Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська дипломна робота

на тему: «Розробка ігрового Web-застосунку для розвитку логічного мислення. Частина 1. Проектування та візуалізація інтерфейсу застосунку»

Виконав: студент 4 курсу групи 2ПІ-20б
спеціальності 121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Мацедонський С.О.
(прізвище та ініціали)
Керівник: к. т. н., доц. каф. ПЗ Майданюк В.П.
(прізвище та ініціали)
Рецензент: к.т.н., доц. каф. ОТ Кожемяко А.В.
(прізвище та ініціали)

Допущено до захисту
Зав. кафедри ПЗ Романюк О. Н.
«10» червня 2024 р.

Вінниця ВНТУ - 2024 рік

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший (бакалаврський)
Галузь знань 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»
Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н._____
«12» березня 2024 р.

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Мацедонському Станіславу Олександровичу

1. Тема роботи – «Розробка ігрового Web-застосунку для розвитку логічного мислення. Частина 1. Проектування та візуалізація інтерфейсу застосунку»

Керівник роботи: Майданюк Володимир Павлович, к.т.н., доц. кафедри ПЗ, затверджені наказом вищого навчального закладу від 11 березня 2024 р. № 80.

2. Строк подання студентом роботи 3 червня 2024 р.

3. Вихідні дані до роботи: середовище розробки Visual Studio Code, мова розробки Typescript, бібліотека React, система управління базою даних Firebase Realtime Database, система авторизації Firebase Authentication, базові алгоритми.

4. Зміст розрахунково-пояснювальної записки: вступ; аналіз стану питання проектування та візуалізації інтерфейсу застосунку, порівняльний аналіз аналогів, аналіз методів розв'язання задачі, постановка задач дослідження; розробка моделі застосунку, розробка структури клієнтської частини веб-застосунку, розробка алгоритмів роботи програми, варіантний аналіз та обґрунтування вибору мови програмування ,вибір середовища розробки, вибір засобів реалізації інтерфейсу програмного застосунку, розробка структурних схем інтерфейсу, розробка

програмного інтерфейсу застосунку; аналіз методів тестування, тестування інтерфейсу застосунку; висновки; список використаних джерел; додатки.

5. Перелік графічного матеріалу: титульний слайд, актуальність теми, мета, об'єкт та предмет дослідження, наукова новизна, практична цінність, аналоги, порівняльний аналіз аналогів, використані технології, модель застосунку, алгоритм роботи застосунку, блок-схема алгоритму перегляду гри, тестування інтерфейсу, тестування інтерфейсу, тестування інтерфейсу, тестування інтерфейсу, тестування інтерфейсу, виконані задачі, апробація результатів роботи і публікації, завершальний слайд.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата			
		Завдання видав		Завдання прийняв	
1-4	Майданюк В.П., к.т.н., доцент кафедри ПЗ	13.03.24		03.06.24	

7. Дата видачі завдання 13 березня 2024

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз стану питання проектування та візуалізації застосунку	14.03.24 –22.03.24	
2	Розробка алгоритмів роботи програми	13.03.24 –22.03.24	
3	Аналіз та обгрунтування вибору мови програмування та середовища розробки	25.03.24 –19.04.24	
4	Розробка програмного інтерфейсу застосунку	22.04.24 –10.05.24	
5	Тестування інтерфейсу застсоунку	13.05.24 – 24.05.24	
6	Оформлення матеріалів до захисту БДР	27.05.24 – 30.05.24	

Студент _____ (підпис) Мацедонський С.О. (прізвище та ініціали)

Керівник бакалаврської дипломної роботи _____ (підпис) Майданюк В.П. (прізвище та ініціали)

АНОТАЦІЯ

УДК 004.51

Мацедонський С. О. «Розробка ігрового Web-застосунку для розвитку логічного мислення. Частина 1. Проектування та візуалізація інтерфейсу застосунку»: бакалаврська дипломна робота зі спеціальності 121 Інженерія програмного забезпечення, освітня програма – Інженерія програмного забезпечення. Вінниця: ВНТУ, 2024. 57 с.

На укр. мові. Бібліогр. : 18 назв ; рис. : 35 ; табл. 3.

У бакалаврській дипломній роботі проведено аналіз стану питання проектування та візуалізації графічного інтерфейсу. Було проведено аналіз аналогів, визначено їх переваги і недоліки та доведено доцільність розробки власного програмного застосунку.

Виконане обґрунтування вибору мови та середовища програмування, а також технологій, які були використані під час розробки мобільного застосунку. Розроблено інтерфейс WEB-застосунку для розвитку логічного мислення.

Робота реалізована за допомогою мови програмування TypeScript. В якості середовища для розробки програмного забезпечення було обрано Visual Studio Code.

Отримані в бакалаврській дипломній роботі результати можна використати для інтеграції із алгоритмами WEB-застосунку, що дозволяє розвивати логічне мислення.

Ключові слова: web-застосунок, логічне мислення, інтерфейс.

ANNOTATION

UDC 004.51

Matsedonskyi S. O. "Development of a game Web application for the development of logical thinking. Part 1. Design and visualization of the application interface": bachelor's thesis on the specialty 121 Software Engineering, educational program - Software Engineering. Vinnytsia: VNTU, 2024. 57 p.

In Ukrainian language Bibliogr. : 18 titles; Fig. : 35; table 3.

In the bachelor's thesis, an analysis of the state of design and visualization of the graphic interface was carried out. An analysis of analogues was carried out, their advantages and disadvantages were determined, and the expediency of developing one's own software application was proven.

The justification of the choice of language and programming environment, as well as the technologies that were used during the development of the mobile application, was carried out. The WEB application interface for the development of logical thinking has been developed.

The work is implemented using the TypeScript programming language. Visual Studio Code was chosen as the software development environment.

The results obtained in the bachelor thesis can be used for integration with the algorithms of the WEB application, which allows for the development of logical thinking.

Keywords: web application, logical thinking, interface.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Розвиток логічного мислення є дуже важливим для людей будь-якого віку. Вміння логічно мислити допомагає нам ефективно вирішувати проблеми, приймати виважені рішення та успішно навчатися протягом життя. На щастя, сучасні технології пропонують чимало цікавих можливостей для тренування та вдосконалення цих навичок.

Зараз існує багато різноманітних мобільних застосунків, спеціально розроблених для розвитку логічного мислення. Ці додатки пропонують широкий спектр завдань та вправ, спрямованих на тренування уваги, пам'яті, гнучкості мислення та інших важливих когнітивних здібностей [1].

Застосунки часто мають ігровий формат, що робить процес навчання та вдосконалення навичок захоплюючим та мотивуючим для користувачів. Вони можуть відстежувати прогрес користувачів, пропонувати персоналізовані рекомендації та адаптувати складність завдань відповідно до їхніх індивідуальних потреб.

Регулярне використання таких застосунків може суттєво покращити здатність людей вирішувати складні завдання, мислити критично та ефективно навчатися протягом життя. Це робить їх дуже актуальними та корисними інструментами для розвитку важливих когнітивних навичок.

Окрім мобільних застосунків, існують й інші цифрові та offline-інструменти, що також можуть бути ефективними для тренування логічного мислення, наприклад, онлайн-курси, інтерактивні головоломки та настільні ігри. Поєднання різноманітних підходів допомагає людям всебічно розвивати свої здібності.

Застосунки для розвитку логічного мислення стають все популярнішими, оскільки вони пропонують зручний та захоплюючий спосіб вдосконалювати важливі когнітивні навички. Регулярні тренування з їх допомогою можуть принести відчутну користь для людей різного віку.

Зв’язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі

програмного забезпечення.

Мета та завдання дослідження. Метою роботи є підвищення зручності використання застосунку для розвитку логічного мислення людини за рахунок розширення функціоналу клієнтської частини застосунку.

Основними задачами дослідження є:

- провести аналіз стану питання проектування та візуалізації інтерфейсу;
- розробити модель застосунку;
- розробити структуру клієнтської частини веб-застосунку;
- розробити алгоритм роботи програми;
- провести варіантний аналіз та обґрунтувати вибір мови програмування, середовища розробки та засобів реалізації інтерфейсу застосунку;
- розробити інтерфейс для реєстрації та авторизації користувача;
- розробити профіль користувача;
- розробити логіку перегляду зіграних ігор;
- розробити інтерфейс для гри;
- забезпечити адаптивність застосунку для різних видів пристроїв;
- провести тестування.

