ДОДАТОК А

Фрагменти програмного коду клієнтської частини

А.1 Код контексту аутентифікації

```
import React, { createContext, useContext, useState, useEffect }
from 'react';
import { jwtDecode as jwt_decode } from 'jwt-decode';

const AuthContext = createContext();

export const AuthProvider = ({ children }) => {
  const [authToken, setAuthToken] = useState(() =>
localStorage.getItem('authToken'));
  const [userId, setUserId] = useState(null);
  const [userProfile, setUserProfile] = useState(() => {
    const savedProfile = localStorage.getItem('userProfile');
    return savedProfile ? JSON.parse(savedProfile) : null;
  });

  const extractUserIdFromToken = (jwtToken) => {
    try {
      const decoded = jwt_decode(jwtToken);
      return decoded?.PlayerProfileId ?? null;
    } catch (error) {
      console.error('Error decoding token:', error);
      return null;
    }
  };

  useEffect(() => {
    if (authToken) {
      const id = extractUserIdFromToken(authToken);
      setUserId(id);
    } else {
      setUserId(null);
      console.log('No authToken found, userId set to null');
    }
  }, [authToken]);

  const updateUserProfile = (profileData) => {
    setUserProfile(profileData);
    localStorage.setItem('userProfile',
JSON.stringify(profileData));
  };

  const handleLogin = (newAuthToken) => {
    localStorage.setItem('authToken', newAuthToken);
    setAuthToken(newAuthToken);
  };

  const handleLogout = () => {
```

```
    localStorage.removeItem('authToken');
    localStorage.removeItem('userProfile');
    setAuthToken(null);
    setUserId(null);
    setUserProfile(null);
  };

  const isAuthenticated = !!authToken;

  return (
    <AuthContext.Provider value={{ authToken, userId, userProfile,
isAuthenticated, handleLogin, handleLogout, updateUserProfile }}>
      {children}
    </AuthContext.Provider>
  );
};

export const useAuth = () => useContext(AuthContext);
```

А.2 Код інтерналізації для головної сторінки

```
import React, { useState, useEffect } from 'react';
import { Container, Row, Col } from 'react-bootstrap';
import { BsArrowRightCircle } from 'react-icons/bs';
import { useTranslation } from 'react-i18next';

function Banner() {
   const { t } = useTranslation();

   const [loopNum, setLoopNum] = useState(0);
   const [text, setText] = useState('');
   const [isDeleting, setIsDeleting] = useState(false);
   const [delta, setDelta] = useState(300 - Math.random() * 100);

   const toRotate = [
        t('banner.words.players'),
        t('banner.words.fun'),
        t('banner.words.friends'),
   ];
   const period = 2000;

   useEffect(() => {
        const ticker = setInterval(() => {
             tick();
        }, delta);

        return () => clearInterval(ticker);
   }, [text]);

   const tick = () => {
        let i = loopNum % toRotate.length;
        let fullText = toRotate[i];
        let updatedText = isDeleting
```

```
                ? fullText.substring(0, text.length - 1)
                : fullText.substring(0, text.length + 1);

            setText(updatedText);

            if (isDeleting) {
                setDelta(prevDelta => prevDelta / 2);
            }

            if (!isDeleting && updatedText === fullText) {
                setIsDeleting(true);
                setDelta(period);
            } else if (isDeleting && updatedText === '') {
                setIsDeleting(false);
                setLoopNum(loopNum + 1);
                setDelta(300 - Math.random() * 100);
            }
        };

        return (
            <div className='banner'>
                <div className="image-container"></div>
                <Container className="text-center text-container">
                    <Row className="align-items-center">
                    <Col xs={12}>
                        <span
                        className="tagline">{t('banner.welcomeTo')}
                        </span>
                        <span className="navbar-title" style={{
                        marginLeft: "10px"
                        }}>{t('banner.tabletop')}</span>
                        <h1 className="main-heading">
                            {t('banner.findPerfect')} <span
                            className="wrap">{text}</span>
                        </h1>
                        <p
                        className="description">{t('banner.descript
                        ion')}</p>
                        <button className="start-button"
                        onClick={() => console.log("connect")}>
                            {t('banner.getStarted')}
                            <BsArrowRightCircle size={25} />
                        </button>
                    </Col>
                    </Row>
                </Container>
            </div>
        );
    }

    export default Banner;
```

ДОДАТОК Б

Фрагменти юніт та інтеграційного тестування

Б.1 Юніт-тестування аутентифікації

```
import { render, screen, fireEvent, waitFor } from '@testing-
library/react';
import Auth from '../path/to/Auth.jsx';
import { useAuth } from '../components/Context/AuthContext.jsx';
import { BrowserRouter } from 'react-router-dom';

jest.mock('../components/Context/AuthContext.jsx', () => ({
  useAuth: jest.fn(),
}));
jest.mock('react-i18next', () => ({
  useTranslation: () => ({
    t: (key) => key,
  }),
}));
jest.mock('../api/authApi.js', () => ({
  authenticate: jest.fn(),
}));

const { authenticate } = require('../api/authApi.js');

describe('Auth Component', () => {
  const mockHandleLogin = jest.fn();

  beforeEach(() => {
    useAuth.mockReturnValue({
      handleLogin: mockHandleLogin,
      isAuthenticated: false,
    });
  });

  afterEach(() => {
    jest.clearAllMocks();
  });

  const renderComponent = () =>
    render(
      <BrowserRouter>
        <Auth />
      </BrowserRouter>
    );

  test('toggles password visibility', () => {
    renderComponent();

    const passwordInput =
screen.getByLabelText('auth.labels.password');
```

```
    const toggleButton = screen.getByRole('button', { name: /eye
icon/i });

    expect(passwordInput).toHaveAttribute('type', 'password');

    fireEvent.click(toggleButton);

    expect(passwordInput).toHaveAttribute('type', 'text');

    fireEvent.click(toggleButton);

    expect(passwordInput).toHaveAttribute('type', 'password');
  });

  test('shows error if fields are empty', async () => {
    renderComponent();

    fireEvent.click(screen.getByRole('button', { name:
'auth.buttons.login' }));

    await waitFor(() => {

expect(screen.getByText('auth.errors.fillAllFields')).toBeInTheDocu
ment();
    });
  });

  test('successful login clears fields and calls handleLogin',
async () => {
    authenticate.mockResolvedValue({
      success: true,
      token: 'fake-token',
    });

    renderComponent();

    fireEvent.change(screen.getByLabelText('auth.labels.email'), {
target: { value: 'test@example.com' } });
    fireEvent.change(screen.getByLabelText('auth.labels.password'),
{ target: { value: 'Password123' } });

    fireEvent.click(screen.getByRole('button', { name:
'auth.buttons.login' }));

    await waitFor(() => {
      expect(authenticate).toHaveBeenCalledWith(true,
'test@example.com', 'Password123');
      expect(mockHandleLogin).toHaveBeenCalledWith('fake-token');

expect(screen.getByText('auth.messages.loginSuccess')).toBeInTheDoc
ument();
    });
```

