

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»

Факультет комп'ютерних наук та технологій
Кафедра комп'ютерних систем та мереж

Пояснювальна записка

до дипломного проекту (роботи)

магістра

(ступінь вищої освіти)

на тему АДАПТИВНА СИСТЕМА ОНЛАЙН-МІНІГОР

Виконав: студент 2 курсу, групи КНТ-612м
спеціальності 123 Комп'ютерна інженерія
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Спеціалізовані комп'ютерні системи

БОЙКО Р.В.

(ПРІЗВИЩЕ та ініціали)

Керівник ТЯГУНОВА М.Ю.

(ПРІЗВИЩЕ та ініціали)

Рецензент МАЛИЙ О.Ю.

(ПРІЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет Комп'ютерних наук і технологій
Кафедра Комп'ютерні системи та мережі
Ступінь вищої освіти магістерський
Спеціальність 123 Комп'ютерна інженерія
(код і найменування)
Освітня програма (спеціалізація) «Спеціалізовані комп'ютерні системи»
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Зав. кафедри Кудерметов Р.К.

“ ” 2023 року

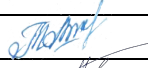
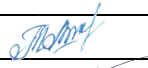
З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

БОЙКО Ростислава Володимировича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Адаптивна система онлайн-мініігор
- керівник проєкту (роботи) к. т. н., доцент, ТЯГУНОВА Марія Юріївна
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)
- затверджені наказом вищого навчального закладу від “24” жовтня 2023 року № 400
2. Строк подання студентом проєкту (роботи) 10 грудня 2023 року
3. Вихідні дані до проєкту (роботи) опис предметної області, існуючі системи-аналоги онлайн-мініігор, концепція адаптивної системи, технічна документація фреймворків та бібліотек
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1) Аналіз предметної області;
2) Проєктування системи;
3) Реалізація системи;
4) Тестування та впровадження.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4	ТЯГУНОВА М.Ю., доцент		
нормоконтроль	ЩЕРБАК Н.В., ст. викл.		

7. Дата видачі завдання 01.10.2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Аналіз популярних вебресурсів, які надають мініігри	10.10.2023 р.	
2	Визначення характеристик адаптивності системи	15.10.2023 р.	
3	Загальний опис системи	20.10.2023 р.	
4	Визначення вимог до розроблюваної системи	05.10.2023 р.	
5	Проектування системи	20.11.2023 р.	
6	Впровадження та тестування системи	25.11.2023 р.	
7	Оформлення графічного матеріалу	01.12.2023 р.	
8	Оформлення допоміжного матеріалу	10.12.2023 р.	

Студент

Керівник проєкту (роботи)


(підпис)

Ростислав БОЙКО
(ім'я, ПРІЗВИЩЕ)


(підпис)

Марія ТЯГУНОВА
(ім'я, ПРІЗВИЩЕ)

РЕФЕРАТ

ПЗ: 71 с., 28 рис., 20 джерел.

АВТОРИЗАЦІЯ, АКАУНТ, БРАУЗЕР, ВЕБРЕСУРС, ДИЗАЙНЕР,
МІНІГРИ, МОДЕРАТОР, ОНЛАЙН-ГРА, РЕЄСТРАЦІЯ, РОЗРОБНИК

Об'єкт дослідження – вебресурси з онлайн мінііграми.

Предмет дослідження – адаптивні властивості онлайн-мініігор.

Метою роботи є розробка адаптивної системи онлайн-мініігор, яка здатна підлаштовуватись під конкретних гравців та враховувати їх вподобання, а також враховувати загальні світові тренди.

Проєкт складається з чотирьох розділів.

У першому розділі описано ідеї ігрового бізнесу, а саме онлайн-ігор. Проведено аналіз вебресурсів зі схожою ідеєю. Описується ігрова індустрія, розробка ігор та їх актуальність. Також визначено характеристики адаптивності системи та визначено вимоги до проєкту.

У другому розділі описано проєктування системи, а саме архітектури та інтерфейсу користувача. Також обрано технології для розробки та реалізації системи. Було обрано React JS в якості бібліотеки для створення вебдодатку та Firebase для зберігання даних та взаємодії з ними.

У третьому розділі наведено програмний код для реалізації системи, описано, як працюють реалізовані методи та підходи.

У четвертому розділі описується впровадження та тестування вебресурсу онлайн-мініігор, а також описано проведену SEO-оптимізацію.

У результаті виконання дипломної роботи проведено аналіз, проєктування, реалізацію та тестування системи онлайн-мініігор у вигляді вебресурсу.

ABSTRACT

EN: 71 p., 28 figures, 20 sources.

AUTHORIZATION, ACCOUNT, BROWSER, WEB RESOURCE, DESIGNER,
MINI-GAMES, MODERATOR, ONLINE GAME, REGISTRATION, DEVELOPER

The object of the study is web resources with online mini-games.

The subject of the study is the adaptive properties of online mini-games.

The aim of the work is to develop an adaptive system of online mini-games that can adapt to specific players and take into account their preferences, as well as take into account general global trends.

The project consists of four sections.

The first section describes the ideas of the gaming business, namely online games. It analyzes web resources with a similar idea. It describes the gaming industry, game development, and their relevance. The characteristics of the system's adaptability are also defined and the requirements for the project are determined.

The second section describes the design of the system, namely the architecture and user interface. We also selected technologies for the development and implementation of the system. React js was chosen as a library for creating a web application and Firebase for storing and interacting with data.

The third section presents the program code for implementing the system, describes how the implemented methods and approaches work.

The fourth chapter describes the implementation and testing of the online mini-games web resource, as well as the SEO optimization performed.

As a result of the thesis, the analysis, design, implementation, and testing of an online mini-games system in the form of a web resource were carried out.

ЗМІСТ

Вступ	7
1 Аналіз предметної області	10
1.1 Аналіз популярних вебресурсів, які надають мініігри	10
1.2 Визначення характеристик адаптивності системи мініігор	13
1.3 Загальний опис системи	15
1.4 Визначення вимог до розроблюваної системи	16
1.5 Очікувані переваги від запропонованих підходів	17
2 Проєктування системи	20
2.1 Проєктування архітектури програмного забезпечення	20
2.2 Проєктування інтерфейсу користувача	22
2.3 Вибір технологій для створення проєкту	29
3 Реалізація системи	37
3.1 Реєстрація та авторизація користувачів	37
3.2 Реалізація головної сторінки	43
3.3 Реалізація адаптивних властивостей системи	45
3.4 Реалізація оцінювання гри	54
4 Тестування та впровадження системи	55
4.1 Вигляд сайту	55
4.2 Тестування сайту	57
4.3 Просування системи в мережі Інтернет	63
4.4 Оптимізація	65
Висновки	68
Перелік джерел посилання	70

ВСТУП

З розвитком інтернет-технологій ігри перейшли в онлайн-середовище та стали доступними для мільйонів користувачів по всьому світу. Онлайн-мініігри – це короткі та прості ігри, доступні через веббраузер, мобільний пристрій чи соціальні платформи. Загалом буде розглянуто загальну характеристику онлайн-мініігор та їх роль у сучасному інтернет-середовищі.

У онлайн-мініігор є широкий спектр різноманітних жанрів. Найпопулярнішими можна виділити такі жанри:

- аркади: ігри, орієнтовані на швидку реакцію та спритність;
- головоломки: завдання, які потребують логічного мислення та вирішення головоломок;
- симулятори: ігри, що імітують різні аспекти реального життя;
- змагальні: ігри, в основному, про змагання серед різних видів спорту.

Це всього лише декілька основних категорій, на які можуть бути розподілені онлайн-мініігри, насправді таких жанрів може бути набагато більше.

Онлайн-мініігри є легко доступними для широкої аудиторії користувачів інтернету. Їхня перевага в тому, що для гри не потрібно завантажувати або встановлювати спеціальне програмне забезпечення. Гравці можуть почати гру прямо з будь-якого встановленого браузера, тобто користувач не витрачає час на завантаження та встановлення та сама мінігра не займає місце на його комп'ютері.

Багато хто відноситься не серйозно до ігор, особливо до браузерних інтерпретацій, однак онлайн-мініігри відіграють багато важливих ролей. Наприклад, розваги, відпочинок, навчання, розвиток, інструмент для реклами, психологічні аспекти. По-перше, ігри дозволяють користувачам відволіктися від повсякденних турбот і розслабитись, провести час із задоволенням. Деякі онлайн-мініігри розроблені з урахуванням освітніх цілей. Вони можуть допомогти розвинути логічне мислення, пам'ять, координацію рухів та інші навички. Це особливо важливо для дітей, оскільки ігри можуть бути не тільки захоплюючими,

а й корисними для їхньої освіти. До прикладу тут можна привести ігри з категорії головоломок, де потрібно задіювати своє логічне мислення. Багато компаній використовують онлайн-мініігри як інструмент маркетингу та реклами. Ігри можуть супроводжуватися брендованою рекламою або сюжетами, які просувають продукти або послуги компанії. Це дозволяє привернути увагу аудиторії та збільшити впізнаваність бренду. Варто зазначити, що реклама може бути різною, добре, коли вона не є сильно нав'язливою та зустрічається не на кожному кроці у грі та не заважає геймплею. Онлайн-мініігри впливають на психологічний стан гравців. Вони здатні викликати емоції, включаючи радість, задоволення, азарт, а також стрес у разі невдачі або залежність у разі надмірного захоплення. Тому важливо використовувати ігри з помірністю та дотримуватися здорових кордонів.

Онлайн-мініігри у браузері залишаються популярними та продовжують розвиватися, щоб задовольнити очікування сучасних гравців. Зараз є декілька ключових та важливих тенденцій розвитку, які можна виділити. По-перше, це графічна складова, тому що завжди приємніше дивитись на якісне, плавне зображення, де немає пікселів та розмиття зображення з ривками. З розвитком технологій та браузерів, розробники можуть створювати більш якісні та графічно просунуті мініігри. Це включає 2D і 3D графіку, анімацію, спецефекти та інші візуальні поліпшення. Ігри стають більш привабливими та захоплюючими. Соціальні функції ігор, такі як змагатися та спілкуватися, можливість ділитися результатами з друзями стають все більш важливими в онлайн-іграх. Ігри інтегруються з соціальними мережами та платформами, що дозволяє гравцям легко підключатися до своїх друзів та обмінюватись ігровим успіхом та досвідом. Моделі монетизації онлайн-мініігор включають мікротранзакції, рекламу і преміум-підписки. Гравці можуть вибирати, як підтримувати ігри, і в деяких випадках це може дозволити грати безкоштовно. Важливо, щоб моделі монетизації були збалансовані, щоб не дратувати гравців.

Фактор адаптивності системи відіграє ключову роль у сфері підбору рекомендованих ігор для користувача, що допомагає покращити досвід гравця та підвищити ефективність монетизації. Адаптивні системи також допомагають

збільшити монетизацію ігрових платформ та магазинів. Вони можуть рекомендувати ігри, в яких гравець схильний здійснювати покупки з огляду на його історію покупок та переваги. Таким чином, обрана тема магістерської роботи є актуальною.

Мета роботи - розробка адаптивної системи онлайн-мініігор, яка здатна підлаштовуватись під конкретних гравців та враховувати їх вподобання, а також враховувати загальні світові тренди.

Для досягнення поставленої мети необхідно виконати такі завдання:

- проаналізувати переваги і недоліки існуючих популярних вебресурсів для гри у мініігри;
- розробити вимоги до власної адаптивної системи з урахуванням проведеного аналізу;
- обрати програмне забезпечення для створення власної системи;
- реалізувати адаптивну систему онлайн-мініігор;
- проаналізувати отримані результати.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз популярних вебресурсів, які надають мініігри

У цьому розділі буде розглянуто вебсайти, на яких можна пограти в онлайн-мініігри та наведено їх переваги та недоліки.

1.1.1 Вебресурс A10.com

A10.com - це один з найпопулярніших вебресурсів, що надають онлайн-ігри. Цей вебсайт пропонує широкий вибір безкоштовних мініігор у різних жанрах, включаючи аркади, кооперативи, головоломки, пригоди, гонки, стрілялки, та багато інших. A10.com також пропонує ігри для різного віку та інтересів, включаючи дітей та дорослих.

На цьому вебресурсі можна знайти як популярні класичні ігри, такі як ігри з швидким сюжетом та головоломки, так і новіші ігри з сучасною графікою та захоплюючим геймплеєм. Вони пропонують різноманітні ігри, які підходять для короткострокових ігрових сеансів або довгих ігрових марафонів. Крім ігор, A10.com також може надавати різні соціальні функції по взаємодії між користувачами, такі як можливість обміну результатами гри з друзями або участь у розрахованих на багато користувачів іграх. Це робить сайт привабливим для тих, хто хоче грати та змагатися з іншими.

Перевагами цього вебресурсу є, по-перше, різноманітність ігор. A10.com надає доступ до безлічі різноманітних ігор у різних жанрах. Цей вебсайт має простий та інтуїтивно зрозумілий інтерфейс, список ігор у вигляді плитки знаходиться на всьому екрані користувача, що робить його зручним для пошуку та ігрового процесу. Всі ігри на A10.com безкоштовні. Проте, відразу варто зазначити, що ігри можуть містити внутрішньо-ігрову валюту для пришвидшення ігрового процесу та подолання деяких складних завдань. Але, як і будь-який вебресурс, a10.com не є виключенням та має недоліки. Перший з них це реклама: як і багато інших сайтів з безкоштовними іграми, A10.com може містити вбудовану рекламу. Реклама може відволікати гравців та впливати на загальний

досвід. Також деякі ігри можуть надавати мікротранзакції, що означає, що гравці можуть витратити гроші всередині гри на різні предмети або покращення.

Невеликим недоліком, який легко пояснюється є постійна залежність від Інтернету. Для доступу до ігор на A10.com потрібне підключення до Інтернету, і відсутня можливість грати без доступу до мережі. Також якщо з'єднання до інтернету було перервано, більшість ігор не мають автоматичного збереження, тому, до прикладу, рівень у грі потрібно буде проходити спочатку.

1.1.2 Вебресурс Poki.com

Poki.com - це популярний вебсайт, що надає широкий вибір безкоштовних онлайн-мініігор. Тут можна знайти ігри різних жанрів, включаючи головоломки, аркади, симулятори, гонки та багато інших. Як і більшість вебресурсів, на яких можна провести час за грою, Poki пропонує широкий вибір ігор різних жанрів та категорій. Сайт має інтуїтивно зрозумілий дизайн та просту навігацію. Він також забезпечує можливість спілкування з іншими гравцями та збереження ігрових даних. Схожим до A10.com недоліком є те, що сайт може містити вбудовану рекламу і всі ігри вимагають підключення до Інтернету.

A10.com та Poki.com надають можливість користувачам створити свій персональний акаунт, це дуже добре покращує видачу рекомендованих ігор, а також надає таку можливість, як збереження ігор до списку обраних, а також можливість залишати коментарі та оцінки до кожної гри. Єдиною різницею може бути асортимент ігор та наповненість їх по жанрам, проте у всьому іншому ці два ресурси мають схожі характеристики та цільову аудиторію.

1.1.3 Вебресурс Nitrome.com

Nitrome.com – це ігровий вебресурс, який легко запам'ятати через його незвичайний стиль оформлення та оригінальність та специфічність ігор. Цей ресурс дійсно має унікальний дизайн з характерним стилем графіки. Загалом, цей стиль можна назвати піксель-артом. Nitrome не тільки дає доступ до ігор, але і сам створює їх, ці ігри можна точно назвати незвичними, адже вони мають специфічний геймплей та задумку. На цьому сайті можна отримати досвід, який не вийде отримати на інших. На цьому ресурсі можна знайти безліч різних ігор,

які розподілені по жанрам, це дозволяє з легкістю знайти те, що буде користувачеві до вподоби. Всі ігри є безкоштовними, що робить цей ресурс доступним для широкої аудиторії.

Можна виділити декілька переваг, а саме унікальність стилю та дизайн самих ігор. Далі можна виділити оригінальність ігор, тому що сайт спеціалізується на оригінальних та інноваційних іграх, це дуже добре допомагає залучати нових користувачів, які хотіли б знайти щось незвичне.

Але така незвичність та унікальність може бути і недоліком, адже такий незвичний дизайн не підійде для користувачів з більш утилітарним мисленням і такий вибір дизайнерів може не сподобатись їм. Також, попри велику кількість ігор, вона, все одно, менша ніж на ресурсах-конкурентах.

Цільовою аудиторією є люди, які втомилися від звичних сайтів та прагнуть знайти щось оригінальне та нестандартне, а також для поціновувачів унікальності в дизайні. Цей ресурс може залучати як дітей, так і дорослих різних вікових груп.

Контент на Nitrome.com включає ігри різних жанрів, де кожна гра зазвичай пропонує свій унікальний геймплей та історію. Ігри можуть змінюватись від аркадних пригод до складних головоломок та логічних завдань. Nitrome намагаються постійно оновлювати асортимент ігор, для того щоб слідувати трендам та забажанкам користувачів.