Об’єкт дослідження - процес створення інтерфейсу ігрового web-застосунку для розвитку логічного мислення.

Предмет дослідження - методи і програмні засоби реалізації інтерфейсу ігрового web-застосунку для розвитку логічного мислення.

Методи дослідження. У процесі досліджень використовувались: теорія людино-машинної взаємодії, теорія алгоритмів, методи комп’ютерної графіки для розробки макетів інтерфейсу застосунку та модулів застосунку; комп’ютерне моделювання для аналізу та перевірки отриманих теоретичних положень.

Новизна отриманих результатів.

1. Подальшого розвитку отримав інтерфейс профілю, у якому, на відміну від існуючих, доступний покроковий перегляд усіх зіграних ігор користувача, що

дозволило проводити аналіз помилок та покращувати когнітивні навички.

2. Подальшого розвитку отримав інтерфейс користувача застосунків для розвитку логічного мислення, у якому, на відміну від існуючих, забезпечено адаптивність та кросплатформність, що забезпечує можливість використання різних пристроїв.

Практична цінність отриманих результатів. Практична цінність одержаних результатів полягає в тому, що на основі отриманих в бакалаврській дипломній роботі теоретичних положень запропоновано алгоритми та розроблено програмні засоби клієнтської частини застосунку для розвитку логічного мислення.

Особистий внесок здобувача. Усі результати, викладені у бакалаврській дипломній роботі, отримані автором особисто. У друкованій праці [2], опублікованій у співавторстві, автору належать такі результати: клієнтська частина модуля авторизація та реєстрація користувача з використанням сервісів Google; модуль формування рейтингу користувача залежно від успішності його гри; клієнтська частина застосунку для розвитку логічного мислення.

Апробація матеріалів бакалаврської дипломної роботи. Основні положення бакалаврської дипломної роботи доповідалися та обговорювалися на «LIII Всеукраїнській науково-технічній конференції факультету інформаційних технологій та комп'ютерної інженерії (2024)».

Публікації. Основні результати досліджень опубліковано в 1 науковій праці у матеріалах конференції.

1 АНАЛІЗ СТАНУ РОЗВИТКУ ПРОГРАМ ДЛЯ РОЗВИТКУ ЛОГІЧНОГО МИСЛЕННЯ

1.1 Аналіз стану питання проектування та візуалізації інтерфейсу застосунку

Розробка інтерфейсу WEB-застосунків, спрямованих на розвиток критичного мислення, вимагає особливого підходу. Структура і навігація інтерфейсу мають бути логічно організовані та інтуїтивно зрозумілими, щоб користувач міг легко орієнтуватися в додатку і зосереджуватися на змісті, а не на пошуку необхідних функцій.

Візуальне представлення інформації у таких додатках має бути розроблене так, щоб сприяти критичному аналізу. Наприклад, за допомогою діаграм, інфографіки, інтерактивних візуалізацій, які допомагають користувачам виявляти закономірності та робити висновки.

Інтерфейс має заохочувати активну взаємодію користувачів - надавати їм можливість ставити запитання, коментувати, робити примітки, експериментувати з варіантами рішень. Така активна участь стимулює критичне мислення.

Додаток повинен надавати користувачам чіткий і своєчасний зворотній зв'язок щодо їхніх дій, а також контекстні підказки, які допомагають зрозуміти завдання та знаходити необхідні інструменти. Це сприяє ефективному навчанню.

Гнучкість і адаптивність інтерфейсу дозволяють користувачам налаштовувати його відповідно до своїх потреб і стилів мислення. Це заохочує критичне ставлення до інформації та підвищує залученість.

Інтерфейс має бути лаконічним, з виразними акцентами на ключових елементах. Це допомагає користувачам зосереджуватися на найважливішому та уникати відволікання. Доступність і універсальний дизайн забезпечують рівні умови для розвитку критичного мислення.

Додаток має можливість взаємодіяти з іншими інструментами, що дозволяє користувачам поєднувати різні джерела інформації та методи аналізу. Це сприяє

всебічному розгляду проблем.

Інтерфейс має передбачати механізми для отримання зворотного зв'язку від користувачів щодо їхнього досвіду та ефективності додатка в розвитку критичного мислення. Це дозволяє вдосконалювати продукт.

Розробка інтерфейсу має бути ітеративним процесом, який враховує відгуки користувачів та нові тенденції в області критичного мислення та взаємодії людини з комп'ютером. Це забезпечує постійне вдосконалення.

Дотримання цих особливостей під час розробки веб-додатків сприяє створенню інтерфейсів, які ефективно підтримують розвиток критичного мислення у користувачів.

1.2 Порівняльний аналіз аналогів

Розглянемо програмні додатки, які спрямовані на розвиток логічного мислення.

Elevate [3] - це мобільний додаток, розроблений для тренування та вдосконалення когнітивних навичок користувачів. Основною його метою є всебічний розвиток розумових здібностей людини.

Ключовою особливістю Elevate є набір персоналізованих вправ та завдань, спрямованих на тренування уваги, пам'яті, швидкості обробки інформації, математичних здібностей, словникового запасу та інших ментальних функцій. Програма адаптує складність вправ під рівень користувача, забезпечуючи поступовий і ефективний розвиток навичок.

Elevate заохочує до щоденних тренувань за допомогою ігрового підходу, який робить процес навчання більш захопливим і мотивуючим. Додаток також дає змогу відстежувати особистий прогрес, аналізувати статистику та отримувати поради експертів для поглиблення знань. Загалом, Elevate спрямований на комплексне стимулювання когнітивного розвитку користувачів, підвищуючи їхню продуктивність, пам'ять та інтелектуальні здібності.

Інтерфейс Elevate зображено на рисунку 1.1.

Рисунок 1.1 – Elevate

Briangle [4] - це інноваційна стартап-компанія, яка спеціалізується на розробці передових рішень у сфері штучного інтелекту та машинного навчання. Основною метою Briangle є створення технологій, здатних вирішувати складні завдання та надавати переваги як бізнесу, так і суспільству в цілому.

Ключовою розробкою Briangle є унікальна платформа для створення та навчання штучних нейронних мереж. Ця платформа дозволяє користувачам, навіть без глибоких знань у галузі комп'ютерних наук, швидко та ефективно створювати моделі ШІ для широкого спектру застосувань. Briangle пропонує інтуїтивний інтерфейс, який значно спрощує процес розробки та оптимізації нейронних мереж.

Окрім базової платформи, Briangle також розробляє спеціалізовані рішення на основі ШІ для різних галузей, таких як охорона здоров'я, фінанси, виробництво та маркетинг. Ці рішення використовують передові алгоритми глибокого навчання та комп'ютерного зору для автоматизації процесів, підвищення ефективності та прийняття більш обґрунтованих рішень.

Briangle особливо пишається своєю соціально-орієнтованою діяльністю. Компанія активно співпрацює з некомерційними організаціями, використовуючи свої технології для вирішення актуальних суспільних проблем, таких як подолання бідності, охорона довкілля та забезпечення рівних можливостей. Briangle демонструє, як інновації в галузі ШІ можуть принести користь суспільству в цілому.

Загалом, Briangle зарекомендувала себе як один з лідерів у сфері штучного інтелекту, пропонуючи передові рішення та технології, які трансформують бізнес-процеси та сприяють сталому суспільному розвитку.

Зображення Briangle подано на рисунку 1.2.