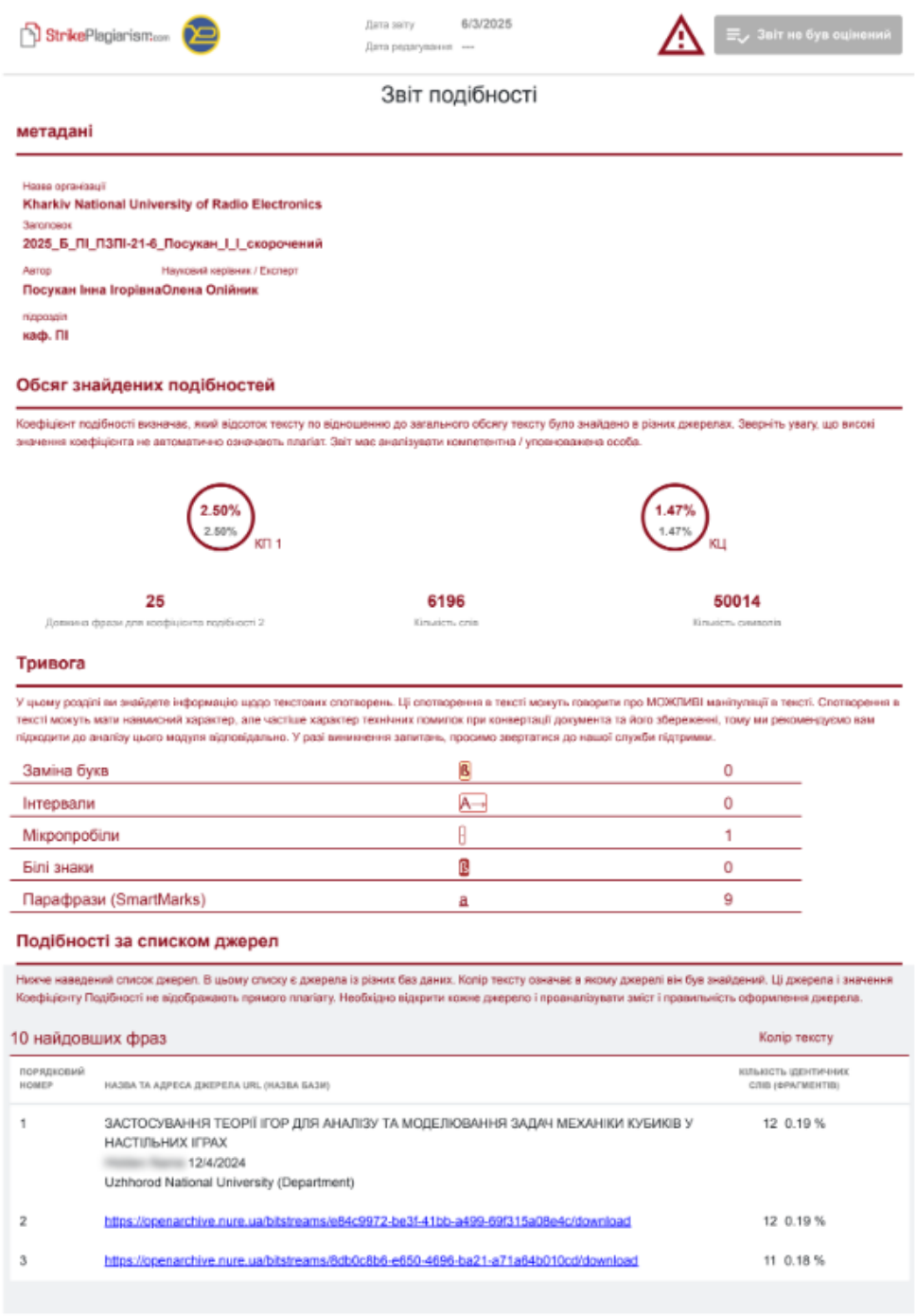
```
expect(screen.getByLabelText('auth.labels.email')).toHaveValue('');

expect(screen.getByLabelText('auth.labels.password')).toHaveValue('
');
  });
});
```

Б.2 Інтеграційне тестування комппоненту «Календар подій»

```
import { render, screen, fireEvent } from '@testing-library/react';
import EventCalendar from './EventCalendar';
import { MemoryRouter } from 'react-router-dom';

const mockNavigate = jest.fn();

jest.mock('react-router-dom', () => ({
  ...jest.requireActual('react-router-dom'),
  useNavigate: () => mockNavigate,
}));

jest.mock('react-i18next', () => ({
  useTranslation: () => ({
    t: (key) => {
      const translations = {
        'calendar.buttonAddEvent': 'Add Event',
      };
      return translations[key] || key;
    },
    i18n: {
      language: 'uk-UA',
    },
  }),
}));

test('TC-EC-001: navigates to /addevent when the "Add Event" button
is clicked', () => {
  render(
    <MemoryRouter>
      <EventCalendar />
    </MemoryRouter>
  );

  const button = screen.getByRole('button', { name: /add event/i });
  fireEvent.click(button);
  expect(mockNavigate).toHaveBeenCalledWith('/addevent');
});
```

ДОДАТОК В

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ



StrikePlagiarism.com

Дата звіту 6/3/2025
Дата редагування ---

Звіт не був оцінений

Звіт подібності

метадані

Назва організації
Kharkiv National University of Radio Electronics
Заголовок
2025_Б_ПІ_ПЗПІ-21-6_Посукан_І_І_скорочений
Автор Науковий керівник / Експерт
Посукан Інна ІгорівнаОлена Опійник
підрозділ
каф. ПІ

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

2.50% 2.50% КП 1

1.47% 1.47% КЦ

25	6196	50014
Довжина фрази для коефіцієнта подібності 2	Кількість слів	Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв		0
Інтервали		0
Мікропробіли		1
Білі знаки		0
Парафрази (SmartMarks)	a	9