1.1.4 Вебресурс Crazygames.com

Crazygames.com – це вебресурс для онлайн-ігор, який пропонує різноманітні ігри, включаючи звичайні так і більш нестандартні варіанти. Цей сайт надає доступ до великої кількості різних ігор, це будуть ігри різних жанрів для різних вікових груп. Crazygames – це ресурс, який експериментує з форматами ігор, незвичним геймплеєм у них та концептами. Як і конкурентів, доступ до ігор тут повністю безкоштовний, що робить його широкодоступним.

Недоліком на цьому сайті можна назвати якість ігор, які представлені, через велику кількість ігор не завжди вдається слідкувати за якістю та відповідністю стандартам та потребам користувачів. Також на сайті присутня реклама, яка, звісно, негативно впливає на досвід користування вебресурсом, адже вона

з'являється в багатьох місцях на сайті та може бути надокучливою.

Цільова аудиторія Crazy Games включає широкий спектр користувачів. На сайті зібрані ігри різних жанрів та складності проходження, що може залучити різні вікові групи користувачів. Особливо це може залучати тих, хто шукає нестандартні та експериментальні ігри.

Контент на Crazy Games включає різноманітні ігри, починаючи від шутерів та бродилок до стратегій, головоломок і симуляторів. Сайт надає як класичні ігри, які є звичними для кожного, так і, що експериментують з геймплеєм та ідеями. Контент на сайті регулярно оновлюється, додаються нові ігри, видаляються ті, що не користуються попитом. Це дозволяє залучати та утримувати якнайбільше користувачів, які залишаються задоволеними та дають змогу завжди знаходити щось нове для себе.

1.2 Визначення характеристик адаптивності системи мініігор

Існує кілька способів, як адаптивні системи впливають на вибір рекомендацій ігор. Адаптивні системи аналізують історію ігор користувача, уподобання, ігрову поведінку і навіть демографічні дані, щоб створювати персоналізовані рекомендації ігор. Це дозволяє гравцям знаходити ігри, які відповідають їхнім інтересам, що покращує рівень задоволення. Адаптивні системи враховують контекст та ситуацію гравця під час рекомендації ігор. Наприклад, системи можуть пропонувати розваги, які підходять для короткострокового ігрового сеансу на мобільному пристрої або для довгої гри на ігровій консолі. Це враховує обмеження часу та влаштування гравця. Адаптивні системи враховують, які ігри користувач грав чи відхилив у минулому, щоб навчитися краще рекомендувати ігри у майбутньому. Вони можуть використовувати зворотний зв'язок від гравця, щоб уточнити рекомендації та уникати нав'язування ігор, які не цікаві.

Адаптивна система онлайн-мініігор повинна мати деякі важливі ключові характеристики, щоб вона могла бути саме адаптивною, тобто підлаштовувала видачу результатів під кожного користувача персонально. Основним фактором є різноманітність ігор: кожен жанр повинен містити як мінімум декілька ігор. Система повинна мати механізми аналізу дій користувача, а саме ігор, які він оцінював, які запускав та які додавав до обраних, збору інформації про гру, її жанр, а також віку користувача, скільки часу він проводив за грою. Саме це буде впливати на коректну та персоналізовану видачу результату.

Також важливим інструментом аналізу даних є зворотній зв'язок з користувачами, а саме їх відгуки про ігри. Таким чином можливо набагато краще персоналізувати результат. Рекомендації обов'язково мають оновлюватись динамічно на основі нових даних та дій користувача, щоб гарантувати актуальність рекомендацій. Ще варто брати до уваги сезонність та деякі часові фактори, такі як пора року або свята. Завдяки цьому можна більш точно дотримуватись трендів та покращувати вибірку ігор. Для дотримання безпеки варто враховувати вікові обмеження.

Деякими факторами, які також будуть впливати на досвід користувача з сайтом, є адаптивний дизайн. Адже в наш час існує безліч різних девайсів, різного розміру та роздільної здатності моніторів, тому адаптивність в дизайні є одним із ключових факторів. Також важливим можна назвати швидке завантаження, загальну оптимізацію системи та низьку затримку. У більшості випадків це буде залежати тільки від якості інтернет з'єднання користувача системи, проте цьому варто приділити увагу. Попри відносно простий та зрозумілий функціонал, ресурс повинен мати підтримку декількох мов, для того щоб користувачі з різних країн мали змогу комфортно використовувати онлайн-систему.

1.3 Загальний опис системи

Вебресурс онлайн-мініігор повинен мати такі сторінки: головна сторінка, сторінка реєстрації та авторизації, сторінка з грою.

На головній сторінці вебресурс користувач повинен мати можливість переглянути весь список ігор, переглянути список категорій. Також повинна бути можливість знайти гру за допомогою пошуку. При введенні ключового слова та натисканні кнопки повинен з'явитися список з підібраними іграми згідно ключового слова користувача.

На сторінці повинен бути хедер, де розташовується логотип або назва сайту та справа ім'я користувача або кнопка Логін. Коли користувач авторизований він має можливість перейти до відібраних ігор, які йому сподобались та кнопка виходу з акаунту.

Блок з іграми – основний контент сторінки, де вони розташовані за датою додавання.

Повинна бути реалізована можливість сортування ігор: за датою додавання на ресурс, за популярністю, за рейтингом.

Кожен блок з грою повинен мати її назву та зображення. При натисканні на конкретну гру повинен відбуватися перехід до неї.

При переході до конкретної гри повинна завантажитись сторінка де основне місце займатиме блок з грою. Сбоку є блок з інформацією про гру: назва, опис геймплею, можливість оцінити - представлений у виді п'яти зірок та кнопка додавання до обраних ігор користувача.

Під цим блоком знаходиться список ігор, які є схожими до тієї, в яку грає користувач.

Сторінка реєстрації та авторизації матимуть спеціальні форми для введення даних. У першому випадку це логін, ім'я та пароль. У випадку авторизації це поля логіну та паролю.

При реєстрації введені дані користувача повинні записуватися до бази

даних. Якщо користувач з такими даними вже зареєстрований, то показати повідомлення про це. При авторизації проходить перевірка даних на те, чи є вони у базі даних, та в випадку, якщо ні, то показати повідомлення про це.

1.4 Визначення вимог до розроблюваної системи

1.4.1 Загальні вимоги

Користувачу повинен надаватися доступ до списку ігор, категорій, а також можливість перейти до обраної гри. Також є функціонал пошуку гри на ресурсі та переходу до знайденої гри.

Користувач може зареєструватись та додавати ігри до списку вподобань.

На сторінці налаштувань акаунту є можливість змінити ім'я, додати дані про вік та місце проживання.

Можливість сортування ігор за різними параметрами.

На сторінці з грою є функціонал оцінювання гри, переходу до наступної або попередньої, можливість перегляду опису гри.

Користувачами вебресурсу онлайн-мініігор можуть бути будь-які користувачі Інтернет. Цільовою аудиторією є користувачі шкільного віку. Проте ресурс можуть використовувати і студенти, діти та люди більш поважного віку. Для цього не потрібно ніяких технічних знань.

1.4.2 Специфічні вимоги

На сторінці реєстрації система повинна зареєструвати користувача за введеними ним даними та натисканні кнопки реєстрації.

На сторінці авторизації система повинна перевіряти дані з вже існуючими та показувати відповідне повідомлення при успіху та при невдалій спробі авторизуватися.

Список ігор на головній сторінці повинен мати функціонал сортування за датою, за популярністю, та за рейтингом. Тобто список ігор повинен бути

відсортований від найновішої гри, якщо обраний пункт дати. Якщо обрати пункт відсортувати за рейтингом список сортується від гри з найвищим рейтингом до гри з найнижчим.

Пошук повинен здійснюватися за ключовими словами, які повинні бути присвоєні кожній грі, ці слова можуть знаходитися в описі гри. Пошук повинен відфільтрувати список ігор та залишити тільки ті, які підходять.

Категорії працюють схожим чином. При обиранні певної категорії ігри фільтруються згідно ключового слова в назві категорії.

На сторінці з грою при натисканні кнопки, яка відповідає за додавання гри до списку обраних, ця інформація повинна зберігатись, та при наступному вході на вебресурс відображатись.

Жорстких вимог до робочих характеристик немає, вебресурс має працювати швидко майже на всіх персональних комп'ютерах, які мають підключення до інтернету. Проте швидкість інтернету не повинна бути занадто низькою, адже тоді гра довго буде завантажуватись та геймплей стане не зручним та неприємним.

База даних матиме поля з унікальним ключем id користувача. Також поля з даними про ім'я, пошту, та пароль в захищеному вигляді. Дані про список обраних ігор.

1.5 Очікувані переваги від запропонованих підходів

Розвиток інформаційних технологій та поширення доступу до інтернету призвело до широкого використання різних вебресурсів, у тому числі вебресурсів онлайн-мініігор. Безліч користувачів заходять на такі вебсайти для того щоб знайти там розвагу, хобі, навіть чомусь навчитися або просто скоротати час. Виходячи з цього створення вебресурсу онлайн-мініігор є актуальним питанням, яке має широкі перспективи та важливість.

Адаптивна система являє собою інноваційний підхід, який має можливість враховувати вподобання, потреби кожного користувача та персонально під кожного показати відповідний, унікальний результат. Це є особливо актуальним, тому що все більше вікових груп вміє або вчиться користуватись браузером та має доступ до мережі Інтернет.

Отже, можливі вигоди можуть мати, як розробники, так і звичайні користувачі, тому така розробка може мати важливі стратегічні задачі.

Залучення аудиторії є важливою задачею, розробка такої адаптивної системи дозволяє досягти більш широкої аудиторії, задіти все більше вікових категорій. Треба добитися того, що користувачі з різними екранами, роздільною здатністю, різною швидкістю інтернету зможуть вільно та без проблем користуватись вебресурсом. Це безумовно призведе до підвищення трафіку, а це являє собою наслідок – залучення більшої кількості користувачів.

Наступним є збільшення задоволених користувачів. Адаптивний дизайн, зрозумілий інтерфейс, персоналізовані дані ведуть до підвищення якості досвіду користувача вебресурсом. Це напряду веде до збільшення вірогідності, що користувачі будуть повертатися на ресурс знову.

Розробка адаптивної системи несе за собою збір даних про дії користувача на сайті, а саме історія ігор, які він грав, тривалість, оцінки до ігор, коментарі, вік, тимчасові фактори. Чим краще буде відбуватись аналіз цих даних, тим кращий досвід використання сайтом отримає користувач. Правильний аналіз цих даних дозволить покращити систему та оптимізувати видачу коректного результату. Тому покращення аналізу зібраних даних є однією з головних напрямків розвитку адаптивної системи.

Важливим аспектом розвитку вебсистеми є інтеграція з соціальними мережами, можливість запросити друзів до спільного проходження гри або змагань, якщо гра підтримує багатокористувальницький режим. Саме цим платформа сприяє зростанню та розвитку комунікабельності серед людей та залучення більшої кількості користувачів. Це веде до збільшення популярності вебресурсу.

Система надає можливість для монетизації, а саме внутрішньо-ігрові покупки, реклама, платні підписки та інші методи для отримання прибутку. Це все буде сприяти фінансовій стійкості вебресурсу.

Розробка адаптивної системи вимагає часу, праці та ресурсів, проте така розробка вносить вагомий внесок в розвиток онлайн-геймінгу та розважально-розвиваючих вебресурсів.

Розробка адаптивних систем сприяє технологічному прогресу в галузі веброзробки та систем штучного інтелекту. Це сприяє зростанню промисловості та створенню більш досконалих рішень. Ресурси, які були інвестовані у розробку адаптивної системи, виправдовуються покращенням досвіду користувачів і створення більш приємних та доступних ресурсів для доступу до онлайн-ігор. Розвиток онлайн-мініігор сприяє гарному розвитку економіки та вносить свій внесок в це за рахунок монетизації та реклами.

Розробка адаптивної системи мініігор у вигляді вебресурсу приносить значні вигоди як розробникам, так і користувачам. Це сприяє залученню та задоволенню аудиторії, монетизації, розвитку технологій та поліпшенню користувацького досвіду. Інвестиції в розробку адаптивних систем є виправданими з огляду на їх внесок у розвиток онлайн-геймінгу та розважальних вебресурсів.

Визначення областей подальших досліджень у цьому напрямку може бути наступним. Розвиток адаптивних систем для мініігор у вигляді вебресурсів є великою і важливою областю досліджень. Можна виділити кілька ключових областей, які можуть стати об'єктом подальших досліджень у цьому напрямку.

Найголовнішим напрямком є покращення алгоритмів рекомендаційних систем, тому що вони відіграють важливу роль в адаптивних системах. Подальші вивчення будуть концентруватися на вдосконаленні алгоритмів, які будуть враховувати більш комплексні фактори, наприклад, емоційний стан гравця, контекст.

Перспективним напрямком є інтеграція штучного інтелекту як до системи для покращення підбору ігор так і в самі ігри. Дослідження в цій галузі можуть

стосуватися створення віртуальних агентів та розумних сценаріїв, які адаптуються до гравців.

Так, як система буде збирати все більше і більше даних про користувача варто покращувати системи безпеки та конфіденційності даних. Варто дослідити методи для забезпечення безпеки зберігання та передачі персональних даних користувача.

Важливим напрямком є дослідження зібраної інформації, наприклад, врахування географічних особливостей, кліматичних та культурних, це в свою чергу може посприяти покращенню видачі результату для користувача та слідування трендам. Також не варто забувати про дослідження психологічної та соціальної складової, це може торкатися питань впливу ігор на психологічно-емоційний стан людини та її поведінку під час гри та після.

Розробка адаптивних систем мініігор у вигляді вебресурсів надає широкий спектр перспективних областей досліджень. Уважне вивчення та розвиток цих областей може зробити розробку та використання онлайн-ігор ще більш привабливими та інноваційними, сприяючи поліпшенню користувацького досвіду та розвитку цієї індустрії.

2 ПРОЄКТУВАННЯ СИСТЕМИ

2.1 Проєктування архітектури програмного забезпечення

Розробкою архітектури програмного забезпечення називають процес створення структури програми або обчислювальної системи, що включає в себе програмні компоненти, видимі зовні властивості цих компонентів, а також відносини між ними. Такий термін також використовується для документування архітектури програмного забезпечення. Документування архітектури ПЗ робить набагато легшим процес комунікації між зацікавленими особами, дозволяє зафіксувати прийняті на ранніх етапах проєктування рішення про

високорівневий дизайн системи і дозволяє використовувати компоненти цього дизайну і шаблони повторно в інших проєктах.

Від принципу організації компонентів всередині системи залежить ціна на підтримку і розробку нових фіч, трудовитрати на побудову цілої системи з використанням даної архітектури. Від архітектури насамперед буде залежати один з важливих параметрів розробки - собівартість. А також ще і можливість використання коду повторно, а разом з ним і зменшення трудовитрат на кожну подальшу розробку. Процес проєктування повинен супроводжуватись створенням діаграми розміщення.

Діаграма розміщення типового вебсайту зображена на рис. 2.1.

На діаграмі зображено декілька об'єктів. Робоча станція з браузером, далі вебсервер, який обробляє дані та база даних з даними. Користувач робить запит на додавання даних або зчитування їх. Сервер отримує запит від користувача та передає отримані дані від нього та записує їх до таблиці бази даних. При запиті на отримання даних сервер формує запит на отримання даних з бази даних і далі передає їх, де вони виводяться на сторінці.