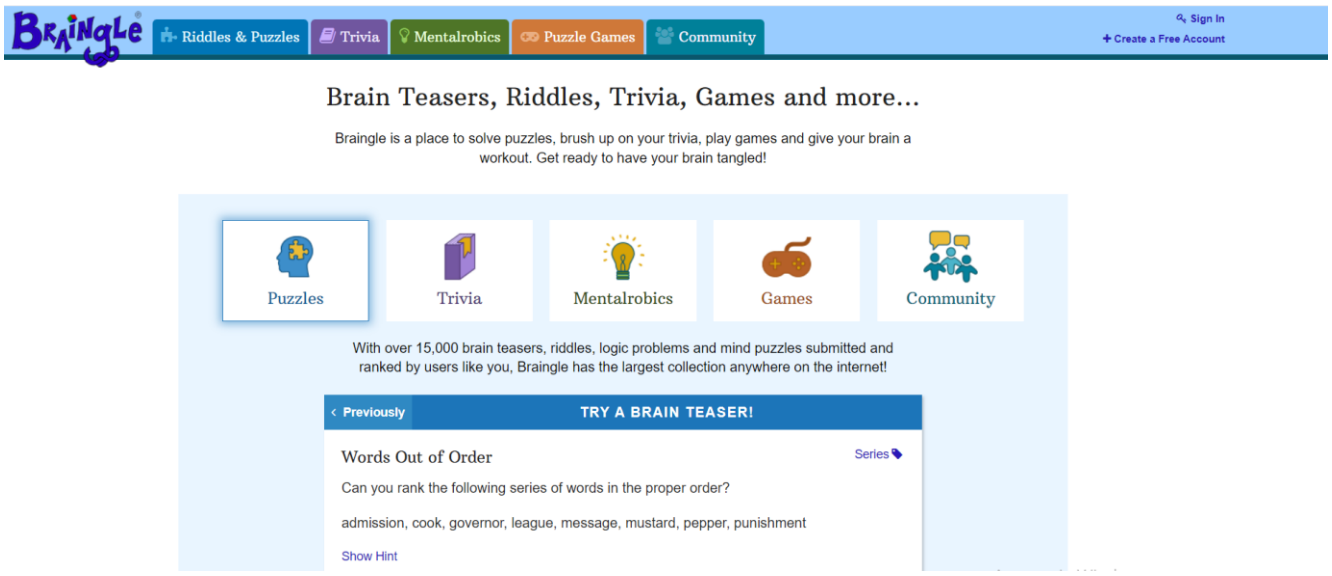


Рисунок 1.2 – Briangle

Cognito [5] - це мобільний додаток, розроблений з метою комплексного тренування когнітивних здібностей користувачів. Ця програма спрямована на стимулювання розумового розвитку людини шляхом виконання різноманітних

інтелектуальних вправ.

Ключовою особливістю Cognito є широкий спектр тренувальних вправ, які охоплюють такі ментальні функції, як увага, пам'ять, швидкість обробки інформації, логіка, вербальні здібності та математичні навички. Програма використовує науково-обґрунтовані методи, щоб забезпечити адаптивні та персоналізовані тренування, які відповідають індивідуальним потребам і рівню кожного користувача.

Додаток заохочує до регулярних щоденних тренувань, надаючи користувачам можливість спостерігати за своїм прогресом та досягненнями. Це допомагає підтримувати мотивацію та забезпечує відчуття постійного розвитку. Окрім того, Cognito пропонує різноманітні режими гри, які роблять тренування більш захоплюючими та приємними.

Крім того, програма забезпечує доступ до навчальних матеріалів, порад від експертів та інформативної статистики, що дозволяє користувачам глибше зрозуміти принципи роботи своїх когнітивних функцій та ефективніше працювати над їх вдосконаленням. Таким чином, Cognito допомагає людям покращувати свої розумові здібності, підвищувати продуктивність та підтримувати гострий розум протягом усього життя.

Зображення Cognito подано на рисунку 1.3.

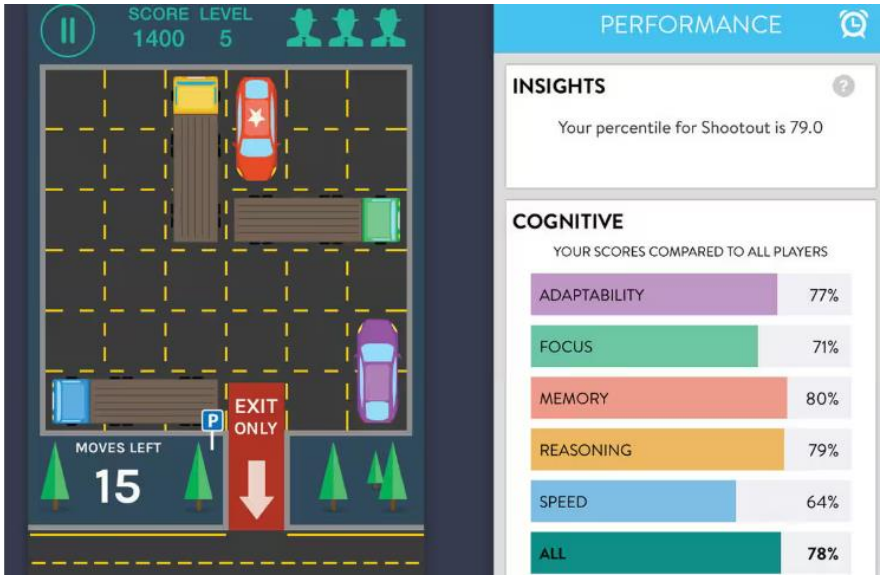


Рисунок 1.3 – Cognito

Для демонстрації відмінностей між цими додатками, була створена порівняльна таблиця (таблиця 1.1).

Таблиця 1.1 — Порівняння характеристик аналогів

	CogniFit	Lumosity	Peak	Власна розробка
Різні рівні складності гри	+	+	+	+
Адаптивність для різних видів пристроїв	+	-	+	+
Можливість переглядати зіграні ігри	-	+	-	+
Великий перелік ігор для розвитку	+	+	+	-
Наявність різних кольорових режимів інтерфейсу	-	-	-	+
Сумарний коефіцієнт	3	3	3	4

Таким чином, власна розробка має найвищий сумарний коефіцієнт порівняно з аналогами. Отже, власна розробка є актуальною.

1.3 Аналіз методів розв’язання задачі

Реалізація інтерфейсу користувача може бути виконана кількома способами, кожен з яких має свої переваги та недоліки.

Використання фреймворків, таких як React, Angular чи Vue.js, дозволяє

швидко створювати прототипи та розробляти користувацький інтерфейс. Ці фреймворки надають вбудовану підтримку для основних компонентів UI, а також пропонують стандартизовану структуру та архітектуру додатку. Однак, фреймворки додають додаткову складність через абстракції, а також можуть призвести до збільшення розміру додатку через включені бібліотеки. Можлива також тісна прив'язка до конкретного фреймворку.

Нативний метод, наприклад, використання UIKit/SwiftUI на iOS або Android Views на Android, забезпечує повний контроль над інтерфейсом користувача та оптимальну продуктивність. Цей підхід дозволяє використовувати низькорівневі API платформи. Проте, нативна розробка вимагає більших витрат часу та необхідності підтримувати окремі реалізації для кожної платформи.

Кросплатформні рішення, такі як Xamarin, React Native чи Electron, дозволяють створювати один код, який працює на різних платформах. Це прискорює розробку та зменшує витрати на підтримку, однак може призвести до компромісів у продуктивності або в Native Look and Feel. Також потрібно враховувати обмеження у доступі до низькорівневих API платформи.

На основі цього аналізу, було прийнято рішення розробляти інтерфейс застосунку з використанням кросплатформних рішень. Вони дозволяють розробити інтерфейс для різних видів пристроїв та легко провести адаптацію про них. Це сприятиме тому, що аудиторія користувачів буде швидко зростати, так як будуть задоволені потреби різних категорій користувачів різних видів пристроїв.

1.4 Постановка задач дослідження

Згідно з проведеним аналізом аналогів, аналізом стану питання розробки інтерфейсу застосунка та аналізом методів розв’язання задачі, було поставлено такі задачі дослідження:

- провести аналіз стану питання проектування та візуалізації інтерфейсу;
- розробити модель застосунку;
- розробити структуру клієнтської частини веб-застосунку;

- розробити алгоритм роботи програми;

- провести варіантний аналіз та обґрунтувати вибір мови програмування, середовища розробки та засобів реалізації інтерфейсу застосунку;

- розробити інтерфейс для реєстрації та авторизації користувача;

- розробити профіль користувача;

- розробити логіку перегляду зіграних ігор;

- розробити інтерфейс для гри;

- забезпечити адаптивність застосунку для різних видів пристроїв;

- провести тестування.