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	ЗАСТОСУВАННЯ ТЕОРІЇ ІГОР ДЛЯ АНАЛІЗУ ТА МОДЕЛЮВАННЯ ЗАДАЧ МЕХАНІКИ КУБИКІВ У НАСТІЛЬНИХ ІГРАХ [redacted] 12/4/2024 Uzhhorod National University (Department)	12 0.19 %
2	https://openarchive.nure.ua/bitstreams/e84c9972-be3f-41bb-a499-69f315a08e4c/download	12 0.19 %
3	https://openarchive.nure.ua/bitstreams/8db0c8b6-e650-4696-ba21-a71a64b010cd/download	11 0.18 %

Рисунок В.1 – Звіт результатів перевірки на унікальність тексту

ДОДАТОК Г

Слайди презентації

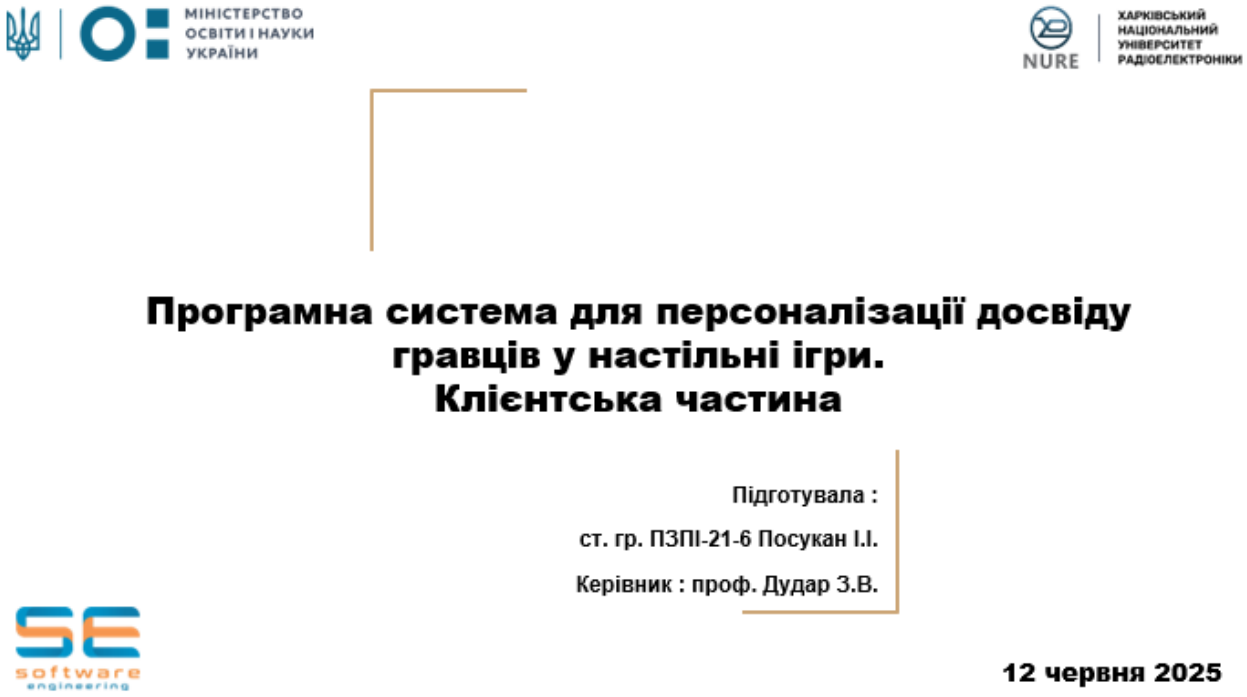


Рисунок Г.1 – Слайд №1 (рисунок виконано самостійно)

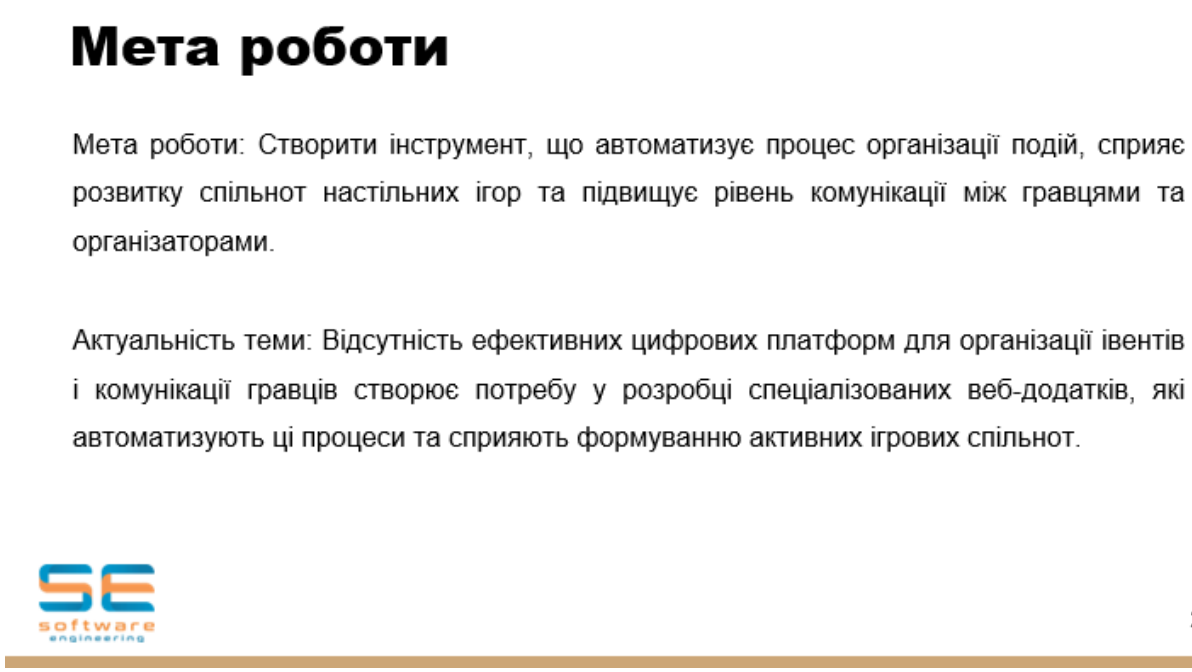


Рисунок Г.2 – Слайд №2 (рисунок виконано самостійно)

Аналіз проблеми

В результаті аналізу конкурентів було виявлено наступні проблеми:

- відсутність ефективних інструментів для організації та управління подіями у настільних іграх;
- обмежені можливості пошуку івентів за географічним розташуванням;
- відсутність персоналізованих рекомендацій ігор та подій для користувачів.