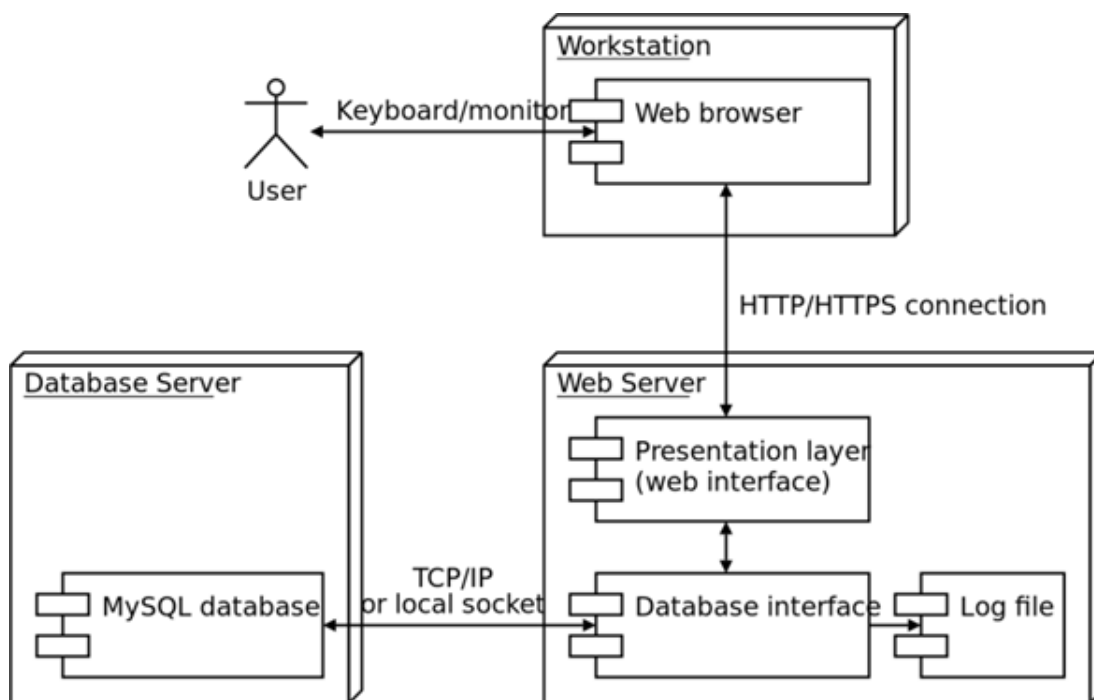


Рисунок 2.1 – Діаграма розміщення вебсайту

Опис вебресурсу онлайн мініігор повинен мати більш детальний опис, того, що може відбуватись на сайті та з чим може взаємодіяти відвідувач. Користувачі можуть зареєструватися, якщо акаунт ще не створений. Далі потрібно авторизуватися за даними, які вводилися при реєстрації. Діаграма розміщення зображена на рис. 2.2. На головній сторінці сайту користувач має декілька варіантів взаємодії. Один з яких, переглянути список ігор та перейти на сторінку з нею. Відвідувач сайту потрапляє на сторінку з грою, де може прочитати правила гри, пограти, оцінити гру та додати її до списку вподобань. Також може перейти до наступної гри по списку та попередньої. На головній сторінці користувач може почати пошук гри за ключовим словом, переглянути список категорій та відфільтрувати по одній з них, відсортувати за датою, або рейтингом, або популярністю.

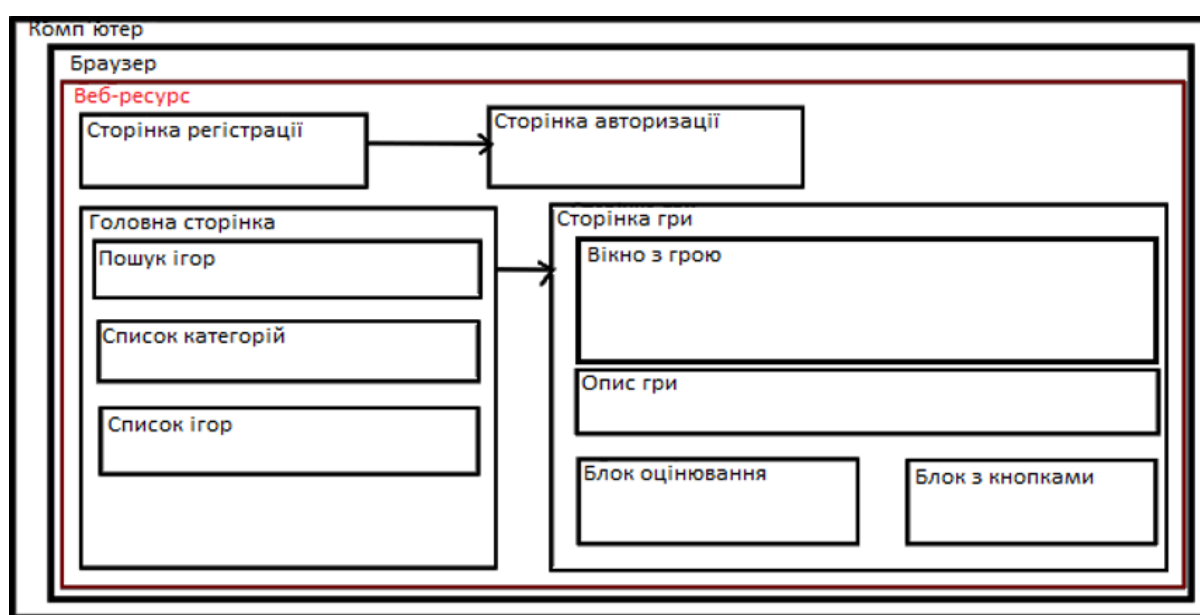


Рисунок 2.2 – Діаграма розміщення

2.2 Проєктування інтерфейсу користувача

Сукупність засобів, якими користувач взаємодіє з різними пристроями, представляє собою форму інтерфейсів, де однією стороною є людина, а іншою - машина (пристрій або програмне забезпечення). Зазвичай термін "інтерфейс користувача" (ІК) сприймається як зовнішній вигляд програмного забезпечення (ПЗ), проте це визначення є надто обмеженим, оскільки через інтерфейс саме користувач сприймає програму в цілому та використовує її функціонал.

ІК визначає не лише зовнішній вигляд ПЗ, але й забезпечує підтримку прийняття рішень в конкретній предметній галузі, а також визначає порядок використання ПЗ і призначену для нього документацію. Фактично ІК об'єднує всі складові ПЗ, які впливають на взаємодію користувача з програмним забезпеченням. Ці складові включають набір задач, які користувач вирішує за допомогою ПЗ, метафору, використовувану програмним забезпеченням (наприклад, "робочий стіл" у операційній системі Windows), елементи управління, навігацію між блоками програмного забезпечення, візуальний та інші аспекти дизайну вікон та екранних форм програми. Стиль інтерфейсу користувача - це набір ознак, методів, прийомів діяльності, які характеризують індивідуальність інтерфейсу користувача, а також сукупність прийомів використання інструментів розроблення ПЗ.

Складність ІК обумовлюється рядом невизначеностей, які суттєво впливають на процес розроблення. Нелінійність проєктування ІК полягає у відсутності фіксованого, впорядкованого і прямолінійного алгоритму від початку до кінця проєктування. Процес проєктування є невизначеним, оскільки не існує рівняння, за яким можна було б одержати однаковий результат при заданих однакових початкових умовах, більш того, одержати ідентичний результат практично неможливо. Інтерфейс користувача неортогональний у тому сенсі, що будь-який аспект проєктного рішення може впливати на інші аспекти, до того ж результат цього впливу не завжди є позитивним та прийнятним.

Процес складається з 3 основних етапів:

- збір та аналіз інформації від користувачів;
- розроблення та реалізація ІК;
- підтвердження якості створеного ІК.

Даний алгоритм може використовуватись як при розробці об'єктно-орієнтованого інтерфейса користувача, так і при проектуванні традиційних проблемно-орієнтованих інтерфейсів. Цей процес не залежить від матеріальної та програмної платформ, ОС, застосовуваного інструментарію.

Розпочинати розробку будь-якої системи слід із користувачів. Перед тим, як приступити до створення системи, необхідно визначити, які проблеми користувачі хочуть вирішити та як вони звикли взаємодіяти. Важливо активно спостерігати за користувачами і розпитувати їх, враховуючи обмеження їхніх комп'ютерних систем на технічному та програмному рівні. Потрібно завжди пам'ятати, що рішення має відповідати не лише поточним, але й майбутнім потребам користувачів.

На етапі аналізу інформації від користувачів існують ключові питання, які слід враховувати. Проектування та постановка питань, а також аналіз, вимагають специфічних навичок. Під час опитування користувачів та аналізу їхніх відповідей важливо вчиняти уважно та обережно.

Перший етап, що включає збір та аналіз інформації, може бути поділений на п'ять кроків: визначення профілю користувача, аналіз задач користувача, збір вимог, аналіз робочого середовища користувачів та відповідність вимог задачам. Цей етап є ключовим для розуміння потреб користувачів та створення високоякісного продукту.

Визначення профілю користувача передбачає визначення характеристик користувачів, таких як вік, освіта та їхні вподобання, а також отримання інших необхідних відомостей. Аналіз задач користувача спрямований на виявлення конкретних завдань, які ставляться перед користувачами, і способів їх вирішення. Збір вимог, наданих користувачами, полягає в розумінні користувачами очікувань від продукту чи інтерфейсу та їхньої участі в його вирішенні завдань. Аналіз

робочого середовища користувачів враховує всі аспекти, такі як освітлення, шум, комфорт робочого місця, фізичний стан користувачів та їхня мобільність. Завершальний етап - відповідність вимог користувачів задачам - включає перевірку, чи відповідають вимоги користувачів реальності та завданням, які вони повинні вирішити, для оптимізації рішень та пропозиції найкращих варіантів для користувачів.

Другий етап включає в себе розробку та впровадження інтерфейсу користувача (ІК). Розробка ІК для програмного продукту, звісно, потребує значних затрат часу та ресурсів. Цей етап складається з ряду кроків, які повинні бути виконані у визначеній послідовності. Хоча існує спокуса розпочати програмування фінальної версії продукту негайно, не звертаючи уваги на розробку інтерфейсу, необхідно пройти всі етапи процесу розробки перед переходом до програмування.

Розробка ІК включає такі етапи:

Визначення цілей з точки зору зручності використання продукту: На ранніх етапах розробки потрібно чітко визначити, що продукт може пропонувати користувачам. Ці цілі найкраще формулювати у термінах дій користувача, охоплюючи області придатності, ефективності, легкості освоєння та оцінки якості продукту.

Розробка задач і сценаріїв дій користувачів: Сценарії - це опис дій, які виконує користувач, представляючи послідовність завдань, що постають перед ним. Важливо розробити кілька сценаріїв користувача, щоб врахувати різні аспекти взаємодії з інтерфейсом.

Визначення цілей та операцій інтерфейсу: Це ключовий крок, який включає в себе виділення об'єктів, даних і дій з сценаріїв, уточнення списку об'єктів і дій, а також створення діаграм взаємодії між об'єктами.

Визначення іконок об'єктів та візуального представлення: Визначення того, як найкраще візуалізувати об'єкти на екрані та представити інформацію, яку вони містять.

Розробка меню об'єктів і вікон: Встановлення способу взаємодії

користувачів з розробленими об'єктами та вікнами.

Оптимізація візуальної розробки: Вдосконалення процесу візуального проектування.

При переході до ітераційного процесу розробки рекомендується надавати перевагу створенню прототипу перед побудовою інтерфейсу. Прототипування є цінним методом для швидкого візуалізування альтернативних варіантів розробки, зокрема на ранніх етапах тестування зручності використання. Слід дотримуватися трьох правил при використанні прототипів:

- прототипувати на ранніх етапах ітеративного процесу розробки;
- створювати різні альтернативні варіанти;
- бути готовим відкинути код прототипу.

Третім етапом є підтвердження якості інформаційної системи (ІК), а ключовим аспектом цього ітеративного процесу розробки є тестування на зручність використання. Суть цього етапу полягає в тому, щоб випробувати продукт на великій кількості користувачів і дослідити, наскільки легко і зручно вони можуть з ним взаємодіяти. Мета тестування на зручність - оцінка та аналіз поведінки, дій і рівня задоволеності користувачів. Багато розробників включають це тестування в кінцеві етапи розробки, хоча це може бути вже пізно для внесення істотних змін на основі його результатів. Важливо, щоб розробники були присутні під час тестування, щоб вони могли вивчати, як саме користувачі використовують продукт. Проте їм не слід забезпечувати технічну підтримку користувачів під час тестування.

У минулому розвиток інформаційних систем обмежувався системно- або технологічно-орієнтованою розробкою. Цей підхід передбачав поліпшення технологій та систем, на основі яких розроблялося програмне забезпечення. Однак він не враховував зворотний зв'язок та побажання користувачів, обмежуючи їх програмними функціями, які розробники могли реалізувати. Однією з головних причин виникнення невдалих програм є недостатнє взяття користувачів на борт проєкту. Також важливо враховувати активну участь користувачів, їх знання про фактичних користувачів продукту та середовище

використання розробленого програмного забезпечення. Детальна інформація про користувачів та їх середовище допомагає встановити рамки для проєктування ІК та гарантує його практичність. Крім того, вона допомагає розробникам визначити особливості продукту, які важливі для користувачів, і вибрати відповідні методи розроблення та підходи до створення єдиної стилістики додатків.

Починаючи з 80-х років, в розробці програмного забезпечення звертали увагу на користувача, залучаючи їх до процесу створення. Проте їх роль була пасивною, вони надавали вимоги до ПЗ і вказували, які завдання вони хочуть вирішувати. Сучасні розробники використовують методології "розроблення із залученням користувачів" та "розроблення, орієнтоване на користувачів, що навчаються". У цьому підході користувачі розглядаються як активні учасники розробки, що сприяє покращенню доступності інтерфейсу та програмного забезпечення, а також гарантує, що отримане ПЗ відповідає їх вимогам та очікуванням. "Розроблення, орієнтоване на користувачів, що навчаються", спрямоване на те, щоб користувач при вирішенні завдань отримував нові навички роботи з програмним забезпеченням, розвивав інтелектуальні здібності та отримував знання в різних областях.

Розроблення, орієнтоване на користувача, базується на наступних керівних принципах:

- розуміння потреб користувачів є рушійною силою усього проєкту;
- все, що користувачі бачать і до чого вони мають доступ, повинно проєктуватись сумісними зусиллями;
- інноваційний проєкт завжди є результатом інтенсивної роботи команди спеціалістів з різних галузей;
- рішення щодо ІК повинні базуватись на зворотньому зв'язку з користувачами;
- результати зворотнього зв'язку з користувачами повинні збиратись з заданою точністю, швидкістю та частотою;
- зворотній зв'язок здійснюється як з потенційними, так і з фактичними

користувачами;

- розроблення, орієнтоване на користувача, повинне стандартизуватись і впроваджуватись;
- розроблення, орієнтоване на користувача, повинне постійно вдосконалюватись.

Макет інтерфейсу головної сторінки вебресурсу зображений на рис. 2.3.

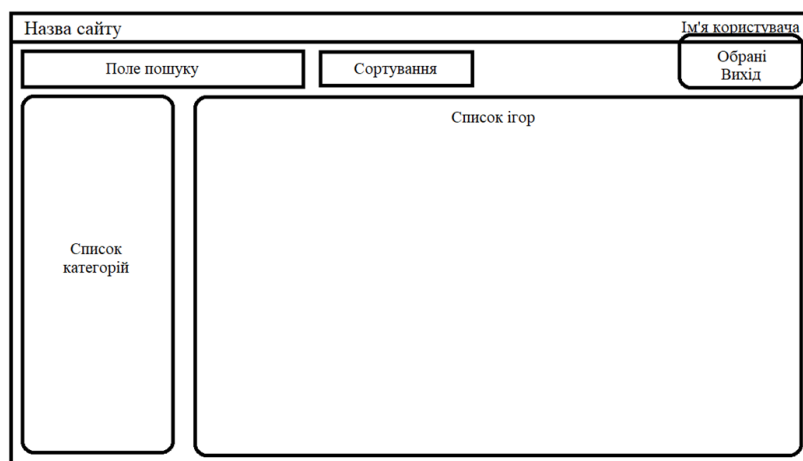


Рисунок 2.3 – Макет інтерфейсу головної сторінки

Макет інтерфейсу сторінки з грою зображений на рис. 2.4.

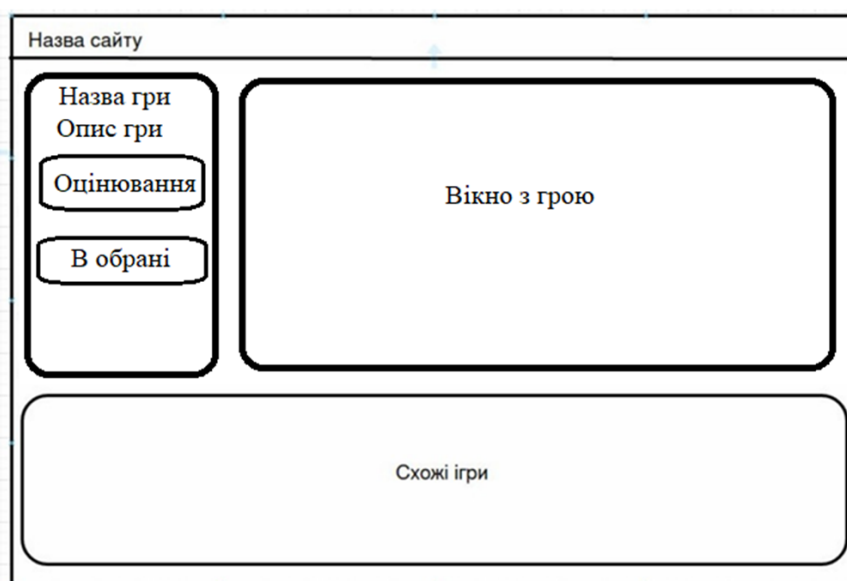


Рисунок 2.4 – Макет інтерфейсу сторінки з грою

2.3 Вибір технологій для створення проєкту

У цьому пункті буде розглянуто та описано основні технології, які були використані при розробці даного проєкту.

2.3.1 Бібліотека React.js

Для написання цього проєкту використано одну з найпопулярніших бібліотек – React JS.

