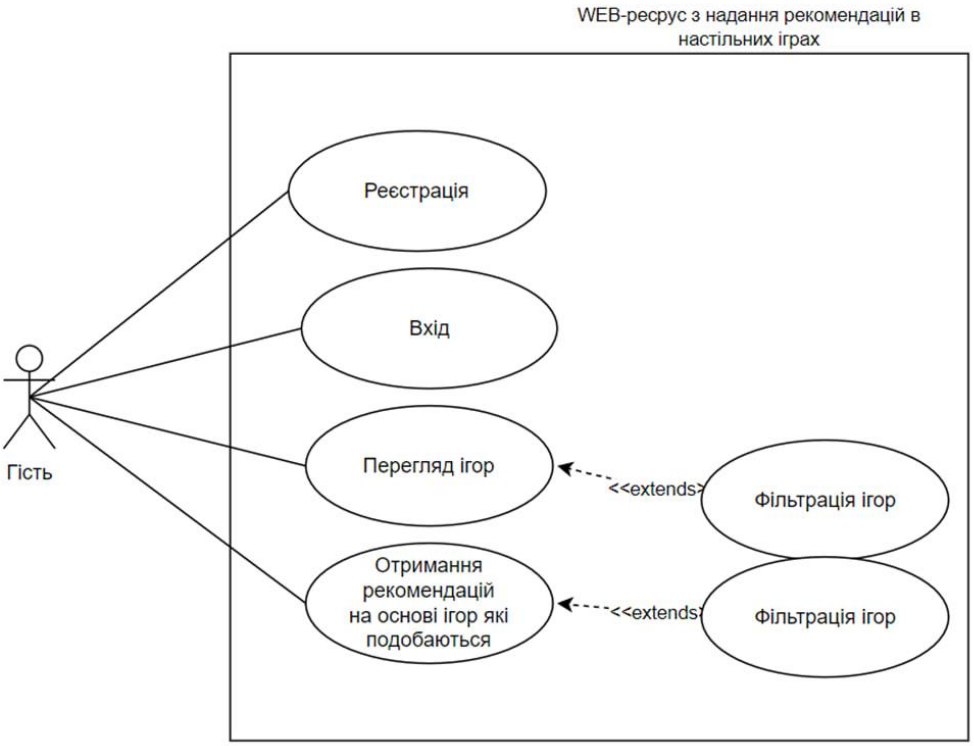



Рисунок Г.3 – Діаграма варіантів використання системи Гість

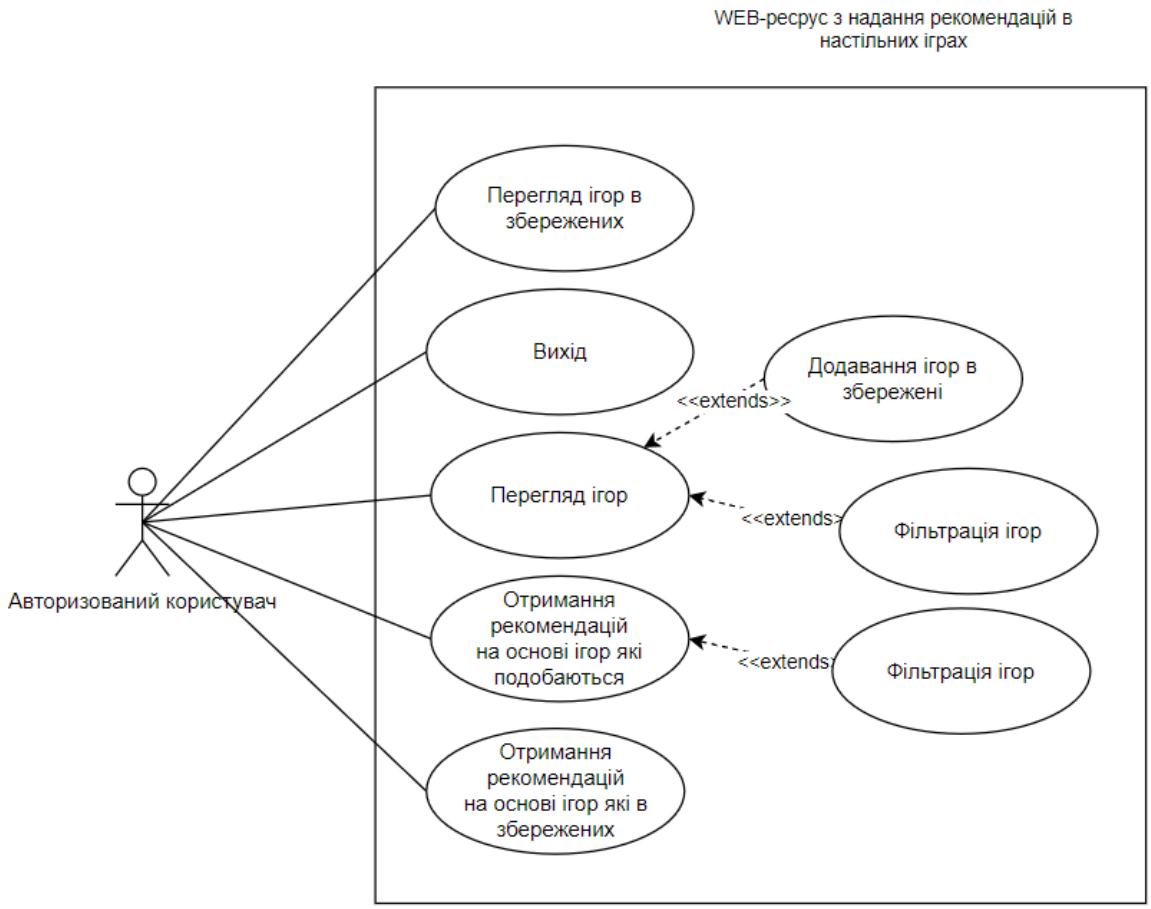



Рисунок Г.4 – Діаграма варіантів використання системи Авторизований користувач

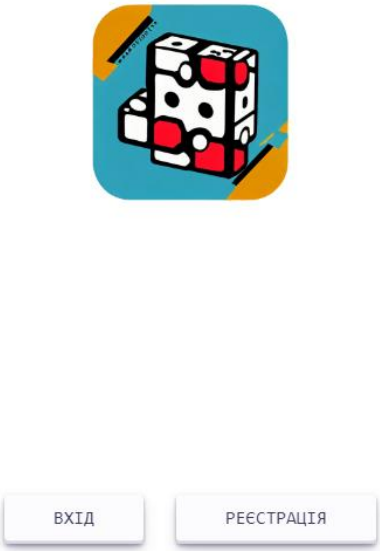



Рисунок Г.6 – Сторінка авторизації