Технічне завдання на розробку наведено в додатку А.

1.5 Висновки

В цьому розділі, було проведено аналіз стану питання розробки інтерфейсу користувача для застосунку для розвитку логічного мислення, проведено порівняльний аналіз аналогів та встановлено актуальність власної розробки. Проведено аналіз методів розв’язання задачі. На основі цього, було поставлено задачі дослідження розробки.

2 РОЗРОБКА МОДЕЛІ, АЛГОРИТМІВ ТА СТРУКТУРНИХ СХЕМ ІНТЕРФЕЙСУ СИСТЕМИ

2.1 Розробка моделі застосунку

Розробка моделі програмної системи є важливим етапом у проектуванні та впровадженні складних програмних систем. Модель дозволяє розробникам та зацікавленим сторонам отримати чітке уявлення про структуру, поведінку та взаємодію різних компонентів системи.

Створення моделі системи є ключовим для аналізу та проектування. На основі моделі можна оцінювати вимоги до системи, проектувати її архітектуру, визначати потоки даних та розподіл функцій. Це дозволяє виявляти та усувати проблеми на ранніх стадіях розробки, коли вартість їх виправлення набагато нижча.

Модель програмної системи також відіграє важливу роль у тестуванні та валідації. Вона дає можливість створювати тестові сценарії, проводити симуляції та перевіряти відповідність системи встановленим вимогам. Це значно підвищує якість кінцевого продукту. Крім того, модель є важливим документом, який покращує розуміння системи серед розробників, менеджерів та користувачів, а також полегшує її подальшу підтримку та супровід.

Було розроблено модель web-застосунку для розвитку логічного мислення. Його зображення подано на рисунку 2.1.

Основою системи є web-клієнт, який складається з кількох ключових компонентів. Компонент "Firebase Authentication"[6] відповідає за процес автентифікації користувачів, забезпечуючи надійну і безпечну перевірку їхніх облікових даних. Компонент "React-router" керує клієнтською маршрутизацією, дозволяючи користувачам переміщуватися між різними сторінками додатка без необхідності перезавантаження сторінки.

Стилізація додатка реалізована за допомогою "CSS-modules", які дають можливість інкапсулювати стилі та легко підтримувати їх у масштабованому

проекті. Сторінка "Authorization" відповідає за процес авторизації користувачів, забезпечуючи контрольований доступ до функціональності системи. "Сторінка навігації" відіграє важливу роль, надаючи користувачам зручні засоби для навігації та взаємодії з різними частинами додатка.

Ключовою є "Main page", яка представляє основний вміст і функціональність системи. Крім того, "Ігрова сторінка" надає користувачам можливість взаємодіяти з ігровими механіками, що є важливою частиною загального досвіду користувача.

"Cloud Storage" відповідає за зберігання та управління даними програми. Компонент "Firebase Realtime Database"[7] забезпечує надійне сховище для даних додатка, дозволяючи синхронізувати та координувати інформацію між клієнтами. Компонент "Firebase Cloud Storage" відповідає за зберігання файлів, таких як зображення або інші мультимедійні дані, які використовуються в системі.

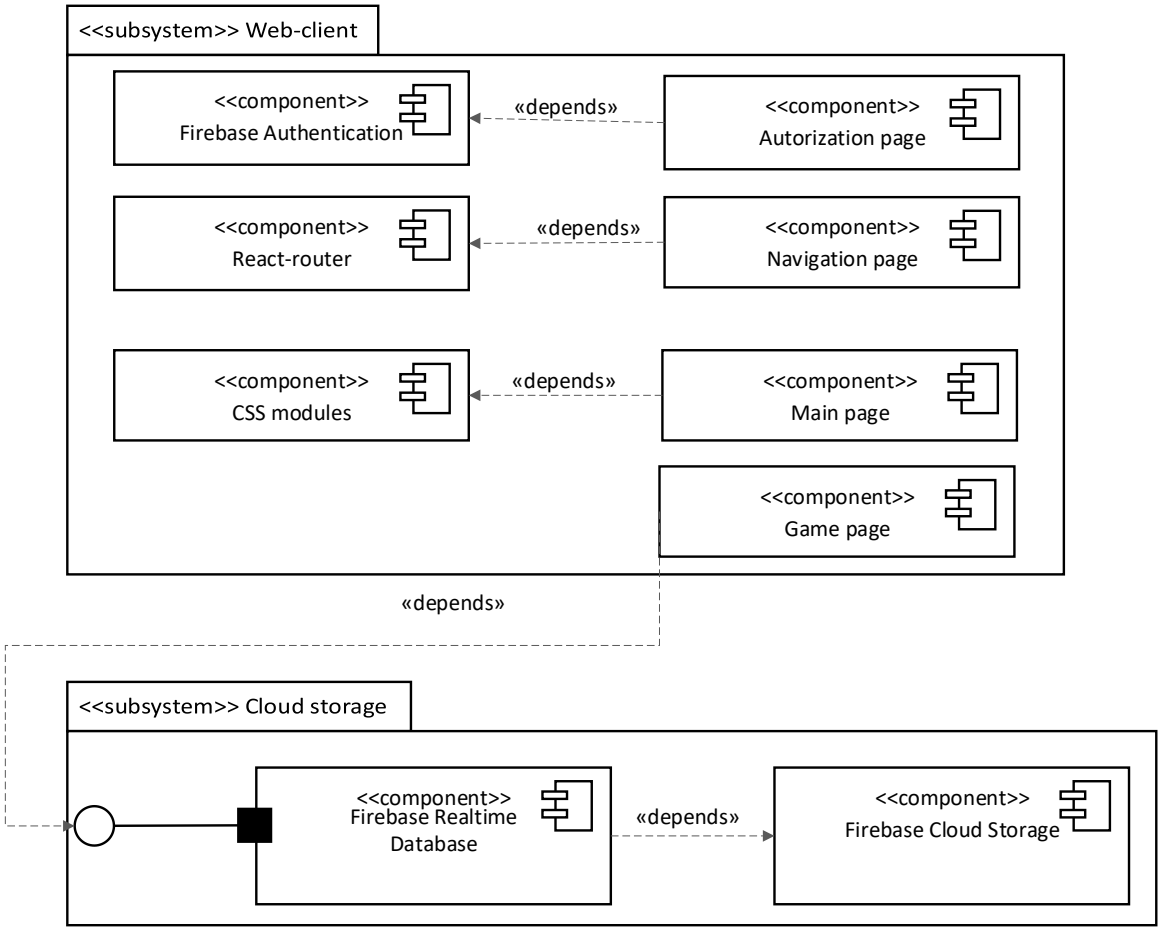


Рисунок 2.1 – Модель застосунку

2.2 Розробка структури клієнтської частини веб-додатку

Розробка структури клієнтської частини веб-додатку полегшує розуміння розробленого коду застосунку. Добре продумана структура підвищує читання коду та полегшує процес подальшого розширення функціоналу та розвитку застосунку.

Було розроблено файлову структуру клієнтської частини додатку, яка продемонстрована на рисунку 2.2.