React - це JavaScript-бібліотека, яка використовується для створення декларативних, ефективних та гнучких інтерфейсів користувача. Основна концепція React полягає в компонуванні складних інтерфейсів з використанням невеликих окремих частин коду, відомих як "компоненти". Ця бібліотека дозволяє розробникам створювати великі веб-застосунки, які можуть динамічно оновлюватися без перезавантаження сторінки.

Мета React - бути швидким, простим у використанні та масштабованим. Ця бібліотека працює з користувацьким інтерфейсом в межах застосунків, дотримуючись парадигми модель-вид-контролер (MVC). React можна використовувати окремо або у поєднанні з іншими JavaScript-бібліотеками та фреймворками, такими як Redux чи AngularJS.

React використовує віртуальний DOM для оптимізації оновлення сторінки. Замість прямого взаємодії з DOM браузера, бібліотека порівнює зміни в віртуальному DOM з попередньою версією, визначаючи оптимальний спосіб оновлення DOM браузера.

JSX є розширенням JavaScript, що дозволяє описувати структуру користувацького інтерфейсу у вигляді XML-подібного коду. Компілятор перетворює код JSX у стандартний JavaScript код, який викликає методи React.

React також надає методи життєвого циклу, які дозволяють розробникам обробляти різні аспекти життєвого циклу компонентів, такі як підготовка до оновлення, вставка у DOM та після рендерингу.

— `componentWillMount`: Викликається перед вставкою компонента у віртуальний DOM. Використовується для підготовки та ініціалізації даних перед монтажем.

— `componentDidMount`: Викликається після успішного вставлення компонента у віртуальний DOM. Використовується для виконання дій, які потрібно виконати після монтажу, наприклад, завантаження даних чи ініціалізація бібліотек.

— `shouldComponentUpdate`: Викликається перед оновленням компонента, дозволяючи вирішити, чи потрібно виконувати оновлення. Повертає `true`, якщо оновлення необхідне, та `false` в іншому випадку.

— `componentWillUpdate`: Викликається перед оновленням компонента, якщо `shouldComponentUpdate` повернув `true`. Використовується для підготовки до оновлення.

— `componentDidUpdate`: Викликається після успішного оновлення компонента та його віртуального DOM. Використовується для виконання дій після оновлення.

У React.js застосунок будується з використанням компонентів. Компонент - це незалежний блок інтерфейсу, який можна перевикористати. Ці блоки можуть включати в себе розмітку та логіку. Компоненти можуть бути вкладені один в одного. Це дозволяє будувати складні та комплексні ієрархії користувацького інтерфейсу. Компоненти можуть бути функціональні та класові, але зараз класові компоненти є застарівшими та перевагу надають саме функціональним.

У React.js існує два важливих поняття - стан та властивості (`state` та `props`). Це два основних способи зберігання даних у компонентах. Властивості передаються ззовні, наприклад, з батьківського компонента, основною метою є передача даних або функцій. State використовується для збереження та роботи з внутрішніми даними компонента, які можуть змінюватись під час роботи застосунку.

Структура проєкту може бути різною та змінюватись в залежності від його розміру та складності. Прикладом однієї з можливих структур React.js

застосунку відображений на рис. 2.5.

```

my-react-app/
├─ node_modules/
├─ public/
│   └─ index.html
│   └─ favicon.ico
├─ src/
│   └─ components/
│       └─ Component1.js
│       └─ Component2.js
│   └─ App.js
│   └─ index.js
├─ package.json
├─ package-lock.json
└─ README.md

```

Рисунок 2.5 - Базова структура React.js застосунку

Директорія `node_modules` містить всі залежності проєкту, що були встановлені, ця папка створюється автоматично.

У директорії `public` зберігаються статичні файли, наприклад, `index.html`, іконка сайту - `favicon.ico`, а також ті ресурси, які повинні бути доступні з кореня проєкту. Основним файлом, в якому зберігається react-застосунок є `index.html`.

Директорія `src`: Вона містить весь код react-застосунку. У під-директорії `components` знаходяться компоненти проєкту. Головним компонентом додатку є `App.js`. Він визначає структуру всього застосунку.

Файли `package.json` та `package-lock.json` містять інформацію про залежності проєкта. Також перший файл може зберігати скрипти для запуску команд, збірки проєкту та розгортання.

`README.md`: Документація та опис проєкту. Цей файл може містити інструкції зі встановлення, налаштування та використання вашого додатка.

Все що було описане вище є базовою структурою проєкта для React-застосунку. Більш складні проєкти будуть мати додаткові директорії та

піддиректорії для зберігання якихось інших видів ресурсів, стилів, тестів. Це все використовується для покращення організації проєкту та його масштабування.

Обробка подій є важливою частиною розробки вебдодатку. Бібліотека React.js надає зручні та ефективні методи управління цими подіями. Обробка подій у React відбувається за участі jsx та самого javascript. React має атрибути, які дозволяють зв'язати обробники подій з елементами віртуального DOM. В той час, коли подія трапляється, визивається потрібний обробник події, який має можливість оновити стан компонента або виконати якісь-небудь інші дії.

Для зв'язування подій з методами React компонентів треба використовувати синтаксис JSX, який дозволяє визначити обробники для подій у самих компонентах.

Лістинг 2.1 - Лічильник

```
import React, { useState } from 'react';

const CounterComponent = () => {
  const [count, setCount] = useState(0);

  const handleClick = () => {
    setCount(count + 1);
  }

  return (
    <div>
      <p>Лічильник: {count}</p>
      <button                    onClick={handleClick}>Збільшити
лічильник</button>
    </div>
  );
}

export default CounterComponent;
```

У цьому прикладі onClick - це атрибут, що викликає обробник подій handleClick з кнопки. Коли користувач натискає на кнопку, то викликається метод handleClick. Цей метод змінює стан компонента, що призводить до перендеру компонента.

При підході використання зв'язування подій з методами компонентів існує багато переваг. Однією з яких є чистота та читабельність коду: код стає більш

організованим та зрозумілим, оскільки такі обробники написані всередині компонента. Також перевагою є легкість у передачі параметрів. Можливо легко передавати параметри в обробники подій, якщо це є необхідним. Розробник може з легкістю керувати станом компонента, тому що візуальною зміною в інтерфейсі займається сам React.

Зв'язування подій з методами компонентів у React є основною частиною створення інтерактивних вебдодатків.

Маршрутизація - дає змогу користувачам навігуватися різними сторінками та надає структуру застосунку. React, як популярна бібліотека для розробки інтерактивних користувацьких інтерфейсів, надає зручний інструмент для реалізації маршрутизації в додатках - React Router. У цьому рефераті ми розглянемо, як реалізувати маршрутизацію з використанням React Router.

Одним з інструментів маршрутизації в додатках на React.js є React Router. Це бібліотека, яка надає безліч засобів для контролю та керуванням маршрутизацією у React-додатках. Вона дозволяє визначати маршрути та пов'язувати їх з компонентами, які мають відображатися при переході користувача за цією url-адресою.

Лістинг 2.2 - React router

```
import React from 'react';
import { BrowserRouter, Route, Link, Switch } from 'react-router-dom';

const Home = () => {
  return <h2>Домашня сторінка</h2>;
}
const About = () => {
  return <h2>Про нас</h2>;
}

const App = () => {
  return (
    <BrowserRouter>
      <div>
        <ul>
          <li>
            <Link to="/">Додому</Link>
          </li>
        </ul>
      </div>
    </BrowserRouter>
  );
}
```

```

        <li>
          <Link to="/about">Про нас</Link>
        </li>
      </ul>
      <Switch>
        <Route path="/about" component={About} />
        <Route path="/" component={Home} />
      </Switch>
    </div>
  </BrowserRouter>
);
}

export default App;

```

У приведеному прикладі визначаються декілька маршрутів за допомогою компонента `Route` - це компонент, що визначає який компонент відображати при певній url-адресі. Компонент `Link` використовується як навігаційний для переходу між сторінками. `Switch` використовується для відображення лише одного певного компонента `Route`, який підходить за url-адресою.

Загалом `React Router` надає зручні засоби маршрутизації для `react`-додатків. Це дозволяє будувати комплексні та складні навігаційні структури.

`React.js` - це потужна бібліотека `JavaScript`, розроблена `Facebook`, яка значно змінила підхід до створення інтерактивних інтерфейсів для вебдодатків. `React.js` надає розробникам безліч інструментів і практик для створення сучасних і продуктивних вебдодатків. Він став однією з найпопулярніших бібліотек для фронтенд-розробки, і його ефективність і велика спільнота розробників продовжують сприяти його успіху. Оволодіння основами `React.js` є важливою навичкою для створення сучасних вебдодатків.

2.3.2 База даних `Firebase Storage`

Також у розробленому проєкті в якості бази даних був використаний `Firebase Storage`.

`Firebase` – це платформа для розробки мобільних додатків з величезним функціоналом. Починалася вона як стартап, а сьогодні її використовують при розробці кращих кросплатформених додатків. `Firebase` пропонує розробникам безліч інструментів та сервісів, які допомагають спростити та прискорити

розробку створення додатків. Firebase є доволі популярним вибором серед користувачів через простоту використання та інтеграції.

Firebase надає зручні засоби для аутентифікації користувачів у додатках, є такі способи аутентифікації як, за допомогою соціальних мереж, електронна адреса.

Firebase Realtime Database - це сервіс, який дозволяє у реальному часі зберігати дані між користувачами додатку. Він є одним з ключовим зі списку сервісів, які пропонуються. Цей сервіс є NoSQL, тому тут використовуються не реляційні бази даних, це дозволяє створювати динамічні додатки. Робота виконується з даними в JSON форматі, використовується зрозумілий та простий API для отримання даних та їх запису. Однією з важливих особливостей є можливість працювати у офлайн режимі, тому дані будуть синхронізовані після повторного підключення до мережі інтернет, додаток просто продовжить свою роботу.

Firebase Hosting є також дуже популярним сервісом, який надає можливість розміщати статичні вебсайти та вебдодатки використовуючи хмару Firebase.

Firebase Cloud Firestore - це сервіс баз даних, який дозволяє гнучко та масштабовано працювати та керувати даними додатків. Firestore використовує колекції та документи для зберігання даних. Документ містить поля та значення, які організовуються всередині колекцій.

Firestore має суттєві переваги, такі як гнучкість, робота в реальному часі, масштабування. За допомогою цього сервісу можливо легко структурувати дані та зробити їх організацію більш зрозумілою. Firestore підтримує синхронізацію даних в реальному часі між користувачами, це дозволяє відразу бачити зміни. Firestore легко масштабувати в залежності від об'єму та складності додатку, це дає більш високу доступність, яка в свою чергу надає більш швидкий доступ до даних.

Але також Firestore має і недоліки, такі як ціна, що може бути значно дорожче у порівнянні з конкурентами, але, також існує і безкоштовний план, який може оформити будь-хто. Також існують деякі обмеження по об'єму даних та

кількості запитів, використовуючи такий план.

Firebase Authentication - це сервіс, який надає можливість з легкістю додавати можливість користувачам реєструватися та авторизуватися на сайті. Зареєструватися можна використовуючи електронну пошту та пароль, соціальні мережі, номер телефона. Firebase Authentication підтримує інструменти для верифікації номера а також використання на таких платформах як, Android, IOS та вебдодатки. Розробники можуть керувати користувачами, змінювати та встановлювати для них доступ та слідкувати їх активність.

За допомогою цього сервісу можливе додавання безпечної та, саме головне, зручної аутентифікації. Він надає гнучку та масштабовану базу даних для додатків, а також зручний спосіб для реєстрації та авторизації. Він відіграє дуже важливу роль у створенні веб та мобільних додатків.

Одним з важливих аспектів, який має бути досягнутий максимально це є безпека даних. Firebase надає безліч інструментів для забезпечення безпеки у проєктах.

Authentication має можливість ідентифікувати користувачів та керувати їх доступом до ресурсів додатку. Рекомендовано використовувати складні методи аутентифікації, наприклад, по пошті та паролю або двофакторну аутентифікацію для посилення безпеки акаунтів. Також бажано обмежити доступ не авторизованим користувачам до ресурсів додатку. Для реалізації такого підходу використовуються механізми, які є вбудованими, до прикладу, рольова модель. Тут можна призначити роль для користувача та налаштувати доступ для цієї моделі. Дуже рекомендується перевіряти на автентичність користувачів перед наданням доступу до важливих даних. Також треба виділити ролі, які можуть змінювати дані, додавати щось нове та видаляти якісь дані.

Firebase легко інтегрується з React.js. Наприклад ініціалізація приведена у лістингу 2.3.

Лістинг 2.3 - Ініціалізація Firebase

```
import React, { Component } from 'react';
```

```
import firebase from 'firebase/app';
import 'firebase/auth';
import 'firebase/firestore';

const firebaseConfig = {
  apiKey: 'YOUR_API_KEY',
  authDomain: 'YOUR_AUTH_DOMAIN',
  projectId: 'YOUR_PROJECT_ID',
};

firebase.initializeApp(firebaseConfig);

const App = () => {
  // Код додатку
}
```

Треба зауважити, що змінні `apiKey`, `authDomain`, `projectId` повинні бути змінені згідно тих, що були згенеровані персонально для вас.

3 РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Реєстрація та авторизація користувачів

Першим кроком було вирішено розробити реєстрацію та авторизацію користувачів, тому що це є важливим інструментом для подальшої взаємодії з сайтом. Для цього був обраний Firebase, тому що він має зрозумілі та зручні методи для реалізації повноцінної реєстрації та подальшого логіну користувачів.

Лістинг 3.1 - Створення провайдеру

```
export const AuthContext = createContext(undefined)
const AuthProvider = ({children}) => {
  const [currentUser, setCurrentUser] = useState(null)
  const [pending, setPending] = useState(true);

  useEffect(() => {
    app.auth().onAuthStateChanged((user) => {
      setCurrentUser(user)
      setPending(false)
    });
  }, []);
```

```

    if(pending) {
      return <>Loading...</>
    }

    return (
      <AuthContext.Provider value={currentUser}>
        {children}
      </AuthContext.Provider>
    );
  };
};

```

Спочатку ініціалізується константа `AuthContext` для того, щоб використовувати контекст в будь-якому компоненті проєкту. В `AuthProvider` буде загорнутий основний компонент додатку, в розробленому проєкті, компонент для роутингу, що відображений на лістингу 2. Все це робиться для того, щоб можна було використати змінну `currentUser` в будь-якому компоненті отримавши значення з нього за допомогою `useContext`.

Лістинг 3.2 - Роутинг

```

<Router>
  <div className="App">
    <Routes>
      <Route exact path="/" element={<HomePage/>}/>
      <Route exact path="/game/:gameId"
element={<GamePage/>}/>
      <Route exact path="/login" element={<Login />}/>
      <Route exact path="/signup" element={<SignUp />}/>
      <Route exact path="*" element={<NotFoundPage/>}/>
    </Routes>
  </div>
</Router>

```

Роутинг, а саме використання бібліотеки `react-router-dom` потрібно для того, щоб сторінка не перезавантажувалась при зміні `url`. Буде використовуватись декілька роутів:

- `HomePage` - це домашня сторінка, де будуть показуватись всі ігри, які є з предвстановленим фільтром для цього, на цю сторінку будуть попадати користувачі при заході на вебресурс;

- `GamePage` - це сторінка гри, відмінність від інших роутів, в тому, що він є динамічним, через змінюваний параметр `gameId`, за допомогою якого і буде зрозуміло, яка гра відкрита:

- `Login` та `SignUp` - сторінки для авторизації та реєстрації відповідно;
- `NotFoundPage` - обов'язкова сторінка для того випадку, коли користувач буде намагатись змінити url самостійно та попаде на сторінку, якої не існує або для обробки якихось помилок при отриманні даних.

Продовжуючи тему авторизації на сайті, потрібно розуміти, що користувач може ввести некоректні дані, або помилитися в написанні пошти або паролю, про це треба сповістити, показуючи відповідне повідомлення. Для цього помилку, яку повертає сервер треба обробити, та показати відповідне повідомлення з помилкою, але щоб користувач зрозумів, що не так, треба щоб повідомлення містило зрозуміле пояснення.