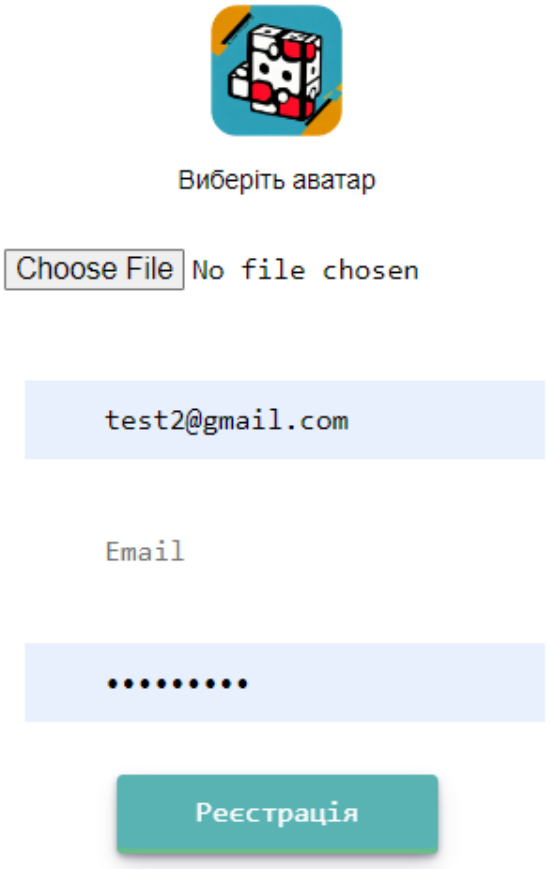



Рисунок Г.7 – Форма реєстрації

Рисунок Г.8 – Форма входу

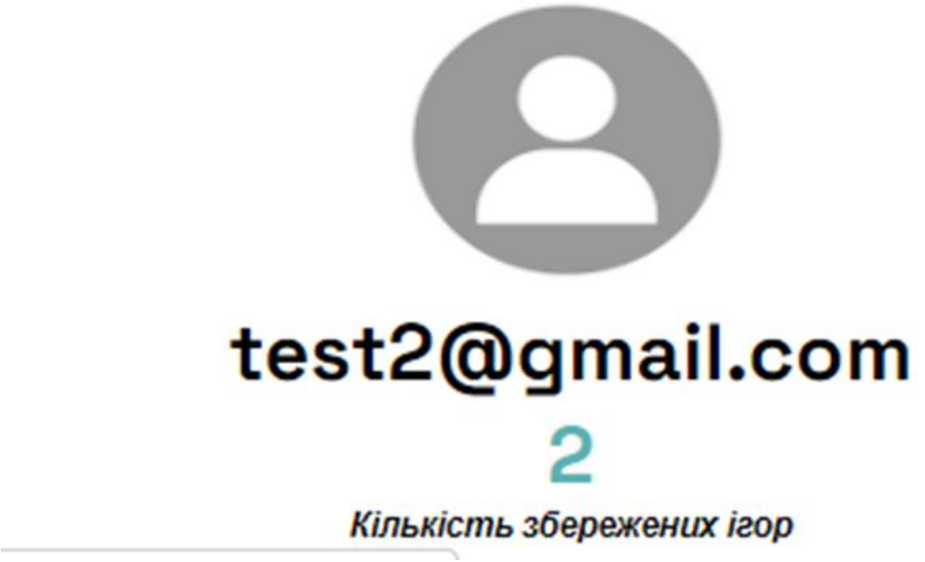



Рисунок Г.8 – Інформація про користувача в профілі

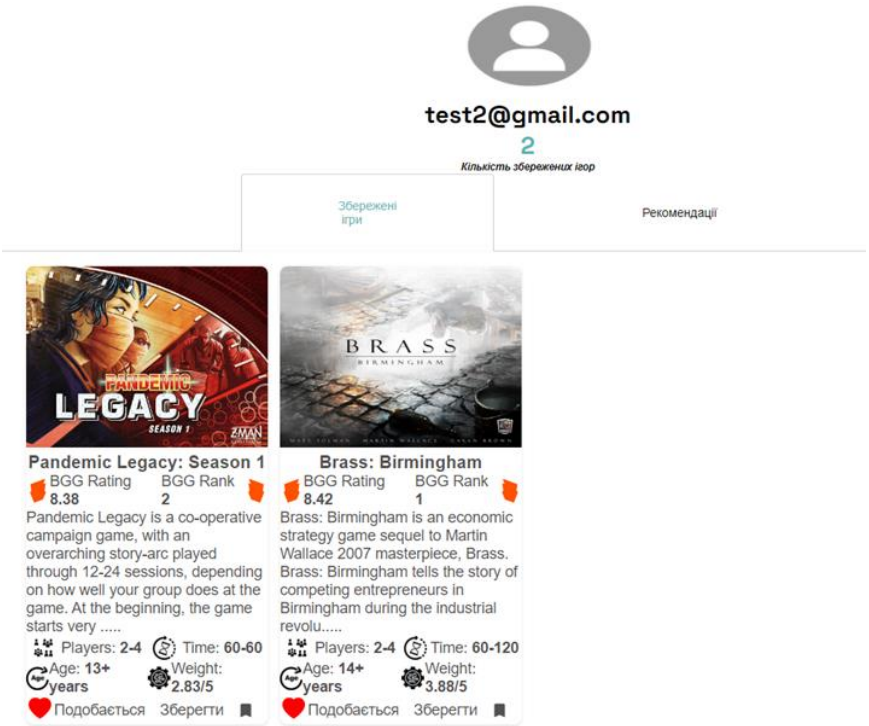



Рисунок Г.9 – Збережені ігри

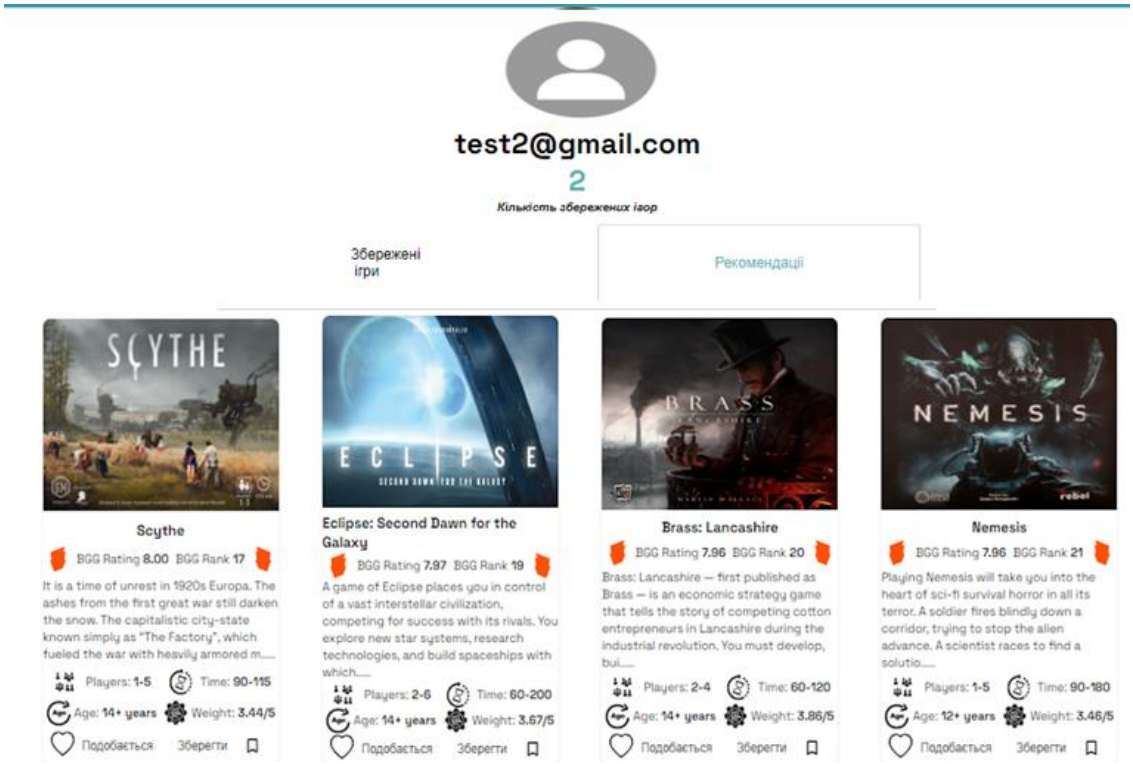