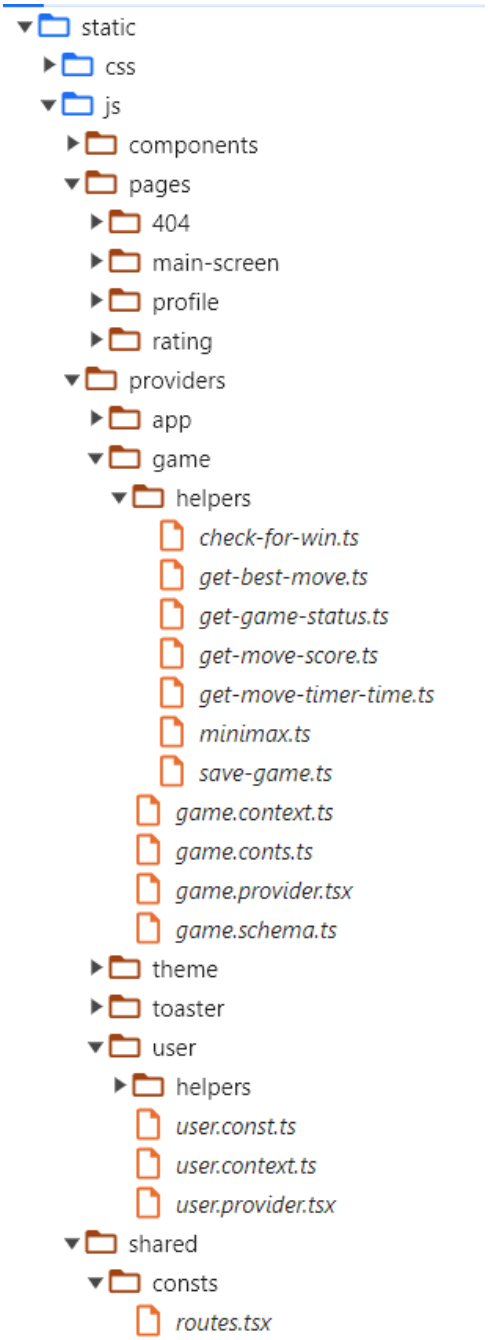


Рисунок 2.2 – Структура клієнтської частини додатку

Розглянемо детальніше основні папки та файли:

- css/ - CSS стилі для додатку;

- js/components - React компоненти;

- js/pages - сторінки додатку, такі як 404, main-screen, profile, rating;

- app/ - головний провайдер додатку;

- game/ - провайдер, відповідальний за ігрову логіку;

- helpers/ - різні допоміжні функції для гри (перевірка виграшу, отримання статусу, обчислення балів тощо);

- js/theme/ - компоненти, пов'язані з темою/оформленням додатку;

- js/user/helpers - допоміжні функції, пов'язані з користувачами;

- js/shared/consts.tsx - спільні константи для усього додатку;

Ця структура відповідає сучасному WEB-додатку з відокремленням фронтенду та бекенду, де фронтенд побудований з React компонентів та сторінок, а бекенд містить логіку провайдерів та допоміжні функції.

2.3 Розробка алгоритмів роботи програми

Алгоритми роботи розробляються для кращого проектування взаємодії вікон інтерфейсів користувача. Вони дозволяють краще спланувати взаємодію елементів інтерфейсу та цим покращити саму розробку, зручність її використання та загальне враження користувачів.

Було розроблено алгоритм авторизації користувача. Блок-схема алгоритму наведена на рисунку 2.3.

Алгоритм описує процес авторизації, починаючи від вибору способу авторизації до перевірки введених даних та передбачення різних сценаріїв залежно від результату перевірки.

Розроблено також алгоритм перегляду гри. Він працює по принципу зв’язаного списку, в якому можна рухатися з обраного елемента лише на попередній або наступний (рисунок 2.4). Перегляд гри дозволяє перемикатися між ходами у грі так, як вони були зіграні. Це дозволяє аналізувати гру, виявляти

помилки та покращувати тактику гри.

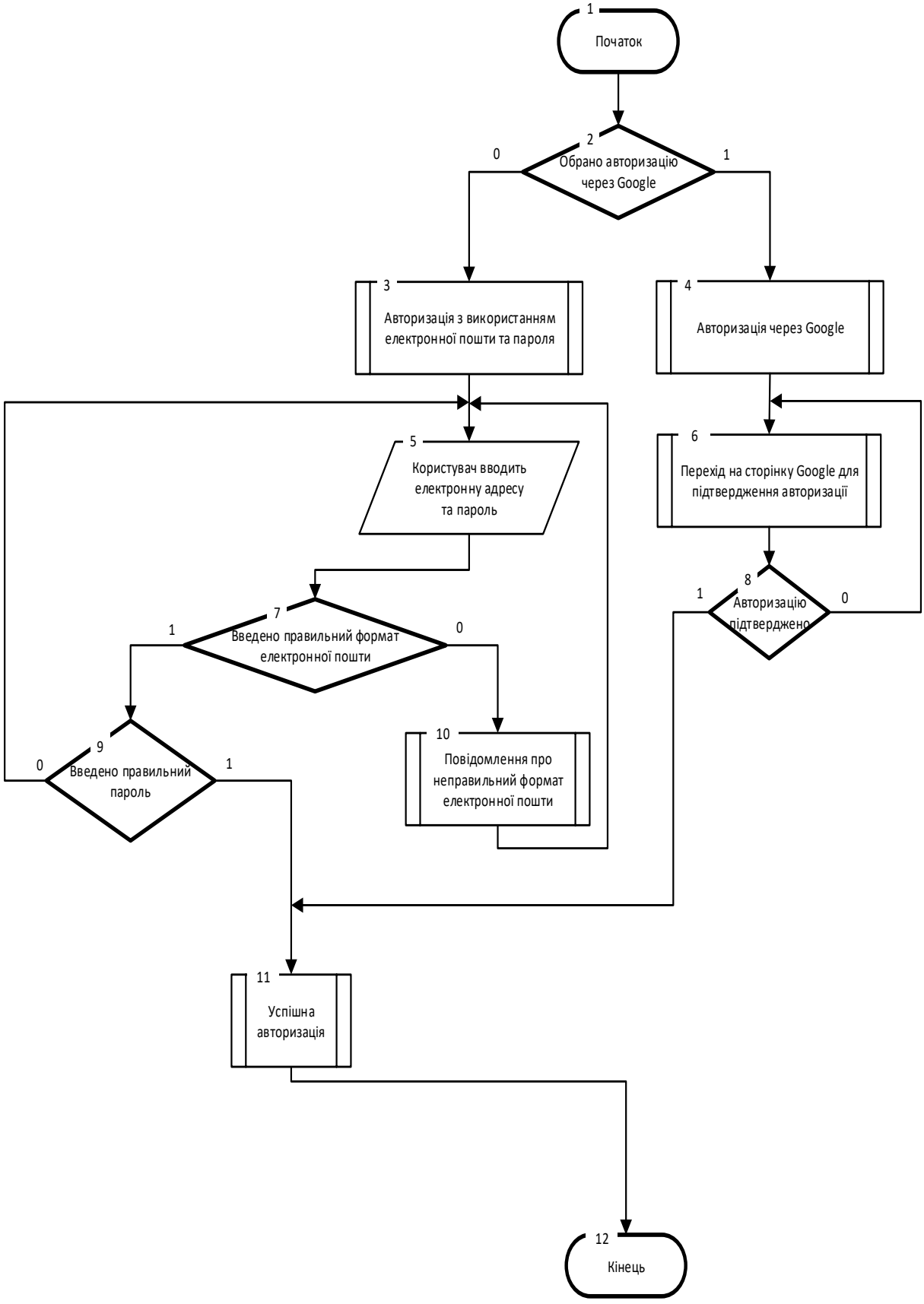


Рисунок 2.3 – Блок-схема алгоритму авторизації користувача

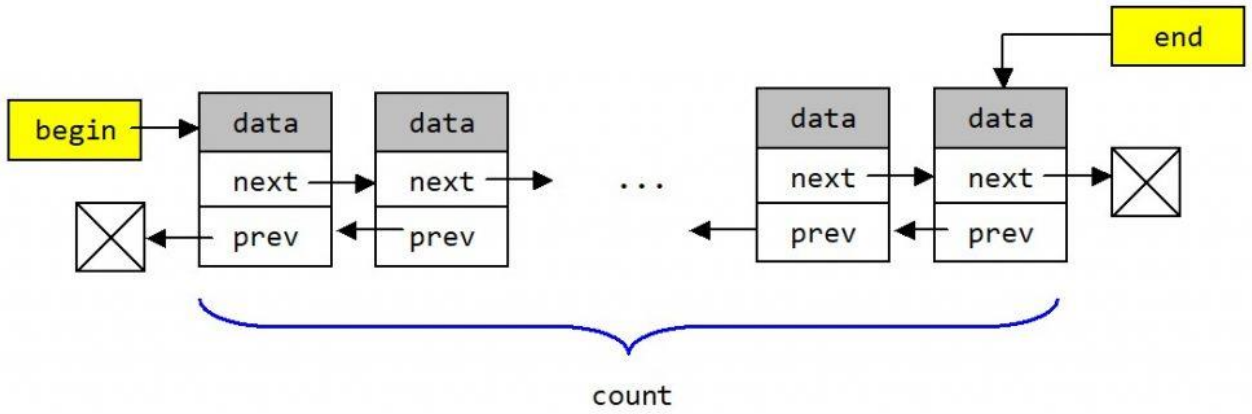


Рисунок 2.4 – Зв’язний список

Блок-схему алгоритму перегляду гри наведено на рисунку 2.5.