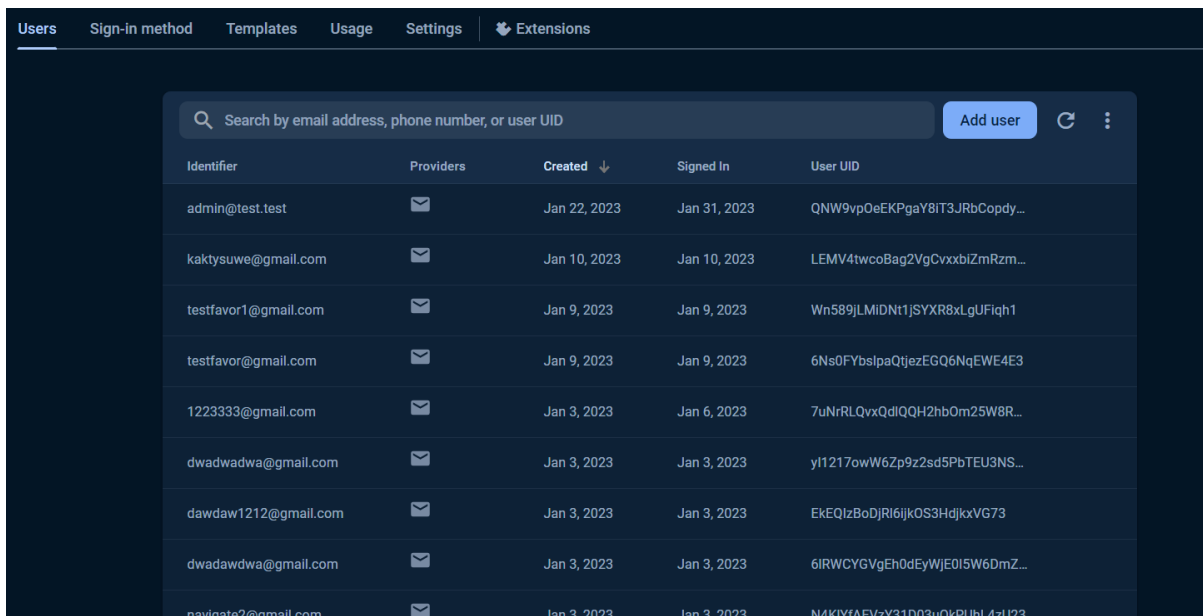
Лістинг 3.3 - Обробка помилки

```
if (e.code === 'auth/user-not-found')
  text = 'There is no user with this credentials'
else if (e.code === 'auth/invalid-email')
  text = 'Enter correct email address'
else if (e.code === 'auth/invalid-password')
  text = 'Enter correct password, it should be larger than 6
symbols'

setAlertState({
  ...alertState, show: true, text: text, type: 'error'
})
```

Подібним чином помилки мають бути оброблені і при реєстрації користувача.

Список користувачів, які зареєструвалися на сайті зберігається у `Firebase` у вкладці `Authentication`, що зображена на рис. 3.1. Тут зберігається лише деяка інформація про користувача, а саме: спосіб аутентифікації, пошта, дата, коли був створений акаунт та коли останній раз був авторизований, а також надважливе поле - унікальний ідентифікатор користувача.



Identifier	Providers	Created	Signed In	User UID
admin@test.test	✉	Jan 22, 2023	Jan 31, 2023	QNW9vpOeEKPgaY8IT3JRbCody...
kaktysuwe@gmail.com	✉	Jan 10, 2023	Jan 10, 2023	LEMV4twcoBag2VgCvxxblZmRzm...
testfavor1@gmail.com	✉	Jan 9, 2023	Jan 9, 2023	Wn589JLMIDNt1JSYXR8xLgUFiqh1
testfavor@gmail.com	✉	Jan 9, 2023	Jan 9, 2023	6Ns0FYbslpaQtjezEGQ6NqEWE4E3
1223333@gmail.com	✉	Jan 3, 2023	Jan 6, 2023	7uNrRLQvxQdlQQH2hbOm25W8R...
dwardadwa@gmail.com	✉	Jan 3, 2023	Jan 3, 2023	yl1217owW6Zp9z2sd5PbTEU3NS...
dawdaw1212@gmail.com	✉	Jan 3, 2023	Jan 3, 2023	EkEQIzBoDjRl6ljkOS3HdjxVG73
dwardadwa@gmail.com	✉	Jan 3, 2023	Jan 3, 2023	6lRWCYGVgEh0dEyWJE0l5W6DmZ...
navigate2@gmail.com	✉	Jan 3, 2023	Jan 3, 2023	N4KIYFAEVzY31D03uQkPUhL4zU23

Рисунок 3.1 - Список користувачів

Також тут можливо створити нового користувача. Ще на інших вкладках можливо переглянути метод для авторизації та за потреби змінити його, але варто зауважити, що при безкоштовному використанні доступний тільки один метод - авторизація по електронній пошті та пароллю (рис. 3.2).

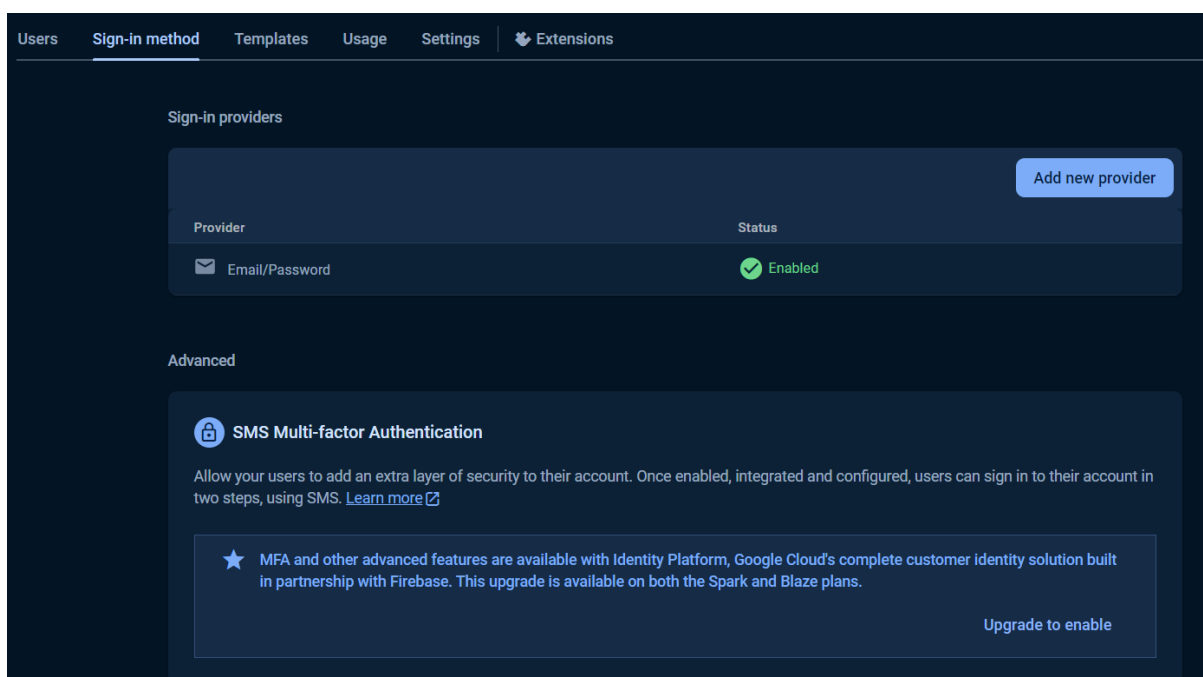


Рисунок 3.2 - Метод аутентифікації

Безумовно корисною функцією є моніторинг використання авторизації, тобто скільки користувачів реєструвалось за певний проміжок часу. Побачити графік можна перейшовши на вкладку Usage (рис. 3.3).

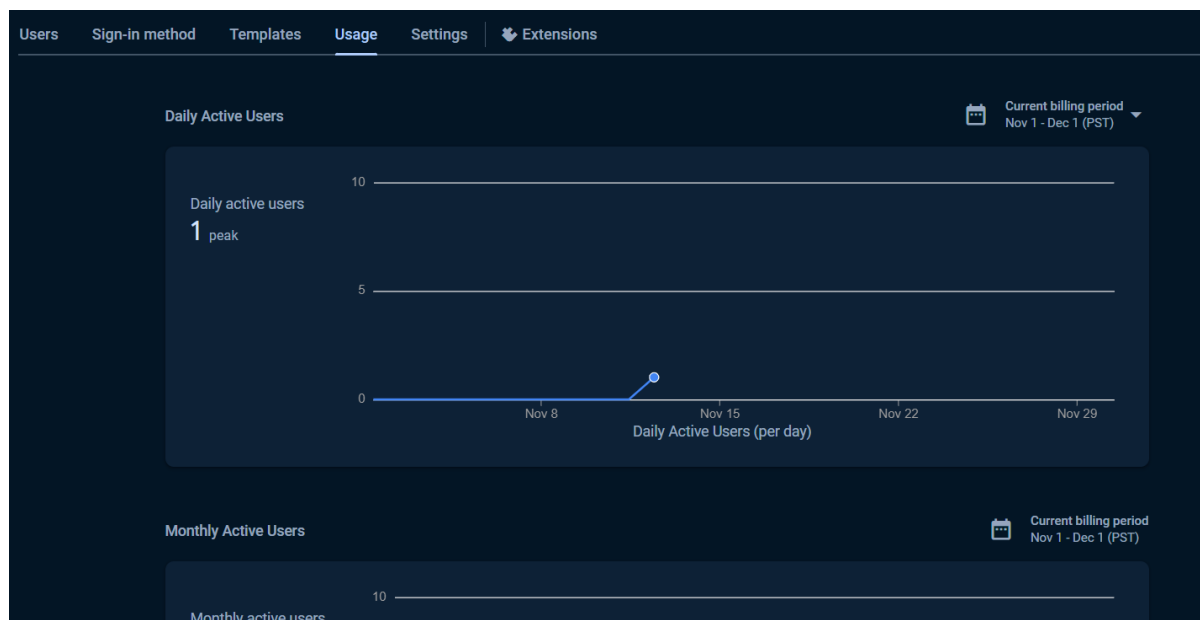


Рисунок 3.3 - Графік активності реєстрації користувачів

Всі дані з додатку зберігаються у Firestore Database. Одиницями збереження даних є колекції та документи. Колекція - це група документів, яка об'єднується по подібним критеріям. Документ є окремим записом у колекції, який представляється у вигляді JSON-файлу.

Спочатку створюється колекція, а потім документи, в яких вже і зберігаються дані. Документ має назву унікального ідентифікатор і за допомогою нього отримуються та перезаписуються дані.

У випадку адаптивної системи онлайн-мінігор було створено дві колекції - users та games. Колекція users зображена на рис. 4.4.

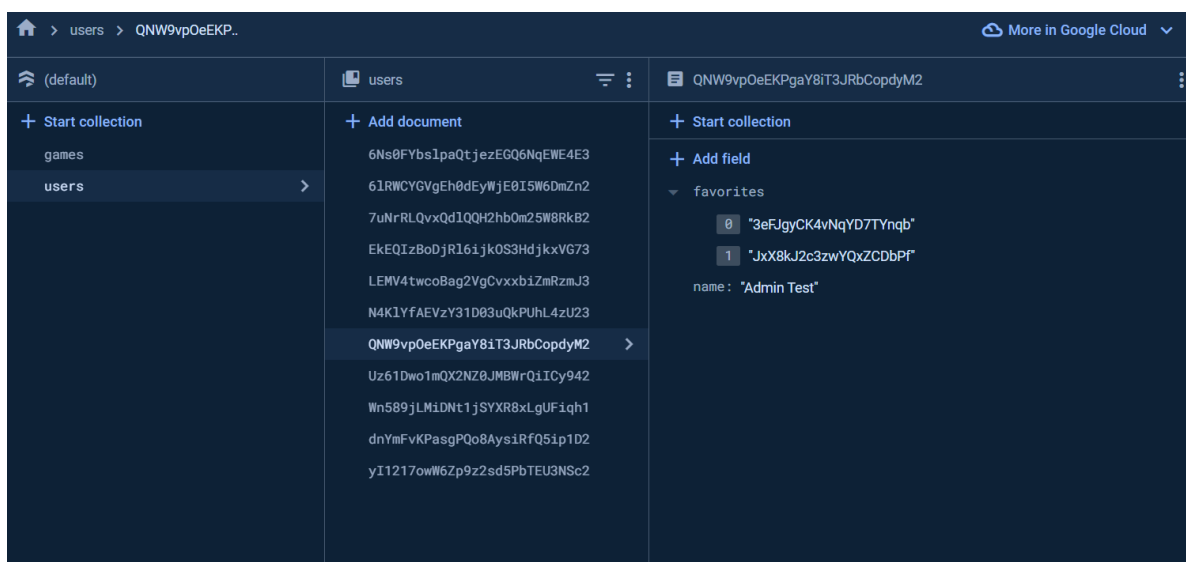


Рисунок 3.4 - Колекція users

Назвою документа є унікальний ідентифікатор користувача, який був створений автоматично при реєстрації. У правому підвікні бачимо дані про користувача, а саме його ім'я, та унікальні ідентифікатори ігор, які він додав до списку улюблених ігор.

Також тут є можливість напряму створити документ, видалити його, створити нову колекцію, змінити дані або додати чи видалити поля з даними.

Перейдемо до колекції games, де зберігатимуться дані про всі ігри, які були додані до вебресурсу. Колекція має документи, що є окремим записами про дані гри. Приклад такої гри зображений на рис. 3.5.

Гра має такі поля: назва, опис, дата створення, посилання на зображення, дані про рейтинг гри, тобто, який юзер як оцінив гру, а також тип гри та складність по шкалі від одного до п'яти. Всі ці дані будуть використані для максимально коректного підбору рекомендованих ігор для кожного користувача.

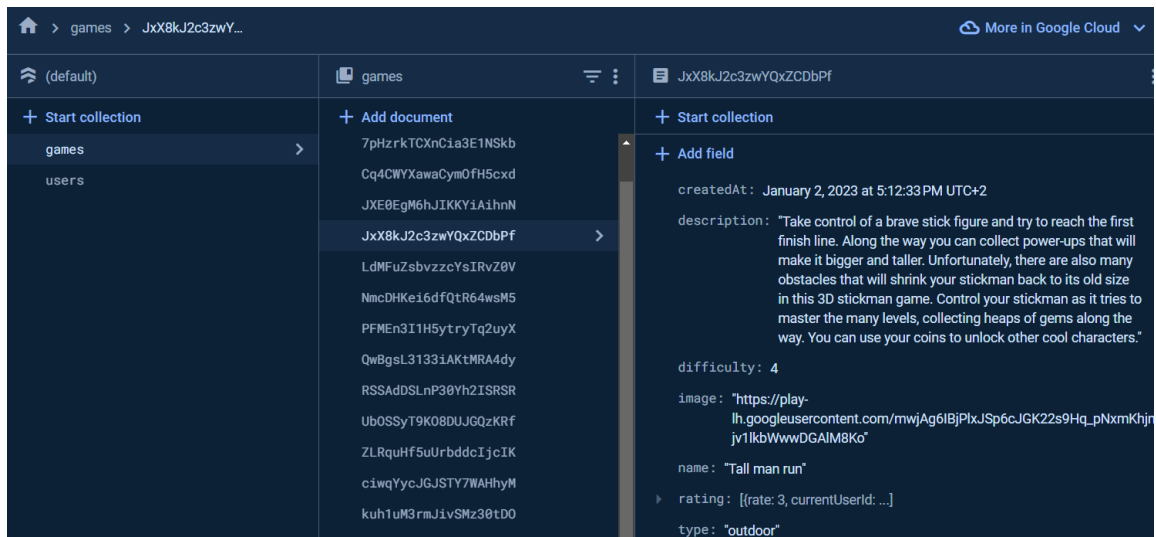


Рисунок 3.5 - Колекція games

Зпроектувавши базу даних, можна перейти до створення компоненту головної сторінки сайту. Планується показувати список категорій та список ігор, відповідно до обраної категорії та враховуючи всі потрібні чинники.

3.2 Реалізація головної сторінки

Отримання даних з бази даних про користувача та ігри написане, використовуючи відповідні функції та хук `useEffect`, що має в залежностях квадратні дужки, що означатиме отримання даних при першому рендері компоненту.

Лістинг 3.5 - Отримання даних на головній сторінці

```
useEffect(() => {
  try {
    if (currentUser)
      fetchUsersData()
      fetchGamesData()

  } catch (error) {
    console.log(error)
  }
}, [])
```

```

async function fetchUserData() {
  const docUser = doc(db, "users", currentUser.uid);
  const docSnapUsers = await getDoc(docUser);

  if (docSnapUsers.exists()) {
    await setUserData(docSnapUsers.data())
  } else {
    console.log("Document does not exist")
  }
}

async function fetchGamesData() {
  const collectionGames = query(collection(db, "games"),
orderBy('createdAt'));
  const querySnapshot = await getDocs(collectionGames);
  setGamesData(querySnapshot.docs)
}

```

Отримані дані записуються в локальний стан компонента для подальшого використання.

Також після авторизації з головної сторінки користувач може переглянути список ігор, які він відмітив як улюблені. Для цього нам треба вже взаємодія з двома колекціями одночасно. Спочатку треба отримати список унікальний ідентифікаторів ігор, які користувач додав, а вже потім відфільтрувати по цим ідентифікатор ігри та залишити лише ті, які були відмічені як улюблені.