Рисунок Г.3 – Слайд №3 (рисунок виконано самостійно)

Аналіз існуючих рішень

Назва критерію	Назва ресурсу			
	Tabletop (наша платформа)	BGG	Try These Games	Geeker
Кількість ігор	Висока	Дуже висока	Середня	Середня
Рекомендації	Висока	Низька	Низька	Низька
Можливість грати онлайн	Ні	Ні	Ні	Ні
Організація подій	Так	Ні	Ні	Так
Підтримка багатомовності	Так	Так	Ні	Так
Спільнота	Активна	Дуже активна	Не активна	Активна
Пошук ігор за місцезнаходженням	Так	Ні	Ні	Так

Рисунок Г.4 – Слайд №4 (рисунок виконано самостійно)

Постановка задачі

Розробити веб-платформу Tabletop, що забезпечує зручну організацію та участь у заходах, пов'язаних із настільними іграми, шляхом реалізації основних функціональних можливостей для користувачів та адміністраторів.

Опис основних функцій платформи:

- авторизація і реєстрація;
- створення та керування подіями;
- реєстрація на події;
- пошук та фільтрація івентів;
- редагування профілю;
- адміністрування системи.

Рисунок Г.5 – Слайд №5 (рисунок виконано самостійно)

Опис системи

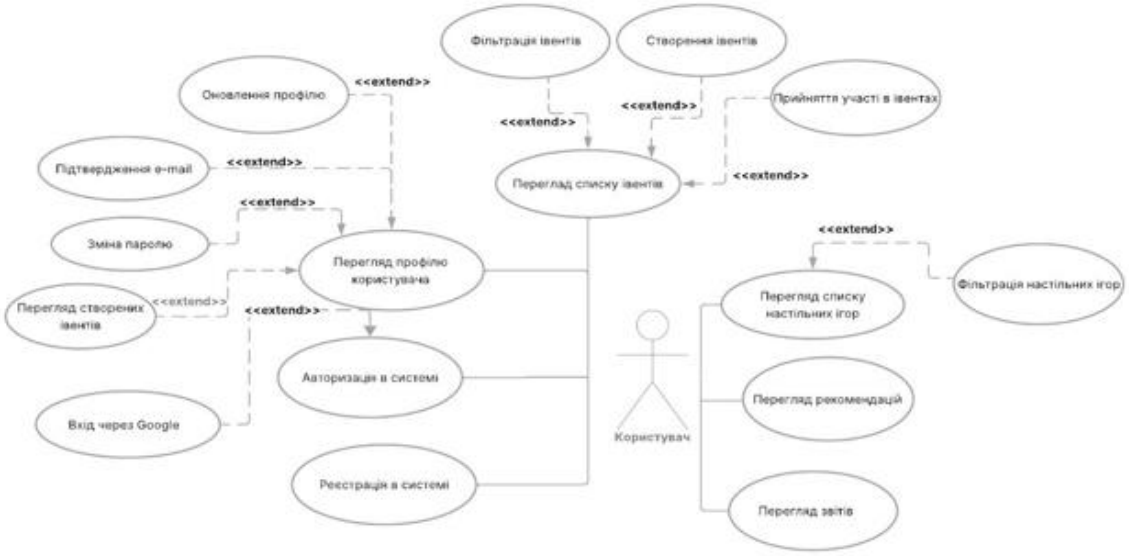
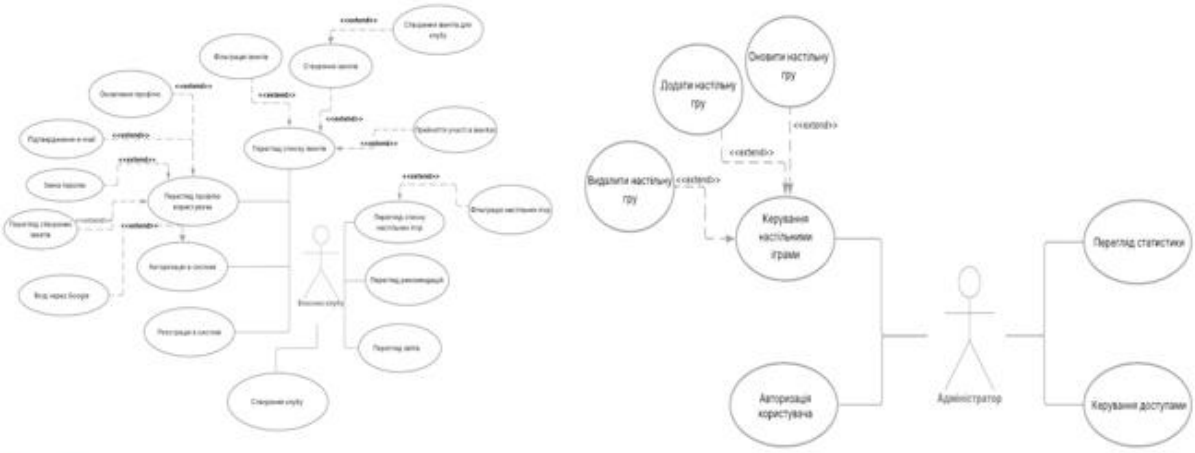


Рисунок Г.6 – Слайд №6 (рисунок виконано самостійно)

Опис системи



software engineering

Рисунок Г.7 – Слайд №7 (рисунок виконано самостійно)

Вибір технологій розробки

React Bootstrap

React

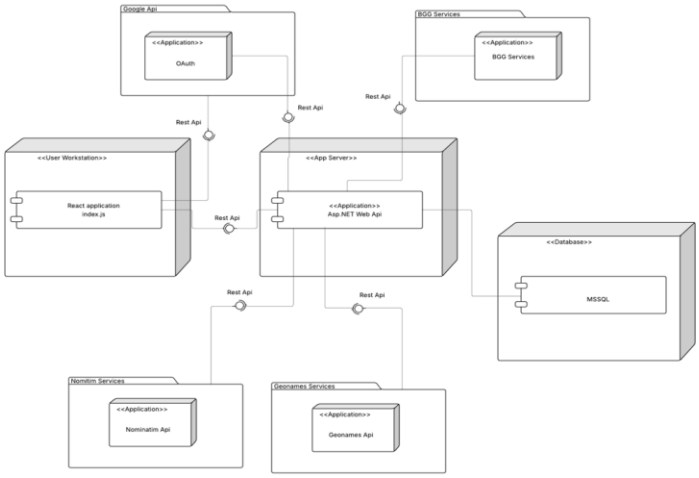
Vite

software engineering

Рисунок Г.8 – Слайд №8 (рисунок виконано самостійно)

Архітектура створенного програмного забезпечення

Архітектура побудована за клієнт-серверною моделлю. Взаємодія між клієнтом та сервером здійснюється через обмін HTTP-запитами.