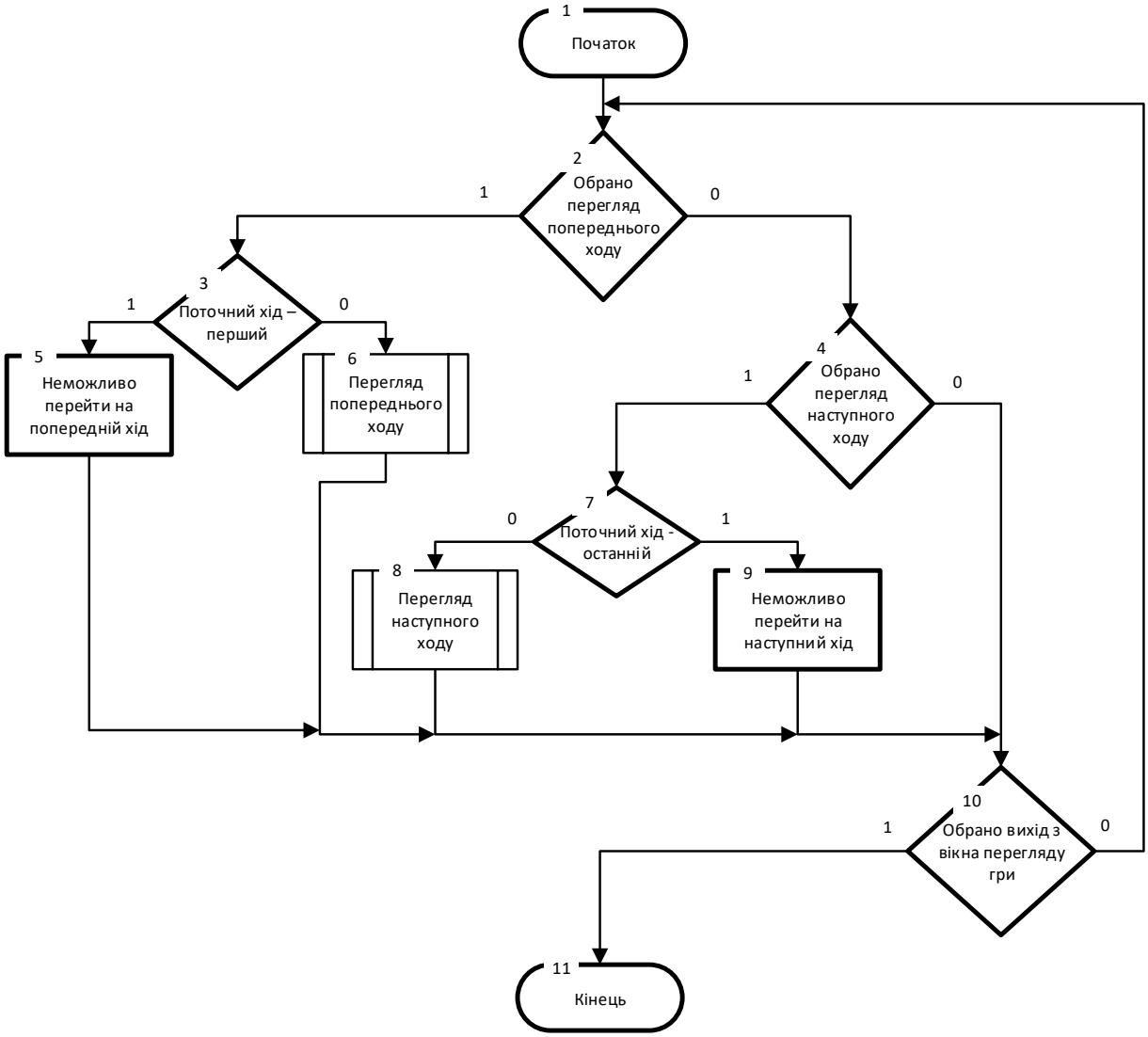


Рисунок 2.5 – Блок-схема алгоритму перегляду гри

2.4 Висновки

В цьому розділі було розроблено модель застосунку, в якій описані складові її компоненти. Було розроблено алгоритми роботи елементів графічного інтерфейсу застосунку та програми загалом, розроблено структурні схеми інтерфейсу застосунку.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Варіантний аналіз та обґрунтування вибору мови програмування

Для реалізації інтерфейсу користувача використовуються такі мови програмування та розмітки.

HTML [8] (Hypertext Markup Language) є стандартною мовою розмітки, яка використовується для створення структури та змісту веб-сторінок. Синтаксис HTML полягає в тому, що документ складається з елементів, які позначаються тегами, наприклад, `<h1>Заголовок</h1>` або `<p>Абзац тексту</p>`. Теги зазвичай містять назву, обрамлену кутовими дужками `<>`, і можуть мати атрибути, які надають додаткову інформацію про елемент, наприклад, `<img src="image.jpg" alt="Зображення">`.

Структура HTML-документа має базову основу з елементами `<html>`, `<head>` та `<body>`. У секції `<head>` міститься метаінформація про сторінку, така як заголовок, посилання на CSS та JavaScript, а секція `<body>` містить фактичний вміст сторінки - заголовки, абзаци, зображення, посилання тощо.

Семантика HTML-тегів має важливе значення, оскільки вони вказують на призначення елемента, наприклад, `<h1>`, `<p>`, `<nav>`, `<article>`. Семантична розмітка покращує доступність сайту та його сприйняття пошуковими системами. HTML дозволяє вставляти на веб-сторінки широкий спектр мультимедійного вмісту, такого як зображення, відео, аудіо, форми тощо. Нові версії HTML (HTML5) розширюють можливості, додаючи теги для семантичної розмітки, offline-можливості, покращену підтримку мультимедіа та інше.

PHP [9] (Hypertext Preprocessor) - це серверна мова програмування, яку широко використовують для розробки динамічних веб-сайтів і веб-додатків.

PHP є інтерпретованою мовою, що означає, що код виконується на сервері, на відміну від клієнтських мов, таких як JavaScript, які виконуються на стороні користувача. Це дозволяє PHP обробляти дані, взаємодіяти з базами даних, генерувати динамічний вміст і повертати результати клієнтському браузеру у

вигляді HTML-сторінок.

Одна з основних особливостей PHP - це його інтеграція з великою кількістю баз даних, включаючи MySQL, PostgreSQL, Oracle та інші. Це дає можливість створювати складні веб-додатки, які зберігають і обробляють дані, такі як інформація про користувачів, товари в інтернет-магазині, публікації в блозі тощо.

PHP також підтримує об'єктно-орієнтоване програмування, шаблони проектування, модульність коду та інші сучасні парадигми програмування. Це дозволяє розробникам створювати масштабовані, структуровані та підтримувані веб-додатки. Крім того, PHP має великий та активний екосистему бібліотек та фреймворків, таких як Laravel, Symfony, Drupal, WordPress, які значно спрощують та прискорюють розробку.

PHP - це потужна та гнучка мова програмування, яка добре підходить для створення різноманітних веб-додатків, від простих статичних сайтів до складних динамічних платформ. Її широке використання та підтримка великої кількості бібліотек і фреймворків роблять її привабливим вибором для веб-розробників.

TypeScript [10] - це мова програмування, яка побудована на основі JavaScript та розширює його можливості. Основна відмінність TypeScript від JavaScript полягає в тому, що TypeScript є статично типізованою мовою, на відміну від динамічної типізації JavaScript.