Лістинг 3.6 - Отримання лише улюблених ігор

```

async function showFavorites() {
  let filteredGameRateArr = []
  userData.favorites?.forEach((gameId) => {
    filteredGameRateArr.push(...gamesData.filter(obj =>
obj.id === gameId))
  })
  setGamesData(filteredGameRateArr)
}

```

3.3 Реалізація адаптивних властивостей системи

Для того, щоб зробити систему більш адаптивною та показувати користувачу персоналізовану підбірку ігор, треба певні критерії для цього. Першим критерієм була обрана погода, це пояснюється тим, що, коли гарна погода на вулиці користувач навряд чи захоче грати в ігри, по типу шахів чи якихось головоломок. На мою думку, користувач віддасть перевагу активним іграм, такі як симулятор футболу, баскетболу, ігри, де треба багато ходити та пізнавати ігровий всесвіт.

Для цього треба отримувати дані про погоду по локації, з якої користувач відвідав вебресурс. Для реалізації подібного функціоналу треба використати декілька сторонніх ресурсів. Першочергово треба отримати дані про місто чи село, де користувач знаходиться. Для цього був використаний ресурс `ipari.co`, але може бути використаний будь-який довільний ресурс зі схожим функціоналом. Даний вебресурс повертає об'єкт з багатьма полями: `ip`, місто, область, країна, географічні довгота та широта та, навіть, площу та валюту країни. Але з цього потрібно буде лише місто.

Лістинг 3.7 - Повний JSON файл з даними

```
{
  "ip": "91.218.52.12",
  "version": "IPv4",
  "city": "Zaporizhzhya",
  "region": "Zaporizhzhia",
  "region_code": "23",
  "country": "UA",
  "country_name": "Ukraine",
  "country_code": "UA",
  "country_code_iso3": "UKR",
  "country_capital": "Kyiv",
  "country_tld": ".ua",
  "continent_code": "EU",
  "in_eu": false,
  "postal": "70423",
  "latitude": 47.7957,
  "longitude": 35.0489,
```

```

    "timezone": "Europe/Kyiv",
    "utc_offset": "+0200",
    "country_calling_code": "+380",
    "currency": "UAH",
    "currency_name": "Hryvnia",
    "languages": "uk,ru-UA,rom,pl,hu",
    "country_area": 603700.0,
    "country_population": 44622516,
    "asn": "AS51528",
  }

```

Процес отримання даних відбувається за допомогою хука `useEffect` в компоненті `LocationProvider` та записується у локальний стан компонента.

Лістинг 3.8 - Отримання та запис даних про локацію

```

const [location, setLocation] = useState(null)
const [pending, setPending] = useState(true);

useEffect(() => {
  const fetchData = async () => {
    try {
      const response = await
axios.get('https://ipapi.co/json/');
      const data = response.data;
      setLocation({
        city: data.city,
        region: data.region,
        country: data.country_name,
        latitude: data.latitude,
        longitude: data.longitude,
      });
      setPending(false)
    } catch (error) {
      console.error('Error fetching location:', error);
    }
  };

  fetchData();
}, []);

```

В даному випадку був використаний контекст та провайдер, в нього загорнутий головний компонент, це зроблено для того, щоб мати легкий та зручний доступ до змінної `location` у будь-якому компоненті проєкту.

Подібну маніпуляцію з отриманням локації можна зробити використовуючи вбудовані інструменти та методи мови `javascript`, але для цього треба, щоб

користувач дозволив використання даних про місцезнаходження. Використовується функція `getCurrentPosition` з об'єкту `navigator`.

Лістинг 3.9 - Код для отримання даних про місцезнаходження користувача

```
if (navigator.geolocation) {
  navigator.geolocation.getCurrentPosition(
    function(position) {

      var latitude = position.coords.latitude;
      var longitude = position.coords.longitude;

      console.log("Latitude: " + latitude + ", Longitude: " +
longitude);
    },
    function(error) {
      console.error("Error getting location: " +
error.message);
    }
  );
} else {
  console.error("Geolocation is not supported by this
browser");
}
```

Після того, як отримали дані про місцезнаходження користувача, потрібно з'ясувати яка погода в його регіоні. Для цього був використаний ресурс openweathermap.org. Безкоштовний план надає доступ до достатньо великого об'єму інформації, але знадобиться лише температура, наявність опадів та відсоток опадів.

Лістинг 3.10 - Отримання даних про погоду

```
const getWeather = async () => {
  const { data } = await
axios.get(`https://api.openweathermap.org/data/2.5/weather?q=${location.city}&units=metric&appid=${process.env.REACT_APP_KEY}`)
  const preparedWeather = {
    temp: data.main.temp,
    type: data.weather[0].main
  }
  setWeatherData(preparedWeather)
}
```

Далі треба прив'язати отримані дані з погодного вебресурсу до отримання персоналізованих даних для певного користувача. Спочатку треба прописати логіку, згідно якої різні ігри будуть отримувати типи: на відкритому повітрі або у приміщенні. Логіка буде простою та зрозумілою, що якщо на вулиці менше 15 градусів тепла та йде дощ, то будуть пропонуватися ігри, в які звично грати у приміщенні. Був використаний хук `useEffect`, з масивом залежностей зі значенням погодних даних, для того, щоб при зміні погоди, список ігор оновлювався відповідно.

Лістинг 3.11 - Логіка вибору типу гри

```
useEffect(() => {
  if (weatherData) {
    const { temp, type } = weatherData
    if (type === 'Rain') {
      setGameType(INDOOR_GAMES)
    }
    else if (temp > 15.0 && type === 'Sunny') {
      setGameType(OUTDOOR_GAMES)
    } else {
      setGameType(INDOOR_GAMES)
    }
  }
  fetchGamesData()
}, [weatherData])
```

Згідно цих оновлень треба змінити функцію отримання списку ігор, а саме додати до запиту параметр типу гри.

Лістинг 3.12 - Отримання списку ігор

```
async function fetchGamesData() {
  const collectionGames = query(collection(db, "games"),
    orderBy('createdAt'),
    where("type", "=", gameType));
  const querySnapshot = await getDocs(collectionGames);
  setGamesData(querySnapshot.docs)
}
```

Ще одним параметром для персоналізації списку ігор, що пропонуються є складність, насамперед вона залежить від віку користувача. Його користувач

вказує при реєстрації на вебресурсі онлайн-мініігор. Складність ігор розбита на три категорії: легкі, середні та складні. Логіка підбору заключається в вікових групах, на які будуть розподілятися користувачі. Перша група – це користувачі молодше десяти років включно та старше сорока років включно. Користувачі такої вікової групи будуть отримувати ігри з категорії складності - легкі. Наступною групою є користувачі від десяти років, але не старше п'ятнадцяти та старше двадцяти п'яти включно, але не старше 40 років. Така група буде отримувати багатший список ігор, який включати в себе також ігри середньої складності. Третьою та заключною віковою групою є проміжок від п'ятнадцяти років та до двадцяти п'яти років не включно.

Лістинг 3.13 - Встановлення складності відповідно до вікової групи користувача

```
if (currentUser && userData) {
  const { yearsOld } = userData;
  if (Number(yearsOld) <= 10 || Number(yearsOld) >= 40) {
    setGameDifficulty(EASY)
  } else if (Number(yearsOld) <= 15 || Number(yearsOld) >= 25
){
    setGameDifficulty(MIDDLE)
  } else {
    setGameDifficulty(ADVANCED)
  }
}
```

Обов'язково треба модифікувати функцію отримання ігор, адже додався ще один параметр відбору - складність. Всі ці умови можна адаптувати та змінити за потребою, змінити вік, перерозподілити вікові групи та складність самих ігор, додати якийсь ще новий різновид складності.

Лістинг 3.14 - Функція отримання списку ігор, що включає всі потрібні параметри

```
async function fetchGamesData() {
  const collectionGames = query(collection(db, "games"),
    orderBy('difficulty'), orderBy('createdAt'),
    where("type", "=", gameType),
```

```

        where("difficulty", "<=", gameDifficulty));
        const querySnapshot = await getDocs(collectionGames);
        setGamesData(querySnapshot.docs)
    }

```

Наступним етапом є створення сторінки самої гри. Концепт включає в себе створення сторінки з наступним наповненням. Основну частину вікна користувача буде займати блок з самою грою. Важливо зробити цей блок максимальним за шириною та висотою, але залишити місце для блоку даних з інформацією про гру.

Наповненням першої важливої секції є блок, де буде розміщена гра та блок з лівої сторони, де буде розміщена інформація про гру, можливість оцінки та кнопка додавання до списку улюблених ігор.

Лістинг 3.15 - Секція з інформацією про гру

```

<div className="game-info">
  <h5 className="game-name">{gameData.name}</h5>
  <h5 className="game-category">{gameCategory}</h5>
  <p className="game-description">{gameData.description}</p>
  <div className="game-rating">
    <Rating
      onClick={handleRating}
      initialValue={rating}
    />
  </div>
  <div className="game-favorites">
    <Button
      variant="contained"
      sx={{background: '#d27979', width: '100%'}}
      color="error"
      size="small"
      onClick={addToFavorites}
    >{ isFavorite ? <>
      Added to favorites
      <FavoriteIcon sx={{color: 'red', marginLeft:
'15px'}}>/>
      </> : <>
        Add to favorites
      <FavoriteBorder sx={{color: 'red', marginLeft: '15px'}}>/>
      </>}
    </Button>
  </div>
</div>

```

В цій секції розташовується така інформація про гру: назва, категорія, опис. Нижче розміщується блок для оцінки гри у вигляді п'яти зірок. Під ним буде знаходитись кнопка для додавання до списку улюблених ігор. Але якщо користувач вже додав гру, то буде відображатись відповідний напис.

В наступному блоці розміщується саме вікно гри, наразі це виконано заглушкою у вигляді зображення гри та напису, що тут буде розміщуватись сама гра.

Лістинг 3.16 - Блок із заглушкою для гри

```
<div className="game-window">
  <img className="game-preview" src={gameData.image}
alt={gameData.name}/>
  Place for the game here.
</div>
```

Під цією секцією буде розміщуватись наступна, яка буде містити в собі список запропонованих ігор. В першу чергу, будуть пропонуватися ігри в точно тій же категорії, яку має гра, на яку перейшов користувач. Але, щоб використати категорію гри для пошуку подібних, треба за допомогою унікального ідентифікатора гри отримати дані про неї.

Лістинг 3.17 - Отримання даних про гру

```
async function fetchGameData() {
  const docGame = doc(db, "games", gameId);
  const docSnapGame = await getDoc(docGame);

  if (docSnapGame.exists()) {
    await setGameData(docSnapGame.data())
    fetchSameGames(docSnapGame.data().category_id)
    let categoryId = await categories.find(category =>
category.id === docSnapGame.data().category_id)
    await setGameCategory(categoryId.name)
    const sumRate = docSnapGame.data().rating?.reduce(
      (accumulator, currentValue) => accumulator +
currentValue.rate,
      0,
    ) / docSnapGame.data().rating?.length;

    await setRating(sumRate ? sumRate : 0)
```

```

        currentUser && await fetchFavorites()
        setPending(false)

    } else {
        console.log("Document does not exist")
    }
}

```

Також в цьому лістингу виконується встановлення рейтингу гри, тобто розраховується сума всіх оцінок, які ставили цій грі та обчислюється середнє арифметичне з неї.

Отримавши категорію, робимо запит на отримання списку ігор, в цій же категорії, але потрібно обробити випадок, якщо таких ігор не знайдеться. Для цього буде використаний звичайний текст, який повідомляє про це.

Лістинг 3.18 - Компонент списку ігор

```

<div className="games-wrapper">
  {gamesList?.length ? gamesList.map((doc, index) => {
    return (
      <GameCard game={doc} key={index}/>
    )
  }) :
  <div>
    There are no games in this category...
  </div>
}
</div>

```

Наступним списком ігор буде той, який містить в собі ігри, які підходять користувачу за складністю та типом. Складність та тип були отримані раніше.

Лістинг 3.19 - Списки ігор

```

<div className="same-games-wrapper">
  <h3 className="same-games-title">Same games</h3>
  <GamesList gamesList={sameGames}/>
</div>
<div className="same-games-wrapper">
  <h3 className="same-games-title">Games you might like</h3>
  <GamesList gamesList={personalGames}/>
</div>

```

Наступним важливим функціональним рішенням є реалізація кнопки, яка буде додавати гру до списку улюблених. Сенс цього списку в швидкому доступі до ігор, які сподобались користувачу. Кожен запис з даними користувача в базі має масив, який складається з унікальних ідентифікаторів ігор, які він додав.

Але для початку треба отримати список обраних ігор та перевірити, чи в цьому масиві є ідентифікатор гри, на яку перейшов користувач.

Лістинг 3.20 - Отримання списку обраних ігор

```
async function fetchFavorites() {
  const docUser = doc(db, "users", currentUser.uid);
  const docSnapUsers = await getDoc(docUser);

  if (docSnapUsers.exists()) {
    const res = await
docSnapUsers.data().favorites?.includes(gameId)
    await setFavorite(res)
  } else {
    console.log("Document does not exist")
  }
}
```

Заповнюється внутрішній стан компонента для того, щоб можна було відображати потрібний напис для кнопки. Тобто, якщо гра вже додана до списку, то показувати відповідний напис та при натисканні на кнопку не додавати її ще раз, а навпаки - видаляти зі списку обраних.

Лістинг 3.21 - Додавання або видалення зі списку обраних ігор

```
async function addToFavorites() {
  if (!currentUser) {
    navigate('/login', {replace: true})
    return
  }
  const usersRef = collection(db, "users");
  await setDoc(doc(usersRef, currentUser.uid), {
    favorites: isFavorite ? arrayRemove(gameId) :
arrayUnion(gameId)
  }, { merge: true });
  await fetchFavorites()
}
```

ArrayRemove та ArrayUnion є вбудованими функціями з пакету Firebase. Такими методами відбувається редагування масиву обраних ігор в документі користувача в самому Firestore.

3.4 Реалізація оцінювання гри

Наступним предметом розробки є реалізація оцінювання гри. Це є важливим, адже таким чином розробники зможуть отримувати зворотній зв'язок від самих гравців. Оцінювання гри буде виконане у вигляді п'яти зірок, кожен користувач може оцінити гру, а також змінити свою оцінку, якщо вона була вже оцінена. Дані про оцінку гри зберігаються в документі з даними про гру в Firestore.

Лістинг 3.22 - Виставлення оцінки до гри

```
const handleRating = (rate) => {
  if (!currentUser) {
    navigate('/login', {replace: true})
    return
  }
  setRating(rate)
  updateGameData(rate)
}
```

Тут відбувається зміна внутрішнього стану компонента для відображення поточної оцінки користувача та, звичайно, оновлення даних самої гри для подальшого коректного відображення даних про гру.

Лістинг 3.23 - Оновлення даних про гру після зміни рейтингу

```
async function updateGameData(rate) {
  const docGame = doc(db, "games", gameId);
  const gameRateArr = gameData.rating
  const currentUserId = firebase.auth().currentUser?.uid
  const filteredGameRateArr = gameRateArr.filter(obj =>
obj.currentUserId !== currentUserId)
```

```

    await updateDoc(docGame, {
      rating: [...filteredGameRateArr, {currentUserId, rate}]
    })
  }
}

```

Спочатку треба відфільтрувати масив об'єктів з рейтингом про гру, для того, щоб уникнути дублювання записів. Далі просто оновити документ з даними про гру в самому Firestore.

На цьому завершується етап реалізації вебресурсу онлайн-мініігор, всі поставлені задачі, які були описані на етапі проєктування були виконані.

4 ТЕСТУВАННЯ ТА ВПРОВАДЖЕННЯ СИСТЕМИ

4.1 Вигляд сайту

Сайт складається з 4 сторінок: реєстрації, логіну, головної сторінки та сторінки гри. Головна сторінка зображена на рис. 4.1.

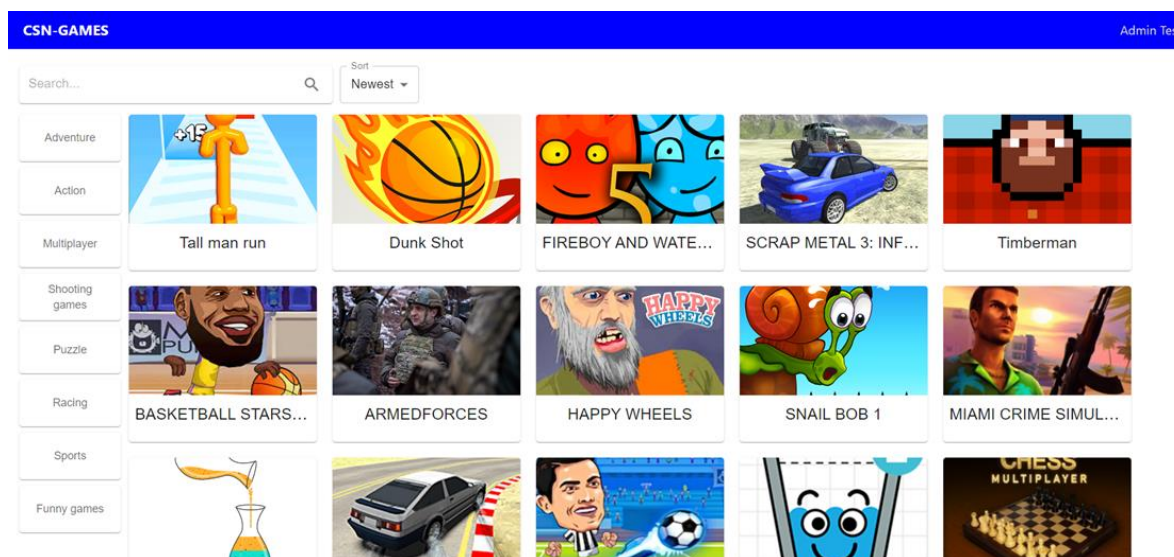


Рисунок 4.1 – Головна сторінка вебресурсу зі списком ігор та категорій

Sign up

Email Address *

Name *

Years Old *

Password *

Confirm password *

Already have an account? [Login](#)

Рисунок 4.2 – Сторінка реєстрації з полями форми та кнопкою

Форма має 4 поля: ім'я, пошта, вік, пароль та підтвердження паролю.