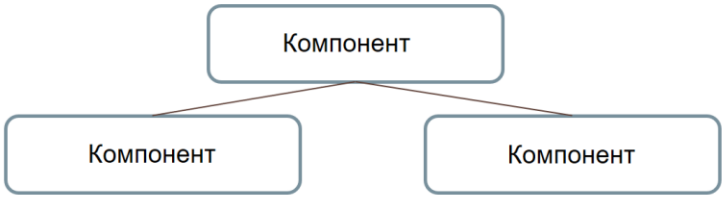
software engineering

Рисунок Г.9 – Слайд №9 (рисунок виконано самостійно)

Компонентний підхід

React побудований за принципом компонентної архітектури, де кожна частина інтерфейсу — це окремий компонент.

- Кожен компонент має власну логіку та вигляд.
- Компоненти можна повторно використовувати в різних частинах застосунку.
- Це спрощує розробку, тестування та супровід коду.
- Компоненти об'єднуються у складніші структури, утворюючи повний інтерфейс.

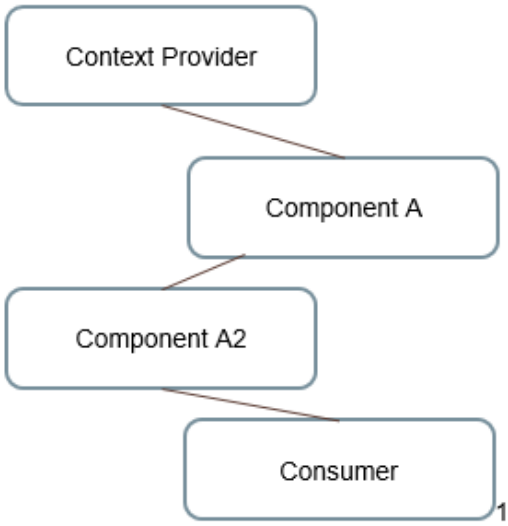


10

Рисунок Г.10 – Слайд №10 (рисунок виконано самостійно)

Управління станом за допомогою Context Api

Context API — інструмент React для передачі даних між компонентами без прокидання пропсів. Дозволяє створити глобальний контекст і забезпечує зручний доступ до спільного стану в будь-якій частині додатку.



11

Рисунок Г.11 – Слайд №11 (рисунок виконано самостійно)

Маршрутизація в веб-застосункі


```jsx
import { Routes, Route } from 'react-router-dom';
import Home from "./pages/Home";
import Auth from "./pages/Auth.jsx";
import Events from './pages/Events.jsx';
import Profile from './pages/Profile.jsx';
import AddEvent from './pages/AddEvent.jsx';
import AllBoardGames from './pages/AllBoardGames.jsx';
import EventDetail from './pages/EventDetail.jsx';
import EventCalendar from './pages/EventCalendar.jsx';
import AdminPanel from './pages/AdminPanel.jsx';
import EmailConfirmationPage from './pages/ConfirmEmail.jsx';
import ResetPasswordConfirm from './pages/ResetPassword.jsx';

const AppRouter = () => {
  return (
    <Routes>
      <Route path="/" element={<Home />} />
      <Route path="/login" element={<Auth />} />
      <Route path="/events" element={<Events />} />
      <Route path="/profile" element={<Profile />} />
      <Route path="/events/addevent" element={<AddEvent />} />
      <Route path="/board-games" element={<AllBoardGames />} />
      <Route path="/events/:id" element={<EventDetail />} />
      <Route path="/calendar" element={<EventCalendar />} />
      <Route path="/adminpanel" element={<AdminPanel />} />
      <Route path="/confirm-email/confirm" element={<EmailConfirmationPage />} />
      <Route path="/reset-password/confirm" element={<ResetPasswordConfirm />} />

    </Routes>
  );
};

export default AppRouter;
```

software engineering

Рисунок Г.12 – Слайд №12 (рисунок виконано самостійно)

Відправка запитів на сервер

```jsx
export async function getJoinedEvents() {
  const authToken = localStorage.getItem("authToken");

  try {
    const response = await axios.get(`${BASE_URL}/PlayerProfiles/participated-events`, {
      headers: {
        Authorization: `Bearer ${authToken}`,
        "Content-Type": "application/json",
      },
    });
    return { success: true, data: response.data };
  } catch (error) {
    const message =
      error.response?.data?.message || error.message || "Unknown error";
    return { success: false, message };
  }
}
```

AXIOS

Рисунок Г.13 – Слайд №13 (рисунок виконано самостійно)

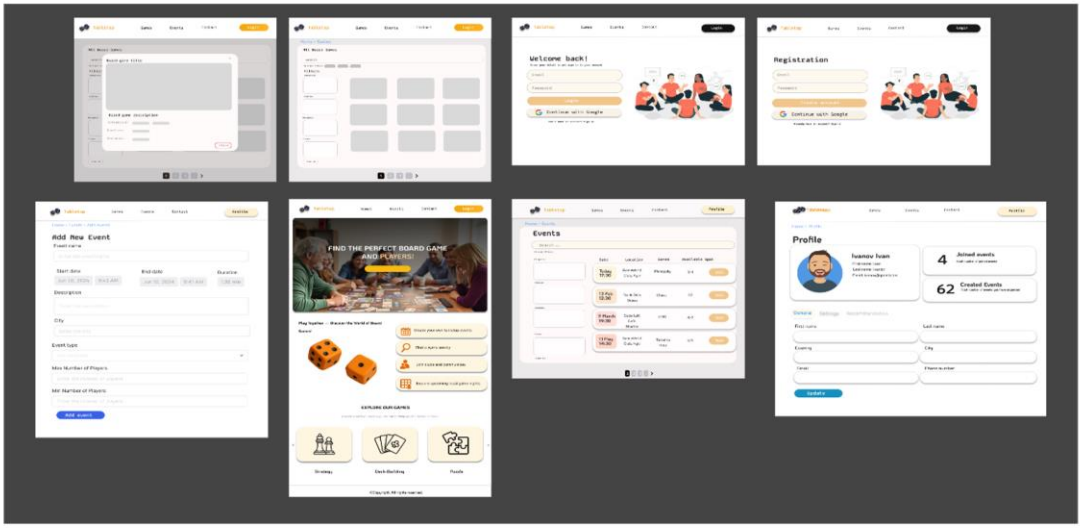
Локалізація

Для забезпечення багатомовної підтримки у проекті використовується бібліотека i18next — популярний інструмент для локалізації React-додатків.