Статична типізація означає, що змінні в TypeScript мають чітко визначений тип даних, який встановлюється під час компіляції коду. Це дозволяє краще виявляти помилки на етапі розробки та надає потужну підтримку від IDE. Розробники можуть чітко описувати структуру даних, функцій та об'єктів, що робить код більш читабельним та масштабованим.

TypeScript також додає до JavaScript низку інших можливостей, таких як класи, інтерфейси, модулі, декоратори та новітні функції мови ECMAScript. Це дозволяє використовувати сучасні парадигми об'єктно-орієнтованого програмування та функціонального програмування при розробці веб-додатків.

Ще однією важливою перевагою TypeScript є його сумісність з JavaScript. Будь-який JavaScript-код є також валідним TypeScript-кодом, тому розробники

можуть поступово мігрувати існуючі проекти на TypeScript, використовуючи переваги статичної типізації. TypeScript також має потужну екосистему бібліотек та фреймворків, які надають готові типізовані визначення для популярних JavaScript-залежностей.

TypeScript - це мова, яка значно покращує розробку та підтримку великих веб-додатків, надаючи статичну типізацію, сучасні можливості мови та покращене інтегрування з IDE. Це робить TypeScript привабливою альтернативою JavaScript для команд, які прагнуть до більш структурованого, надійного та масштабованого коду.

Сформуємо таблицю порівняння характеристик цих мов для реалізації клієнтської частини системи.

Таблиця 3.1 – Таблиця порівняння мов програмування

Характеристики	HTML	PHP	TypeScript
Об’єктно-орієнтоване програмування	-	+	+
Функціональне програмування	-	-	+
Типізація	-	-	+
Розробка великих проектів	-	+/-	+
Інтеграція з фреймворками	-	-	+
Простота вивчення	+	+/-	+/-
Кросплатформність	+	-	+
Загальна оцінка	2	3	6.5

Якщо порівнювати ці мови між собою, то можна зробити висновок, що HTML не підходить для розробки інтерфейсу додатку для розвитку критичного мислення через те, що має негнучкий набір інструментів для розробки сучасних інтерфейсів, вимагає великих затрат часу на розробку інтерфейсу користувача.

Якщо ж порівнювати між собою PHP та TypeScript, то TypeScript надає статичну типізацію, що означає, що типи даних визначаються під час компіляції. Це дозволяє виявляти помилки на ранніх етапах розробки та надає кращу

підтримку з боку IDE. Статична типізація покращує читабельність коду, робить його більш структурованим та масштабованим, особливо для великих проектів. На відміну від цього, PHP - це динамічно типізована мова, що може створювати більше помилок під час виконання.

TypeScript має відмінну підтримку від популярних IDE, таких як Visual Studio Code, WebStorm, IntelliJ IDEA, що значно покращує продуктивність розробників. Екосистема TypeScript включає величезну кількість бібліотек, фреймворків та інструментів, оптимізованих для статичної типізації. Це дозволяє швидше створювати та підтримувати користувацькі інтерфейси, використовуючи перевірені, типізовані рішення.

Враховуючи вище наведені аргументи, можна зробити висновок, що TypeScript є найкращим вибором для розробки графічного інтерфейсу користувача веб-застосунку для розвитку логічного мислення.

3.2 Вибір середовища розробки

Існує великий перелік середовищ розробки, що спеціалізовані на веб-програмуванні та зокрема на розробці веб-інтерфейсів. Проведемо аналіз деяких із них.

Sublime Text [11] - це потужний, швидкий та гнучкий текстовий редактор, який став одним із найпопулярніших середовищ розробки серед веб-програмістів. Він вирізняється своєю високою продуктивністю та низьким споживанням системних ресурсів, що дозволяє ефективно працювати навіть з великими кодовими базами. Редактор підтримує безліч мов програмування, розмітки та скриптових мов, включаючи JavaScript, HTML, CSS, PHP, Python, Ruby та інші, забезпечуючи розробникам комфортну роботу з широким спектром технологій.

Одним із ключових переваг Sublime Text є його розширюваність. Редактор має великий арсенал вбудованих функцій, а його можливості можуть бути значно розширені за допомогою додаткових пакетів та плагінів, створених активною спільнотою розробників. Це дозволяє налаштовувати редактор відповідно до

індивідуальних потреб та вподобань кожного програміста, підвищуючи його ефективність та зручність використання.

Sublime Text також пропонує ефективні інструменти для навігації по коду, такі як панель проекту, швидкий пошук, переміщення по файлах та інші. Ці можливості значно спрощують та прискорюють роботу веб-розробників, особливо при роботі над великими проєктами. Редактор також підтримує інтеграцію з популярними системами контролю версій, що дозволяє легко виконувати основні операції, пов'язані з управлінням кодом.

Visual Studio Code [12] - це одне з найпопулярніших та найбільш функціональних середовищ розробки, яке користується широкою популярністю серед веб-розробників. Це безкоштовний та універсальний редактор коду, який підтримує велику кількість мов програмування, включаючи JavaScript, TypeScript, HTML, CSS та інші технології, необхідні для веб-розробки.

Основною перевагою Visual Studio Code є його розширюваність та інтегрованість. Редактор має величезну екосистему плагінів та розширень, створених спільнотою розробників, які дозволяють адаптувати його функціональність під конкретні потреби кожного програміста. Від додаткових інструментів для налагодження коду до інтеграції з системами контролю версій, Visual Studio Code забезпечує широкий спектр можливостей для підвищення продуктивності веб-розробки.

Ще однією важливою особливістю Visual Studio Code є його висока швидкість та ефективність. Редактор відрізняється низьким споживанням системних ресурсів, що дозволяє йому швидко завантажуватись та працювати навіть з великими проєктами. Це робить Visual Studio Code ідеальним вибором для веб-розробників, які цінують продуктивність та зручність використання.

Крім того, Visual Studio Code надає потужні інструменти для навігації, рефакторингу та відлагодження коду. Вбудовані засоби, такі як підтримка IntelliSense, автоматичне завершення коду та інтерактивна дебаг-консоль, значно спрощують і прискорюють процес розробки веб-застосунків. Редактор також надає можливість інтеграції з основними системами контролю версій, що дозволяє

ефективно керувати змінами в коді.

IntelliJ IDEA [13] – це потужне середовище розробки від компанії JetBrains, яке вважається одним з найефективніших інструментів для веб-програмування на Java-орієнтованих технологіях.

Однією з ключових переваг IntelliJ IDEA є його глибока інтеграція з Java-екосистемою. Редактор надає розвинені можливості для роботи з фреймворками та бібліотеками, такими як Spring, Angular, React тощо. Це дозволяє веб-розробникам, які використовують Java-стек, ефективно працювати з усіма необхідними компонентами, не виходячи за межі єдиного інтегрованого середовища.

Крім того, IntelliJ IDEA відрізняється своєю надзвичайною продуктивністю. Потужний редактор коду, вдосконалені засоби для навігації, рефакторингу та відлагодження роблять процес розробки надзвичайно ефективним. Інтелектуальні можливості автодоповнення, швидкий пошук та контекстно-залежні підказки значно прискорюють написання та редагування коду.

Ще однією перевагою IntelliJ IDEA є його розширюваність. Як і більшість середовищ розробки від JetBrains, цей редактор має велику екосистему плагінів та додаткових інструментів, створених як самою компанією, так і активною спільнотою розробників. Це дозволяє налаштувати IntelliJ IDEA під конкретні потреби кожного програміста, підвищуючи його ефективність та зручність використання.

Складемо таблицю порівняння цих середовищ розробки (таблиця 3.2).

Таблиця 3.2 – Порівняння середовищ розробки

Характеристика	Sublime Text	Visual Studio Code	IntelliJ IDEA
Розширюваність	+/-	+	+/-
Вбудовані інструменти для веб-розробки	+/-	+/-	+/-

Продовження таблиці 3.2.

Підтримка сучасних веб-фреймворків	+/-	+	+
Безкоштовність	+/-	+	+/-
Підтримка мов програмування для веб-розробки	+	+	+
Сумарний бал	2.5	4.5	3.5

Таким чином, Visual Studio Code має найвищий сумарний бал та є найкращим середовищем для розробки веб-інтерфейсу застосунку для розвитку логічного мислення.