Вигляди форм для реєстрації та авторизації зображені на рис. 4.2 та рис. 4.3.

Sign in

Email Address *

Password *

Don't have an account? [Sign Up](#)

Рисунок 4.3 – Сторінка логіну з двома полями, за якими відбувається авторизація

Сторінка гри з головною секцією та блоком з інформацією про гру зображений на рис. 4.4.

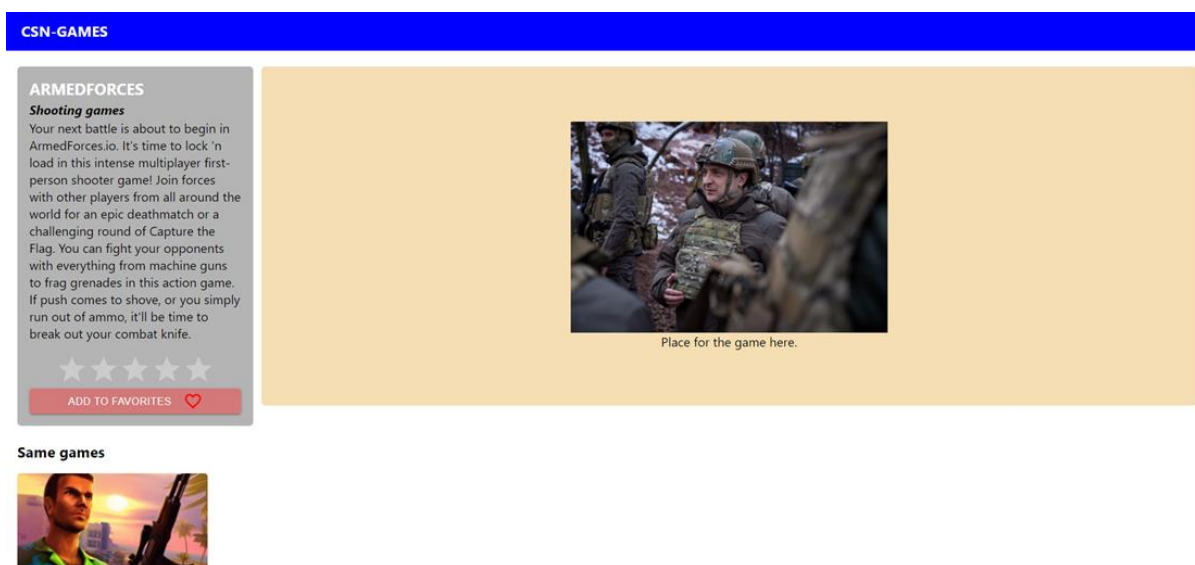


Рисунок 4.4 – Сторінка гри вебресурсу онлайн-мініігор

4.2 Тестування сайту

За допомогою спеціальних ресурсів, а саме validator.w3.org, була виконана валідація коду. Результат тестування приведений на рис 4.5. Він свідчить про те, що вебресурс не містить помилок або попереджень, а значить, що код написаний коректно та містить в собі всі потрібні теги та атрибути.

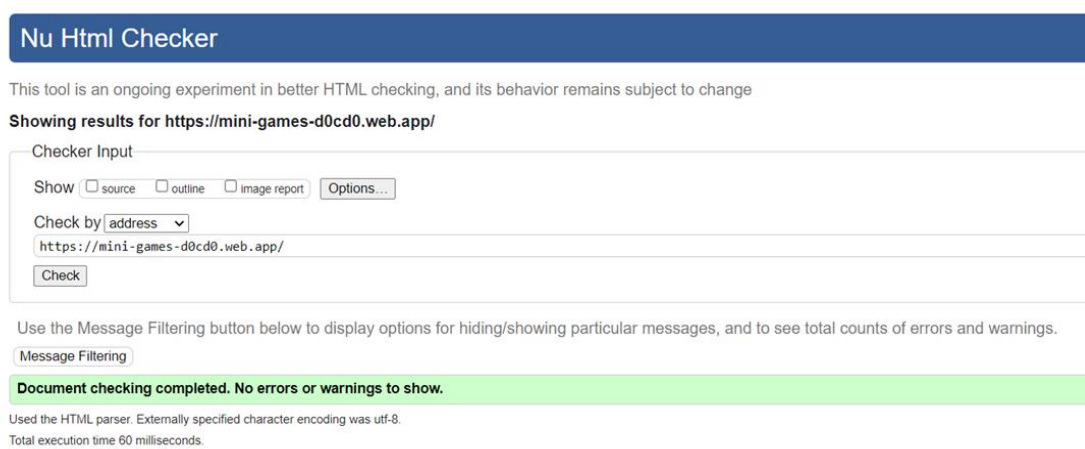


Рисунок 4.5 – Тестування сайту на валідність коду

Один і той же сайт може по-різному виглядати на різних комп'ютерах і пристроях. Це залежить від того, як певний браузер зчитує код сторінки. Кросбраузерність говорить про сумісність сайту з популярними браузерами і є ключовим показником якості верстки. Вигляд сайту в браузері “Google Chrome” (рис.4.6).

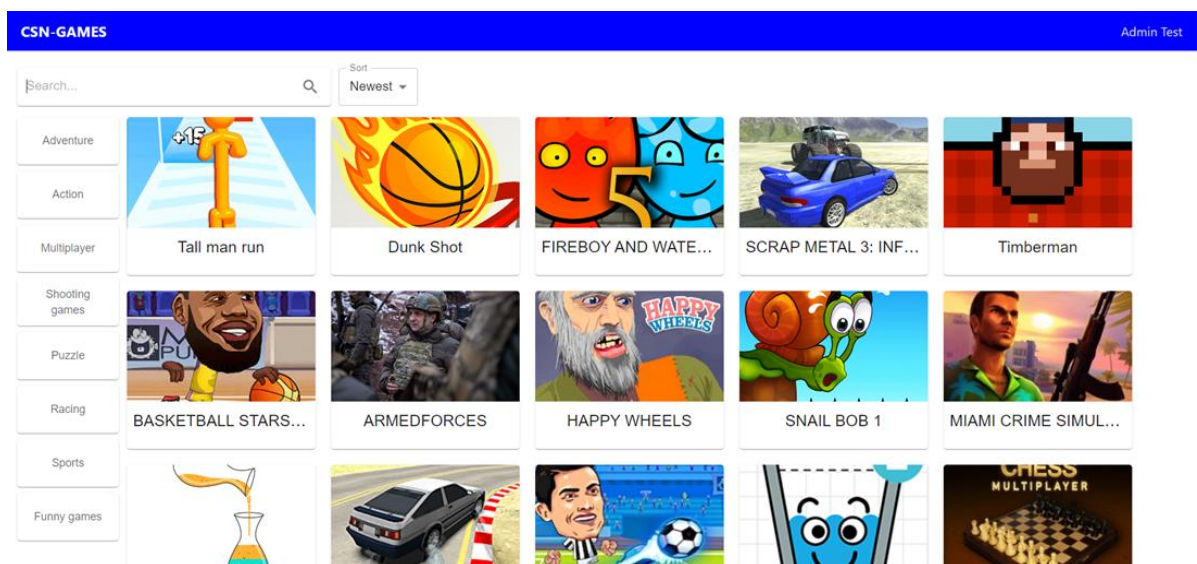


Рисунок 4.6 – Вигляд сайту в браузері “Google Chrome”

Вигляд сайту в браузері “Microsoft Edge” (рис.4.7).

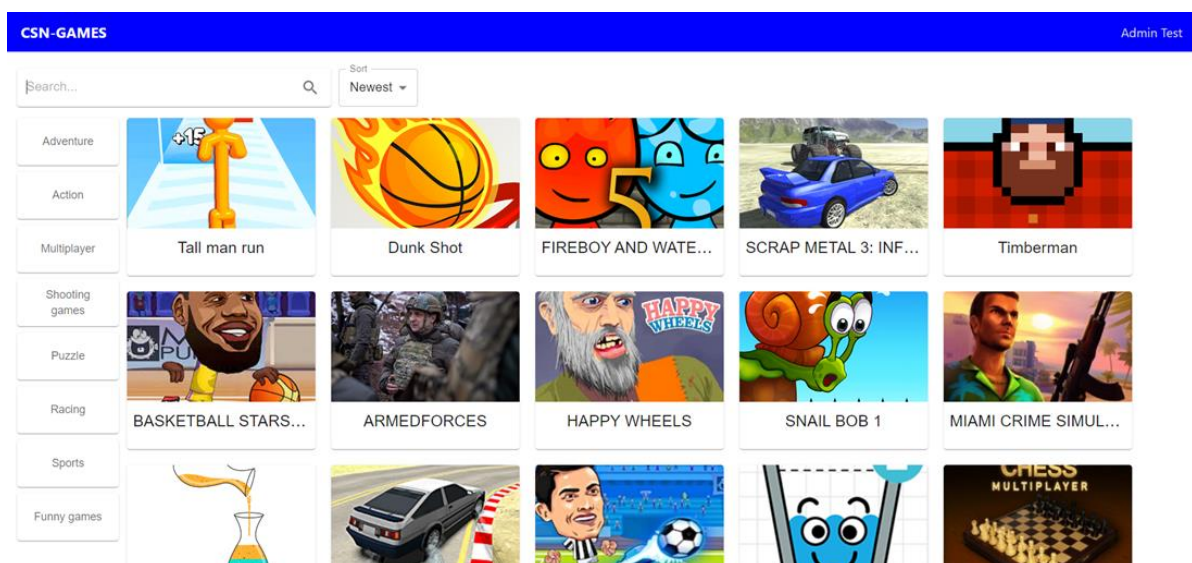


Рисунок 4.7– Вигляд сайту в браузері “Microsoft Edge”

Як можна побачити, сайт виглядає коректно в різних браузерях. Це означає, що сайт підтримує кросбраузерність.

Також важливим фактором є адаптивність сайту, він повинен коректно відображатись не тільки на дисплеях моніторів комп'ютера, а також і на телефонах. Вигляд головної сторінки на екрані з невеликою шириною зображений на рис. 4.8.

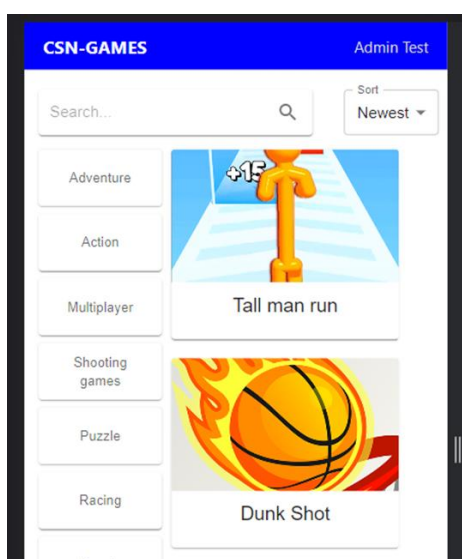


Рисунок 4.8 – Вигляд головної сторінки сайту на мобільному розмірі дисплея

Вигляд сторінки гри на екрані з невеликою шириною зображений на рис. 4.9.



Рисунок 4.9 – Вигляд сторінки з грою на мобільному розмірі дисплея

Можемо побачити, що сайт виглядає коректно, не має горизонтального скроллу та неправильно чи не гарно виглядаючих блоків.

Спочатку треба перевірити на поведінку сайту при введенні правильних даних та тих, яких немає в базі даних.

Sign in

Email Address *

Password *

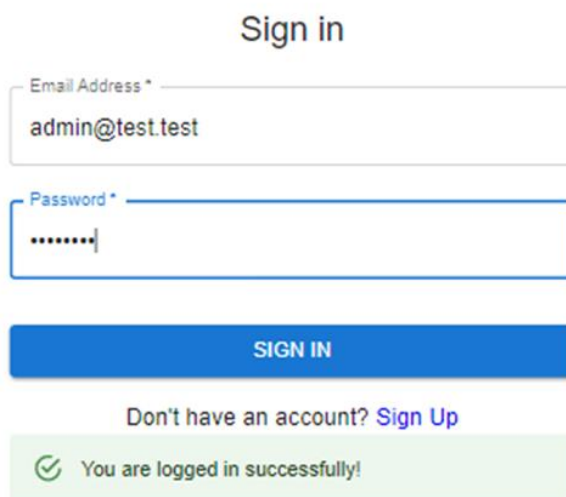
SIGN IN

Don't have an account? [Sign Up](#)

❗ There is no user with this credentials

Рисунок 4.10 – Вигляд сторінки логіну після введення некоректних даних

З'явився блок, який повідомляє нас про помилку при введенні даних (рис.4.10). Спробуємо ввести коректні дані.



The image shows a 'Sign in' form. At the top, the text 'Sign in' is centered. Below it, there are two input fields. The first field is labeled 'Email Address *' and contains the text 'admin@test.test'. The second field is labeled 'Password *' and contains a series of dots, indicating a password. Below the password field is a blue button with the text 'SIGN IN'. Under the button, there is a link that says 'Don't have an account? Sign Up'. At the bottom, there is a green box with a checkmark icon and the text 'You are logged in successfully!'.

Рисунок 4.11 – Вигляд сторінки логіну після введення вірних даних

Тепер дані були введені, які є в базі даних, про це свідчить зелений блок з відповідною інформацією там та відбувається перехід на головну сторінку сайту (рис. 4.11).

На головній сторінці присутній пошук та сортування, вибір ігор по категорії. Протестуємо саме фільтрацію по категорії. Наприклад, оберемо категорію Racing (рис. 4.12).

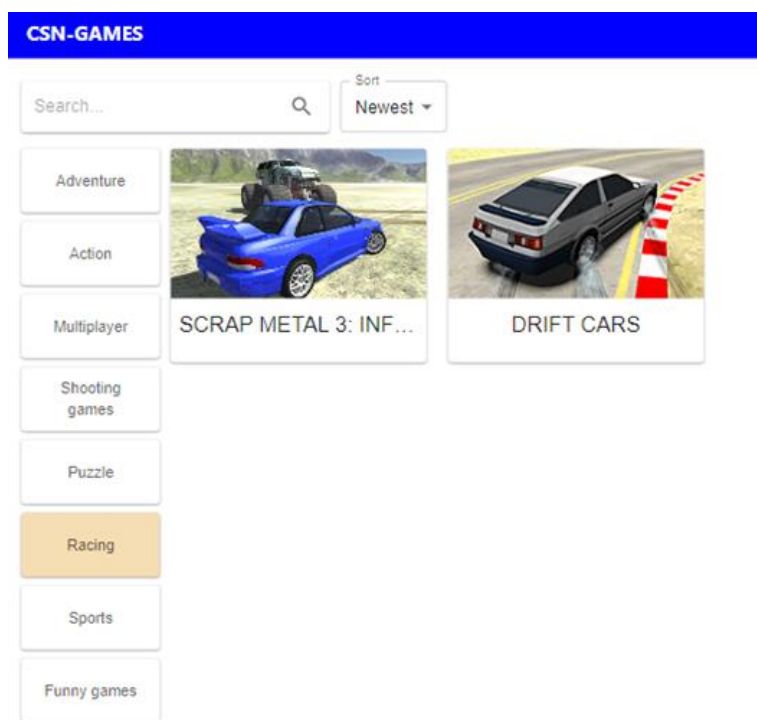


Рисунок 4.12 – Вигляд сторінки з обраною категорією Racing

Категорія підсвічена іншим кольором та показані ігри тільки з цієї категорії.
Перевіримо пошук.