14

Рисунок Г.14 – Слайд №14 (рисунок виконано самостійно)

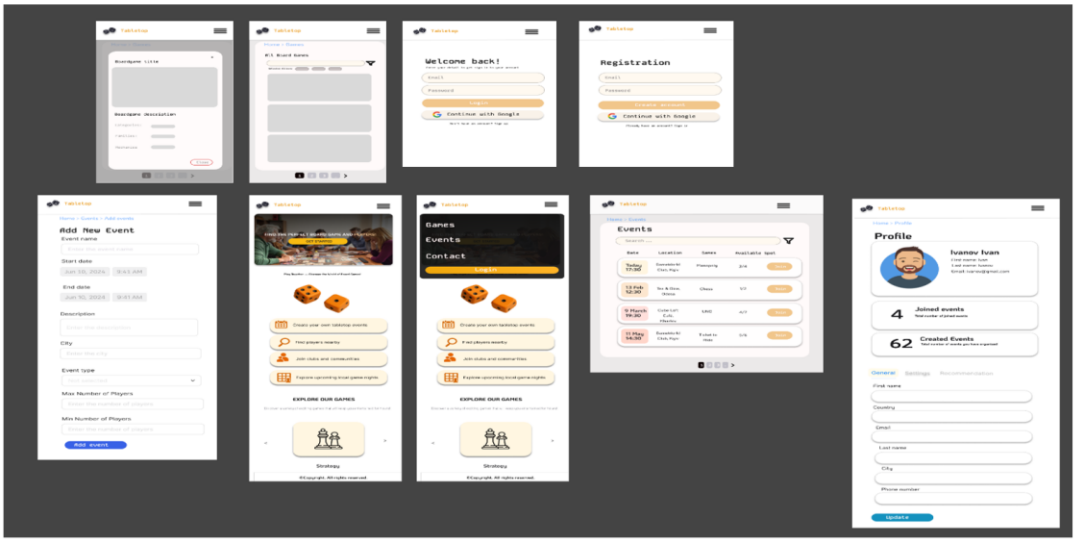
Проєктування дизайну десктопної версії



15

Рисунок Г.15 – Слайд №15 (рисунок виконано самостійно)

Проєктування дизайну мобільної версії



16

Рисунок Г.16 – Слайд №16 (рисунок виконано самостійно)

Інтерфейс головної сторінки

SE software engineering

17

Рисунок Г.17 – Слайд №17 (рисунок виконано самостійно)

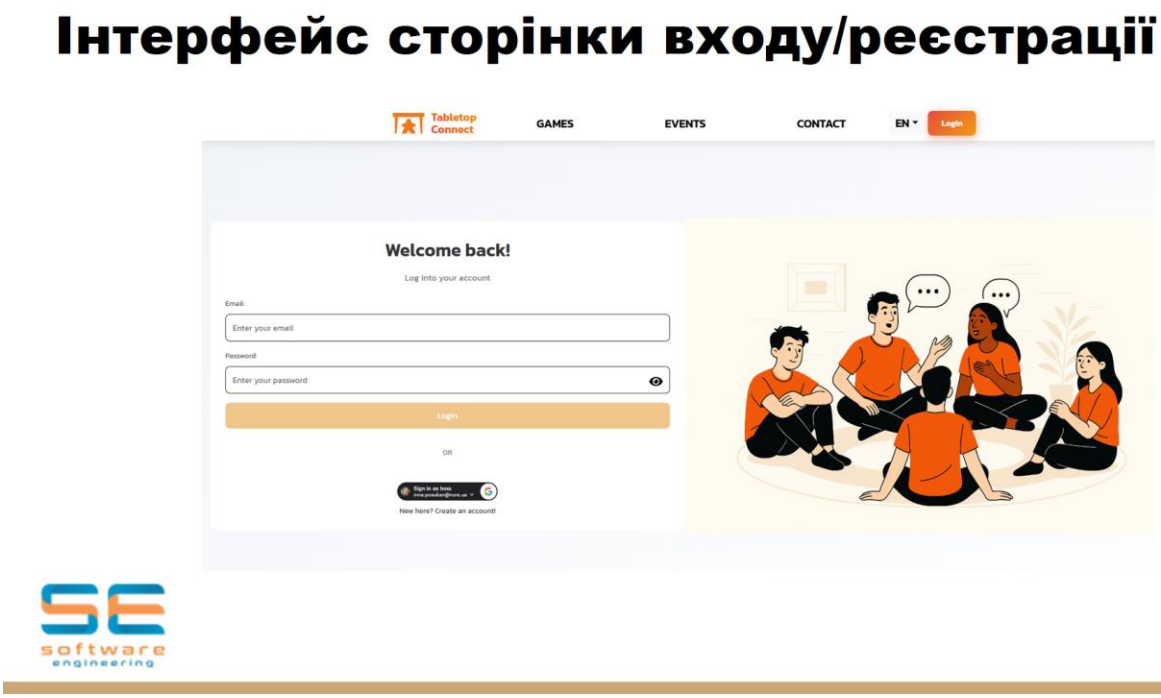


Рисунок Г.18 – Слайд №18 (рисунок виконано самостійно)