Рисунок Г.10 – Рекомендації на основі збережених ігор

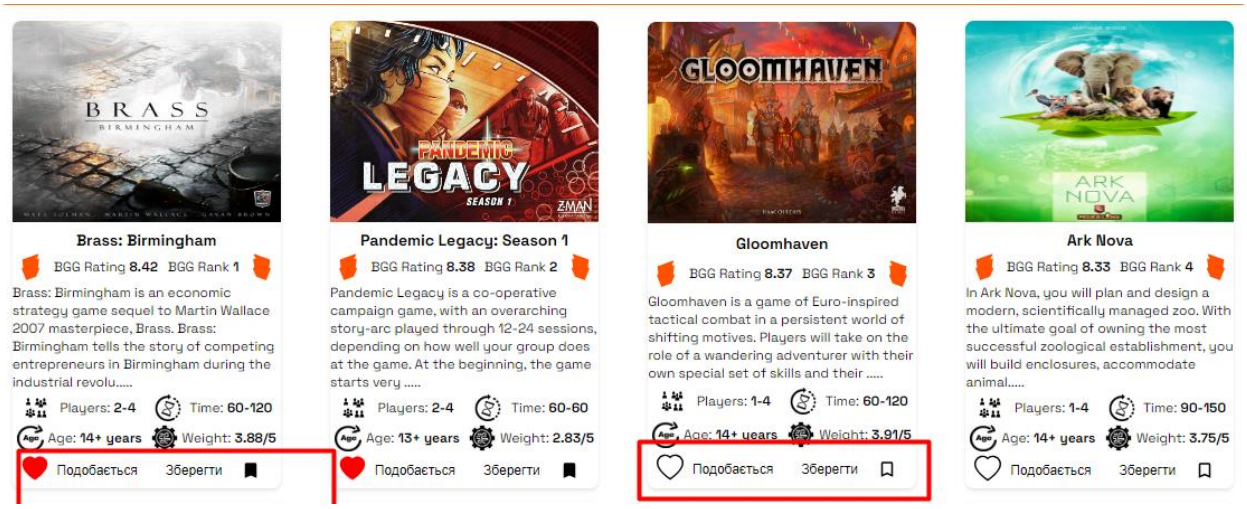



Рисунок Г.11 – Список ігор з кнопками подобається та зберегти

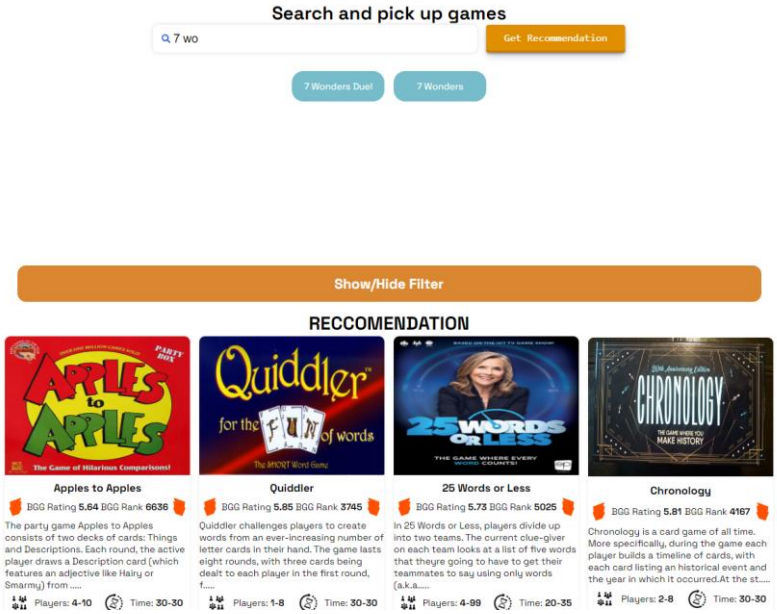



Рисунок Г.12 – сторінка рекомендацій

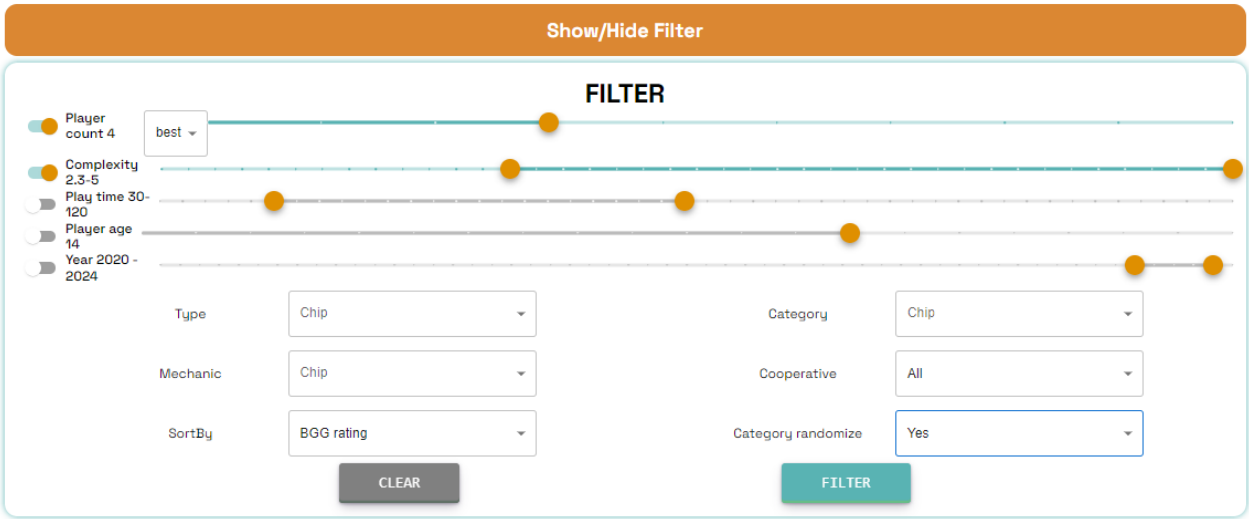



Рисунок Г.13 – Фільтр

**ДОДАТОК Д**

**ДОВІДКА ПРО ВПРОВАДЖЕННЯ**

**ТОВ «Намір Ентерпрайз»**
23310, Вінницька обл., Вінницький р-н., місто Гнівань., вул. Соборна, будинок 66
Тел +38 097 066 5320
email: namir.dev@gmail.com, https://namir.pro/

# ДОВІДКА

Дана Колісніку Сергію Миколайовичу в тому, що результати, одержані ним в процесі виконання бакалаврської дипломної роботи, а саме алгоритм та програмне забезпечення з надання рекомендацій в настільних іграх, планується використати в розробках ТОВ «Намір Ентерпрайз».

Генеральний директор
ТОВ «Намір Ентерпрайз»

підпис

Знаміровський А. А.

Товариство з обмеженою відповідальністю «НАМІР ЕНТЕРПРАЙЗ» 44866044 Україна