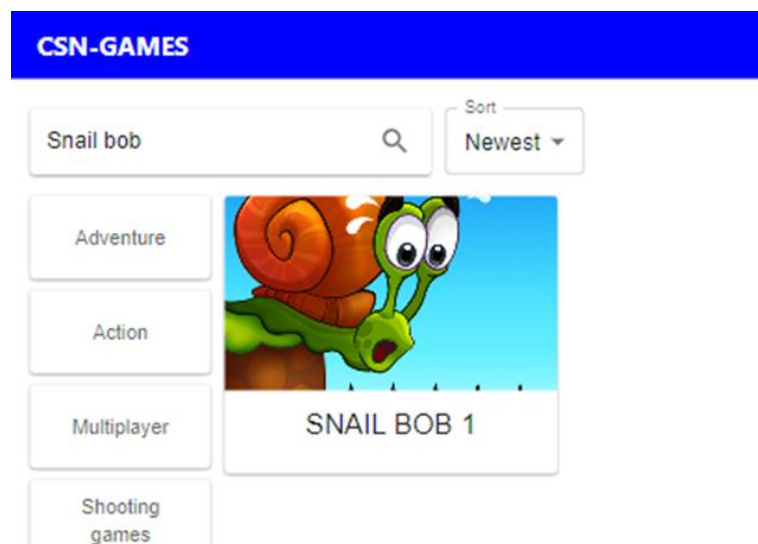


Рисунок 4.13 – Результат по запиту “Snail bob”

Пошук працює коректно та видав нам гру саме з такою назвою (рис. 4.13).

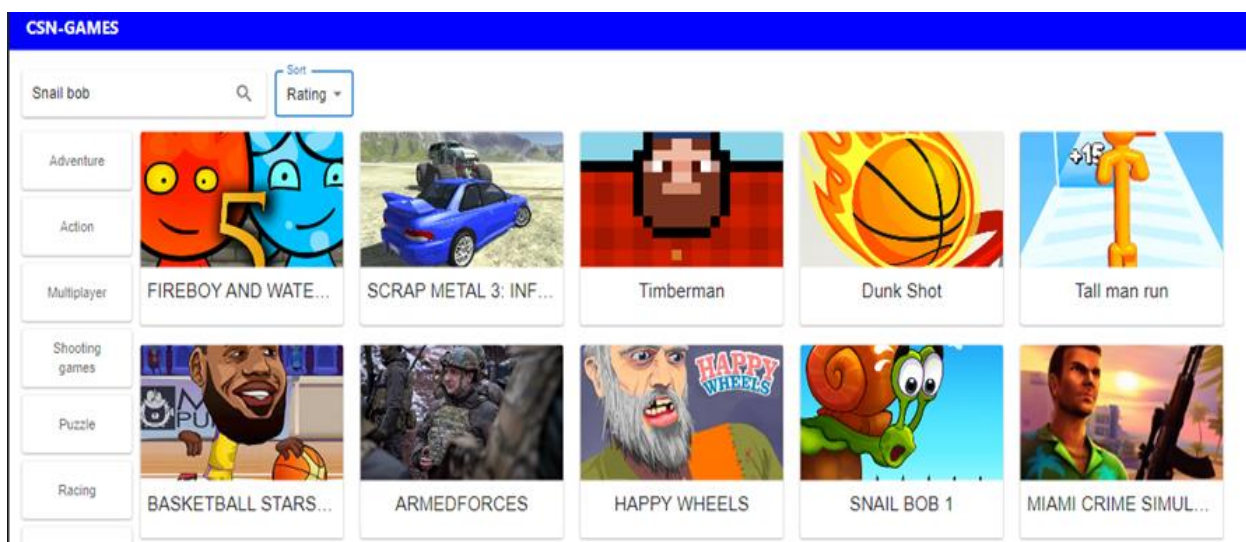


Рисунок 4.14 – Результат по сортуванню за рейтингом

Список ігор змінив свій порядок і тепер на першому місці гра з найкращим рейтингом (рис. 4.14).

4.3 Просування системи в мережі Інтернет

Просування сайту в Інтернеті – це дуже популярна тема на сьогоднішній день. Зробивши власний сайт, розробник-початківець стикається з тим, що його оригінальна ідея чомусь не приваблює маси. Так, ресурс може зацікавити людей, але лише тоді, коли про нього дізнаються. Саме для цієї мети існує оптимізація і просування сайтів в мережі Інтернет.

Є деякі способи просування сайту в мережі Інтернет:

- пошукова оптимізація;
- банери;
- контекстна реклама;
- соціальні мережі та спільноти.

SEO-просування сайтів в Інтернеті направлено на поширення ресурсу на інших джерелах, а також на боротьбу за його лідерські позиції в пошуковій

системі. Головне – підвищити товарообіг. За допомогою даного способу можна і модернізувати сам ресурс (тобто працювати тільки з внутрішніми його факторами), і взаємодіяти з методами розкрутки, представленими далі.

Банери при просуванні сайтів відіграють дуже важливу роль. Це метод, який не змусить вас довго чекати підвищення показника відвідуваності ресурсу. Однак варто врахувати, що рекламу бажано поширювати на ті сторінки, які пов'язані з вашим родом діяльності, інакше банери виявляться марні.

Контекстна реклама є досить витратним методом просування сайту в Інтернеті, результати якого вельми ефективні.

Соціальні мережі та спільноти, як спосіб просування сайтів в Інтернеті дозволяє контактувати зі споживачем. Люди пишуть відгуки, діляться думкою, задають питання, переглядаючи ресурс. Так ви можете не тільки залучити клієнтів, але і якось удосконалити свій продукт.

Сайт пройшов перевірку на Search Engine Optimization за допомогою ресурсу «www.semrush.com» (рис. 4.15). А саме, на індексацію, на санкції, перевірку на віруси, також на контент: заголовки, кількість слів, розмір сторінки та ресурсів; швидкість завантаження.

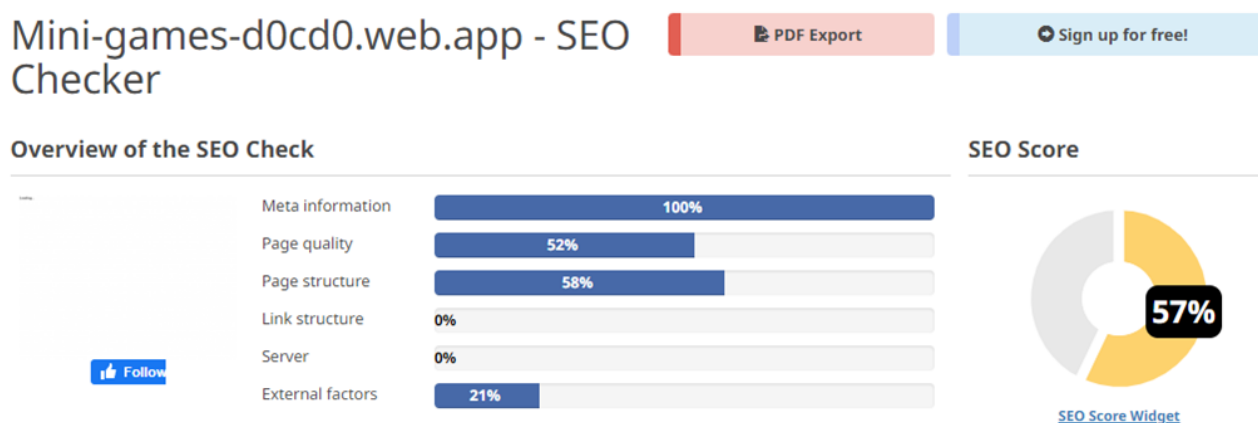


Рисунок 4.15 – Перевірка сайту на SEO

Як бачимо, сайт має результат в 57%. Це задовільний результат, адже над сайтом не працював професіональний SEO-спеціаліст і є над чим попрацювати задля покращення просування сайту в мережі інтернет.

4.4 Оптимізація

Грамотне складання семантичного ядра дозволить сайту потрапити в ТОП видачі пошукових систем. Для складання семантичного ядра були використані різні сервіси, та обрані такі ключові слова, як: «games», «mini-games», «browser games», «puzzle game», «flash games».

На головній сторінці розміщується список ігор з категоріями та зручним пошуком. Також на кожній другорядній сторінці є логотип зверху з посиланням, який повертає на головну сторінку сайту.

Важливим є використання метатегів, одними з найважливіших є: тег заголовку, тег опису, заголовки, використання атрибуту alt в зображеннях та robots метатеги.

Тег заголовка – ваш головний і найважливіший тег.

Елемент <title> зазвичай відображається як активний заголовок у результатах пошуку, а також відображається в соціальних мережах і браузерях.

Заголовок сторінки все ще є першим, що шукач бачить у результатах пошуку та вирішує, чи ймовірно, що сторінка відповідає меті пошуку.

Добре написане може збільшити кількість кліків і трафік, що принаймні трохи вплине на рейтинг.

Простий експеримент також може показати, що Google більше не потребує вашого тегу заголовка для включення ключового слова з точною відповідністю, щоб знати тему, яку охоплює сторінка.

Метаопис також міститься в <head> вебсторінки та зазвичай (хоча точно не завжди) відображається у фрагменті SERP разом із заголовком і URL-адресою сторінки.

Опис займає більшу частину фрагмента SERP і запрошує шукачів натиснути на ваш сайт, обіцяючи чітке та вичерпне рішення для їхнього запиту.

Якщо опис містить ключові слова, які шукач використав у своєму пошуковому запиті, вони відображатимуться в пошуковій видачі жирним шрифтом. Це значною мірою допоможе ввиділитися та повідомити шукачу, що саме він знайде на вашій сторінці. Ключові слова, які використовуються зображені на рис. 4.16.

```
<title>Csn Games</title>
<meta
  name="description"
  content="Web site where you can play any mini-games you want. Action ,puzzle, shooting, multiplayer"
/>
```

Рисунок 4.16 – Мета теги заголовку та опису

Теги заголовків — це теги HTML, які використовуються для визначення заголовків і підзаголовків у вмісті з інших типів тексту (наприклад, тексту абзацу).

Хоча теги H2-H6 не вважаються такими важливими для пошукових систем, правильне використання тегів H1 наголошується в багатьох галузевих дослідженнях. Приклад використання тегу H1 (рис. 4.17).

```
<h1
  className='header-title'
  onClick={headerLogoClick}
>
  CSN-GAMES
</h1>
```

Рисунок 4.17 – Мета теги заголовків

Атрибут alt зображення додається до тегу зображення, щоб описати його вміст. Атрибути Alt важливі для оптимізації на сторінці з двох причин:

Альтернативний текст відображається відвідувачам, якщо не вдається завантажити певне зображення (або якщо зображення вимкнено).

Атрибути Alt забезпечують контекст, оскільки пошукові системи не можуть «бачити» зображення (рис. 4.18).

```
<div className="game-window">  
  <img className="game-preview" src={gameData.image} alt={gameData.name}/>  
  Place for the game here.  
</div>
```

Рисунок 4.18 – Атрибут alt для зображення

Саме тут атрибут має значення назви гри, тому легко буде зрозуміти, що це навіть, якщо по якійсь причині зображення не буде відображатись.

Інструкції у файлі robots.txt мають сильний вплив на SEO (оптимізацію пошукових систем), оскільки файл дозволяє керувати пошуковими роботами. Однак якщо агенти користувача надто сильно обмежені інструкціями щодо заборони, це може негативно вплинути на рейтинг вашого вебсайту. Також треба враховувати, що класифікування не буде проходити разом із вебсторінками, які були виключені за допомогою disallow у robots.txt.

Правильно використовуючи robots.txt, дуже легко переконатися, що всі важливі частини вашого сайту скануються пошуковими роботами. Отже, ваш важливий вміст сторінки може бути проіндексовано Google та іншими пошуковими системами.

ВИСНОВКИ

У сучасному світі онлайн-мініігри відіграють ключову роль та вже встигли стати доступними майже для кожної людини на планеті. Це все відбулося завдяки розвитку інтернет-технологій. Онлайн-мініігри можуть бути короткими та простими, проте можуть бути і більш складні та високоінтелектуальні. Доступ до таких ігор може відбуватись через браузер, мобільні пристрої.

Однією з найголовніших переваг є проста доступність для широкої аудиторії та різних вікових груп. Головне, що ігри не треба завантажувати чи встановлювати якесь додаткове програмне забезпечення. Гравці починають гру, використовуючи тільки браузер та інтернет. Користувач не витрачає час на встановлення та зберігає ємність свого накопичувача.

В ході дослідження та розробки адаптивної системи онлайн мініігор було проведено аналіз існуючих систем-аналогів та визначено ключові аспекти, які необхідно врахувати для досягнення ефективного ігрового досвіду. З урахуванням отриманих результатів було обрано концепцію, яка максимально відповідає вимогам та цілям дослідження.

Обрана концепція адаптивної системи базується на індивідуальному підході до користувачів, де основні параметри ігрового середовища автоматично адаптуються до їхніх уподобань та навичок. Це дозволяє забезпечити персоналізований ігровий досвід, що підвищує задоволення від гри та сприяє залученню аудиторії.

У проєкті були розроблені вимоги до створюваного ПЗ, а саме враховано ціль створюваної системи, цільову аудиторію, вимоги за функціональністю.

Спроектowana архітектура програмного забезпечення та змодельована діаграма розміщення типового вебресурсу та вебресурсу онлайн мініігор.

Також спроектований макет вебінтерфейсу користувача, який зазначає розміщення різних елементів, блоків, секцій на сторінках.

Адаптивна система онлайн-мініігор була втілена у життя з повною відповідністю до розробленого проєкту та визначених вимог. Кожен етап розробки, від аналізу і вибору концепції до проєктування архітектури та створення вебінтерфейсу, був здійснений із великою уважністю до деталей та відповідності стандартам якості. Результатом є функціональна та естетично збалансована система, яка готова до використання та відповідає усім поставленим завданням і вимогам.

Онлайн-мініігри в браузері залишаються популярними і продовжують розвиватися, щоб відповідати очікуванням сучасних гравців. Існує кілька ключових тенденцій розвитку, таких як різні моделі монетизації та адаптивні системи рекомендацій, поліпшення графіки, інтеграція соціальних функцій. Різні моделі монетизації дають змогу гравцям вибирати, як підтримувати ігри, і важливо, щоб вони були збалансовані. Графіка стає більш якісною та привабливою, що робить ігри більш захоплюючими. Соціальні функції, такі як змагання та спілкування, стають дедалі важливішими в онлайн-іграх. Адаптивні системи допомагають рекомендувати ігри на основі вподобань гравців, покращуючи їхній досвід і монетизацію.

Важливість такої системи та онлайн-ігор полягає в тому, що вони можуть бути використані, як розважальний, освітній та маркетинговий інструмент.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Що таке гра ? URL: <https://blog.liga.net/user/amyisenko/article/38093>
2. Класифікація ігор. URL: https://gamesisart.ua/game_class_all.html
3. Етапи створення ігор. URL: <https://blog.fugas.space/gamedev-stages>
4. Структура гри. URL: <https://blog.fugas.space/gamedev-stages/>
5. Інститут Інженерів з Електротехніки та Радіоелектроніки (IEEE)
Рекомендована методика складання специфікацій вимог до програмного забезпечення, New York, NY 10017-2394, USA 2014. 34 с.
6. React Js – Документація: <https://react.dev/learn>
7. Firebase – Документація: <https://firebase.google.com/docs>
8. Проєктування та моделювання програмного забезпечення сучасних інформаційних систем: навчальний посібник / Г. В. Табунщик, Т. І. Каплієнко, О. А. Петрова. – Запоріжжя: ЗНТУ, 2016. – 270 с.
9. Проєктування програмного забезпечення з використанням UML: М. Ф. Єфремов, Ю. М. Єфремов, В. М. Єфремов. – Житомир: ЖДТУ, 2016.
10. Методи та засоби виявлення дефектів проєктування об'єктно-орієнтованого програмного забезпечення: О. С. Нечай, М. О. Сидоров. – НАУ, 2009.
11. Основне використання REACT JS / Р.В. Бойко, М.Ю. Тягунова // Тиждень науки-2023. Факультет комп'ютерних наук і технологій: наук.-техн. конф., 24-28 квітня 2023 р.
12. Основне використання Firebase / Р.В. Бойко, М.Ю. Тягунова // Тиждень науки-2023. Факультет комп'ютерних наук і технологій: наук.-техн. конф., 24-28 квітня 2023 р.
13. Розробка комп'ютерних ігор: Біловол Є. В., Глушко І. М. – 2013. – 146 с.
14. Класифікація комп'ютерних ігор: В. С. Ашанін, С. С. Пятисоцька. – Житомир: ЖДТУ, 2021.

15. Основні етапи створення комп'ютерної гри з використанням web технологій: С. О. Музика, О. О. Мосіюк. – Житомир: ЖДУ, 2022.
16. Комп'ютерна гра. Інструменти і методологія створення комп'ютерних ігор: Н. Тарапата, М. Семьонова. – ЛДУ БЖД: - 2017.
17. Особливості розробки комп'ютерних ігор: М. Р. Моцний. – ЧНУ: - 2015.
18. Проектування архітектури web–застосування на основі моделі якості проектування: О. Г. Харченко, І. О. Галай. – НАУ, 2010.
19. Експерта система проектування архітектури програмного забезпечення: О. Харчеко, І. Бондарчук. – НБУВ, 2013.
20. Архітектура та проектування програмного забезпечення: А. О. Лісняк, О. М. Мильцев. – Запоріжжя: ЗНУ, 2022.