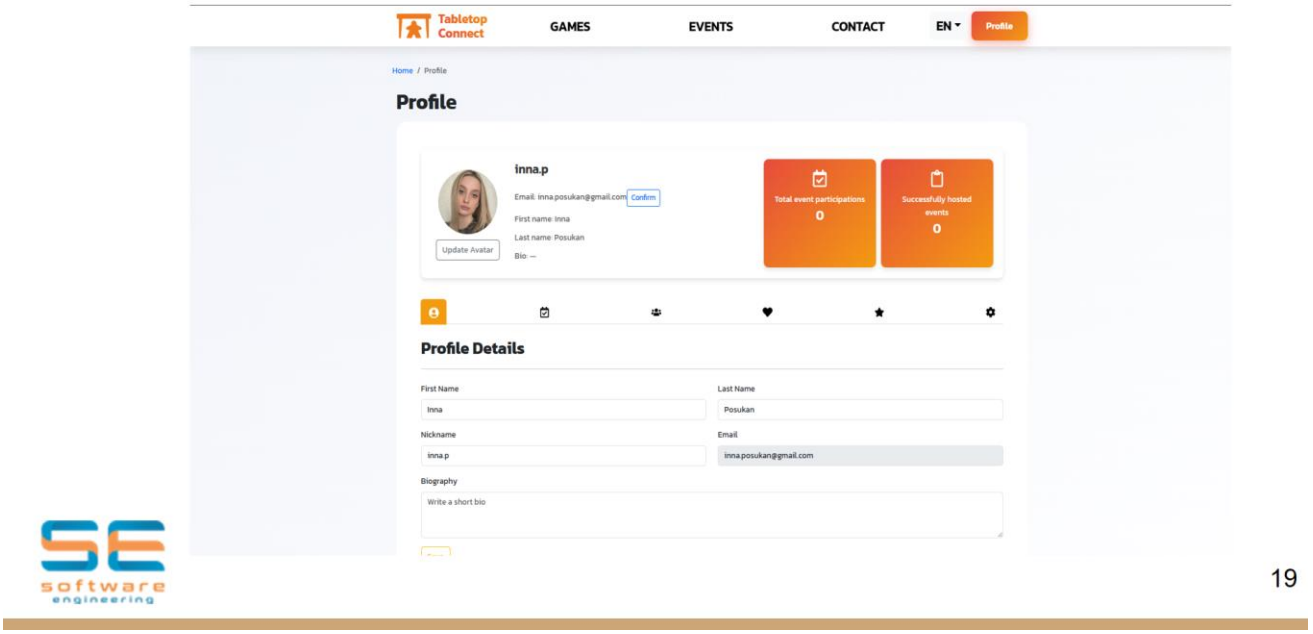


Рисунок Г.19 – Слайд №19 (рисунок виконано самостійно)

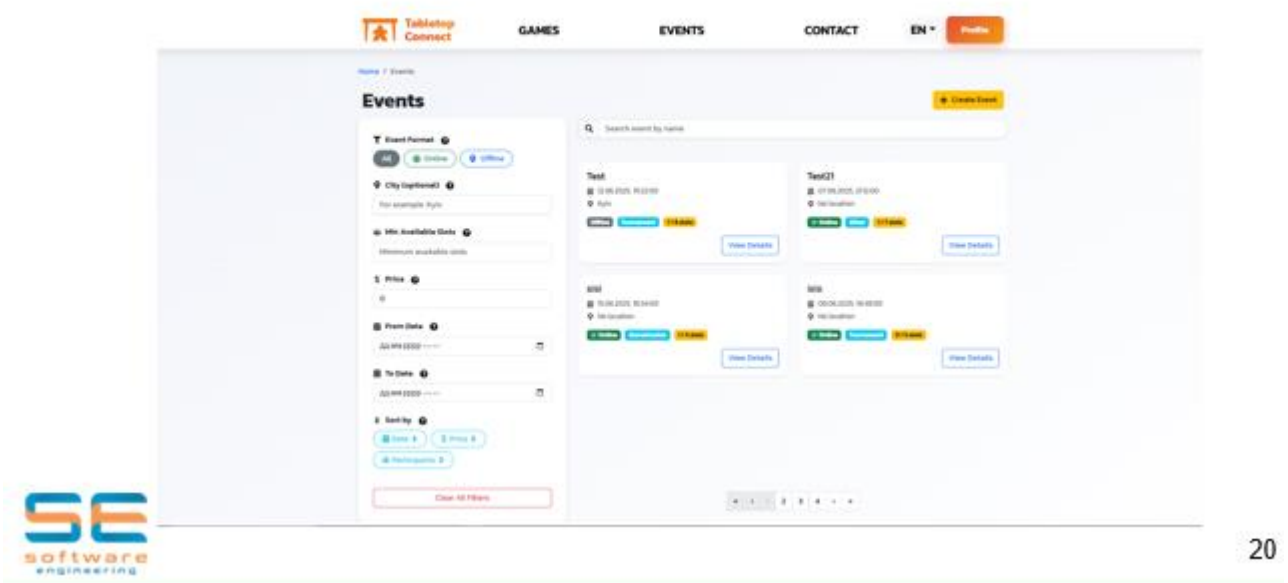


Рисунок Г.20 – Слайд №20 (рисунок виконано самостійно)

Тестування (мануальне)

Опис	Очікуваний результат	Результат
Авторизувати зареєстрованого користувача в системі з використанням валідних даних	Користувач успішно авторизований в системі	Виконано
Створити івент, зареєструвавшись під роллю «Regular User»	Івент успішно створений і відображається у списку івентів	Виконано
Сформувати звіт по учасникам конкретного івенту	Звіт по учасникам відображається у створеному звіті	Виконано
Фільтрація івенту за датою «06.06.2025»	У списку івентів відображаються івенти, які відбудуться «06.06.2025»	Виконано

Рисунок Г.21 – Слайд №21 (рисунок виконано самостійно)

Публікація результатів

Рисунок Г.22 – Слайд №22 (рисунок виконано самостійно)

Підсумки

- Створено програмну систему для персоналізації досвіду гравців у настільні ігри.
- Реалізовано основні функції: організація подій, управління клубами, багатомовність, адаптивний дизайн.
- Використано сучасні технології фронтенду — React, React Router, Context API, i18next.
- Розвиток у перспективі : монетизація та мобільний застосунок

Рисунок Г.23 – Слайд №23 (рисунок виконано самостійно)

ДОДАТОК Д

Копії тез доповіді на 29-му міжнародному молодіжному форумі

«Радіоелектроніка та молодь У XXI столітті»

УДК 004.89

ДОСЛІДЖЕННЯ ВИКОРИСТАННЯ АЛГОРИТМІВ ДЛЯ ПЕРСОНАЛІЗОВАНИХ РЕКОМЕНДАЦІЙ НАСТІЛЬНИХ ІГОР

Цвик В.І., Посукан І. І., Ляпота В.М.
email: vladyslav.tsvyk@nure.ua, inna.posukan@nure.ua, vitaliy.lyapota@nure.ua
Харківський національний університет радіоелектроніки, каф. ПІ
м. Харків, Україна

This work is devoted to the analysis of recommendation algorithms for board games based on the content-based approach. The study explores various recommendation methods, including vector-based similarity calculations and hybrid techniques to improve accuracy. The problem of cold start and data normalization is addressed, ensuring better personalization for users. Additionally, the effectiveness of the proposed system is assessed using precision, recall, and F1-score metrics. The research highlights the importance of combining multiple filtering techniques for enhancing recommendation relevance.

На сьогоднішній день сучасні програмі системи все частіше включають в себе рекомендаційні підсистеми для персоналізації контенту та підвищення залученості користувачів. Вони використовуються в різних сферах, зокрема в електронній комерції, пошукових сервісах, соціальних мережах та освітніх платформах. В останні роки зростає інтерес до впровадження рекомендаційних алгоритмів у сфері настільних ігор, що обумовлено значною різноманітністю ігрового контенту та необхідністю допомагати користувачам у виборі відповідних ігор. Традиційні методи пошуку нових настільних ігор, такі як рейтинги або відгуки, мають певні обмеження, оскільки вони не враховують індивідуальні вподобання гравців та їхній попередній досвід.

Основною проблемою у створенні такої системи є велика кількість параметрів, що характеризують ігри, а також необхідність ефективного аналізу вподобань користувачів. У цьому дослідженні розглядаються методи побудови рекомендаційних систем для настільних ігор, зокрема підходи, засновані на контентній фільтрації (Content-based Filtering) [1], а також можливості поєднання різних алгоритмів для підвищення точності рекомендацій.

Серед основних методів для побудови рекомендаційних систем можна виділити наступні:

– контентна фільтрація (Content-Based Filtering) – рекомендація об’єктів (настільних ігор) на основі їхніх характеристик та схожості з тим, що вже сподобалося користувачу;

– колаборативна фільтрація (Collaborative Filtering) – рекомендації, що базується на даних про взаємодію користувачів з об’єктами, не враховуючи їх характеристик;

– гібридні методи (Hybrid) – поєднання контентної та колаборативної фільтрації для забезпечення більшої релевантності рекомендації.

При відсутності великої кількості даних про взаємодію користувачів варто використовувати Content-Based Filtering [2], наприклад, у випадку запуску системи, де початковими даними є лише дані про настільні ігри та їх характеристики. У такій програмній системі варто передбачити можливість оцінювати об’єкти взаємодії користувачами. Отримавши такі дані можна застосувати методи Collaborative Filtering і об’єднавши їх з попередніми алгоритмами сформувати гібридну рекомендаційну систему, яка б поєднувала обидва підходи.

При застосуванні Content-Based Filtering потрібно враховувати такі властивості настільних ігор, як популярність, складність, ігрові механіки, тематика, категорії, вікові обмеження, тривалість партії, кількість гравців, залежність від знання мови (кількість важливої текстової інформації). Також необхідно врахувати попередній досвідом гравця, а саме зі список ігор, які сподобалися користувачу. Одним з джерел даних, яке б містило може стати сервіс BoardGameGeek, де зібрано дані більше ніж про 20000 настільних ігор [3].

Кожна настільна гра може бути представлена у вигляді векторного профілю, що містить її характеристики. Основне завдання алгоритму – порівняти ці профілі з іграми, які вже сподобалися користувачу, та знайти найбільш схожі варіанти. Для цього спочатку будується матриця ознак, де кожен рядок відповідає певній грі, а стовпці містять значення таких параметрів, як складність, тематика, механіки, категорії, тривалість партії, популярність тощо.

Одним із найпоширеніших способів обчислення схожості між іграми є використання косинусної подібності (cossine similarity) [4] між їхніми векторними представленнями. Косинусна подібність визначає ступінь схожості між двома векторами за допомогою косинуса кута між ними. Чим ближче значення до 1, тим більше схожі ігри. Інший підхід – евклідова відстань, яка визначає наскільки близько розташовані дві гри у багатовимірному просторі ознак. Для категоріальних ознак (наприклад, механіки, тематика, жанри) можна використовувати методи One-Hot Encoding або TF-IDF для текстових описів, що дозволяє будувати багатовимірне представлення гри та порівнювати їх між собою.

Важливим аспектом є нормалізація числових параметрів, таких як рейтинг, складність чи тривалість партії, оскільки вони мають різні масштаби. Наприклад, рейтинг може мати значення від 1 до 10, тоді як тривалість партії – від 5 до 300 хвилин. Щоб уникнути домінування окремих параметрів, застосовується min-max нормалізація або z-score стандартизація.

Одним із викликів цього підходу є проблема холодного старту, коли у нового користувача немає історії вподобань, і система не має достатньо інформації для формування рекомендацій. У таких випадках можна комбінувати Content-Based Filtering з іншими методами, наприклад, використовувати популярні ігри з високими рейтингами або враховувати схожість з профілями інших користувачів (елементи колаборативної фільтрації).

Після впровадження рекомендаційної системи варто визначити, наскільки добре розроблений алгоритм пропонує користувачам релевантні ігри. Для цього можна використовувати такі метрики, як Precision, Recall та F1-score, які аналізують, яка частка рекомендованих ігор дійсно цікава користувачу (Precision) і наскільки повно система покриває всі можливі релевантні варіанти (Recall). F1-score комбінує ці два показники, дозволяючи збалансовано оцінити модель.

Для реалізації описаних вище алгоритмів Content-Based Filtering існує декілька підходів і технологій які можуть бути використані.

По-перше, можна використовувати мову Python, бібліотеки для машинного навчання та роботи з аналізом даних, такі як Sciki-learn, Pandas та NumPy.

По-друге, для платформи .NET існують бібліотеки ML.NET та MathNet для роботи з машинним навчанням та великими числовими даними.

По-третє, існують хмарні (cloud) рішення, такі як Google Cloud AI чи Azure AI для розробки та навчання моделей машинного навчання [5].

Отже, при проектуванні програмної системи рекомендації настільних ігор варто ретельно підійти до проектування та вибору алгоритмів. У випадку відсутності даних про взаємодію користувача з об’єктами варто використовувати Content-Based Filtering, який у процесі розвитку системи можна скомбінувати з Collaborative Filtering.

Список використаних джерел:

1. Ms. Tejashri Sharad Phalle. Content Based Filtering And Collaborative Filtering: A Comparative Study. Journal of Advanced Zoology. 2024. Vol. 45. P. 96–100.

2. Marwa Hussien Mohamed, Mohamed Khafagy, Mohamed Hasan Ibrahim. Recommender Systems Challenges and Solutions Survey. International Conference on Innovative Trends in Computer Engineering. URL: http://dx.doi.org/10.1109/ITCE.2019.8646645 (date of access: 04.03.2025).

3. Phil Woodward, Sam Woodward. Mining the Boardgamegeek. Significance. 2019. Vol. 16, Issue 5. P. 24–29.

4. Muhammad Falah, Dewi Handayani. Content-based filtering using cosine similarity algorithm for alternative selection on training programs. Journal of Soft Computing Exploration. 2023. Vol. 4. P. 204 – 212.

5. Макєєв О., Кравець Н. Дослідження методів створення сервісно-орієнтованих програмних систем у Azure. Computer Systems and Information Technologies. 2023. №2(11). С. 38–47.