3.3 Вибір засобів реалізації інтерфейсу програмного застосунку

Розглянемо та порівняємо популярні фреймворки для розробки користувацького веб-інтерфейсу застосунка.

Express.js [14] - це популярний веб-фреймворк для Node.js, який спрощує створення REST API та веб-застосунків. Він надає абстракцію над низькорівневим Node.js HTTP-сервером, дозволяючи розробникам швидко створювати маршрути, обробляти запити і відповіді, керувати middleware-модулями тощо.

Основна перевага Express.js - в його мінімалістичності та гнучкості. Він не нав'язує жорсткої структури проекту, а лише надає базову функціональність для побудови веб-серверів. Це дає розробникам свободу вибору архітектури, шаблонів проектування та інших компонентів, які найкраще підходять для їхнього конкретного додатка.

Express.js широко використовується для створення REST API, веб-сайтів, мобільних додатків, а також фонових процесів. Він має велику екосистему

плагінів та middleware-компонентів, які дозволяють розширювати функціональність базового фреймворку - від маршрутизації та автентифікації до логування та обробки статичних файлів.

Популярність Express.js обумовлена його простотою, продуктивністю та гнучкістю. Розробники можуть швидко створювати прототипи або повноцінні веб-застосунки з мінімальною кількістю коду. Водночас фреймворк надає достатньо механізмів для побудови складних, масштабованих систем.

Angular [15] - це потужний, повнофункціональний фреймворк для розробки односторінкових веб-застосунків (SPA). Він був розроблений компанією Google і швидко набув популярності серед розробників завдяки своїй комплексності та ефективності.

Основною перевагою Angular є його модульна архітектура. Додаток в Angular складається з окремих модулів, компонентів, сервісів та директив, які взаємодіють між собою. Це дозволяє будувати масштабовані, добре структуровані застосунки, які легко підтримувати та розширювати. Angular також надає вбудовану підтримку для таких можливостей, як маршрутизація, управління станом тощо.

Одна з ключових особливостей Angular - це використання TypeScript, надбудови над JavaScript, яка додає статичну типізацію, покращує підтримку IDE та загалом підвищує якість коду. TypeScript дозволяє писати більш надійний, безпечний та масштабований код, особливо в великих проектах.

Сильною стороною Angular є його потужна екосистема та інструментарій. До складу фреймворку входять вбудовані утиліти для тестування, збірки, розгортання та моніторингу додатків. Крім того, існує велика кількість сторонніх бібліотек та інструментів, які полегшують розробку Angular-застосунків.

React [16] - це бібліотека JavaScript для побудови інтерфейсів користувача. На відміну від комплексних фреймворків, таких як Angular, React фокусується виключно на представленні користувацького інтерфейсу, надаючи розробникам максимальну гнучкість у виборі архітектури та інших компонентів застосунку.

Ключовою концепцією React є компоненти - окремі частини інтерфейсу, які

можна багаторазово використовувати та комбінувати. Компоненти в React є самодостатніми, мають власний стан і логіку, що робить код модульним, повторно використовуваним та легшим для розуміння та підтримки.

Іншою важливою особливістю React є віртуальний DOM (Document Object Model). Замість прямого оновлення реального DOM, React створює свою власну, оптимізовану копію дерева DOM, а потім обчислює і застосовує мінімальні необхідні зміни. Це забезпечує високу продуктивність навіть при складних інтерфейсах.

Популярність React зумовлена його простотою, продуктивністю та гнучкістю. Хоча сам React - це лише бібліотека для відображення, він має велику та активну екосистему, яка включає маршрутизатори, системи управління станом, UI-бібліотеки та інші інструменти, необхідні для побудови сучасних веб-застосунків. Це дозволяє розробникам швидко створювати масштабовані, високопродуктивні рішення.

Оцінюючи переваги React порівняно з Angular та Express.js, можна виділити кілька ключових аргументів на користь React.

По-перше, React відрізняється високою гнучкістю та модульністю. На відміну від комплексного фреймворку Angular, React є бібліотекою, яка не нав'язує жорстку структуру додатка. Це дає розробникам свободу вибору архітектури, бібліотек та інших компонентів, що найкраще підходять для конкретного проекту. Водночас Express.js, хоч і є мінімалістичним фреймворком, все ж стосується лише серверної частини, тоді як React фокусується на клієнтському інтерфейсі.

Не менш вагомою перевагою React є його висока продуктивність. Завдяки використанню віртуального DOM, React здатен оптимізувати процес перерендерингу, забезпечуючи високу швидкість навіть для складних інтерфейсів. Хоча Angular та Express.js також мають механізми для оптимізації продуктивності, вони не настільки просунуті, як віртуальний DOM у React.

Іншим аргументом на користь React є його потужна екосистема та інструментарій. React має дуже велику та активну спільноту розробників, які створюють величезну кількість сторонніх бібліотек, фреймворків та інструментів,

що значно спрощують розробку. Хоча Angular також має розвинену екосистему, вона більш обмежена рамками самого фреймворку. Натомість Express.js, як мікрофреймворк, має дещо меншу екосистему.

React виграє й у питанні кросплатформності. Будучи бібліотекою TypeScript, React може використовуватись на будь-якій платформі, де працює TypeScript – веб та мобільні пристрої.

3.4 Розробка структурних схем інтерфейсу

Структурна схема інтерфейсу програмного застосунку є графічним представленням організації та взаємозв'язків між різними компонентами користувацького інтерфейсу. Вона допомагає розробникам та дизайнерам візуалізувати та планувати структуру та потік взаємодії користувача з програмним застосунком.

Основними елементами структурної схеми інтерфейсу є сторінки, екрани, вікна або модальні вікна, які представляють собою окремі функціональні блоки. Ці блоки пов'язані між собою за допомогою переходів, які визначають логіку навігації користувача. Переходи можуть бути ініційовані користувачем, наприклад, натисненням на кнопку, або можуть відбуватися автоматично, наприклад, при зміні стану застосунку.

Структурна схема також може включати в себе компоненти, такі як меню, панелі інструментів, форми, повідомлення про помилки тощо. Ці компоненти можуть бути повторно використані на різних сторінках або екранах, забезпечуючи послідовність та зручність користування застосунком.

Створення структурної схеми на ранніх етапах розробки програмного продукту допомагає команді розробників та дизайнерів узгодити концепцію інтерфейсу, виявити потенційні проблеми та прийняти обґрунтовані рішення щодо навігації, взаємодії та візуального оформлення. Це, у свою чергу, полегшує процес розробки та забезпечує послідовний та інтуїтивно зрозумілий досвід користувача.

Структурна схема інтерфейсу є важливим документом, який слугує основою

для подальшої розробки, тестування та вдосконалення програмного застосунку. Вона також може бути використана для комунікації з клієнтами, зацікавленими сторонами та іншими членами команди.

Було розроблено структурні схеми інтерфейсу ігрового застосунку для розвитку логічного мислення.

Структурну схему меню інструментів для не авторизованих користувачів наведено на рисунку 3.1.

Рисунок 3.1 – Структурна схему меню інструментів для не авторизованого користувача

Складові меню інструментів для не авторизованого користувача:

1. Логотип.
2. Кнопка «Авторизуватися».
3. Кнопка зміни кольорової теми застосунку.

Крім цього, розроблено структурну схему меню інструментів для авторизованих користувачів. Вона наведена на рисунку 3.2.

Рисунок 3.2 – Структурна схема меню інструментів для авторизованого користувача

Складові меню інструментів для авторизованого користувача:

1. Логотип.
2. Кнопка переходу до профілю.
3. Кнопка рейтингу.
4. Кнопка виходу з акаунту.

5. Кнопка зміни кольорової теми застосунку.

Структурну схему головного вікна застосунку наведено на рисунку 3.3. Це вікно відкривається при початковому вході в додаток. Воно містить в собі гру як для авторизованих, так і не авторизованих користувачів. Відмінність полягає в тому, що для неавторизованих користувачів не зберігається історія ігор та ігровий прогрес.

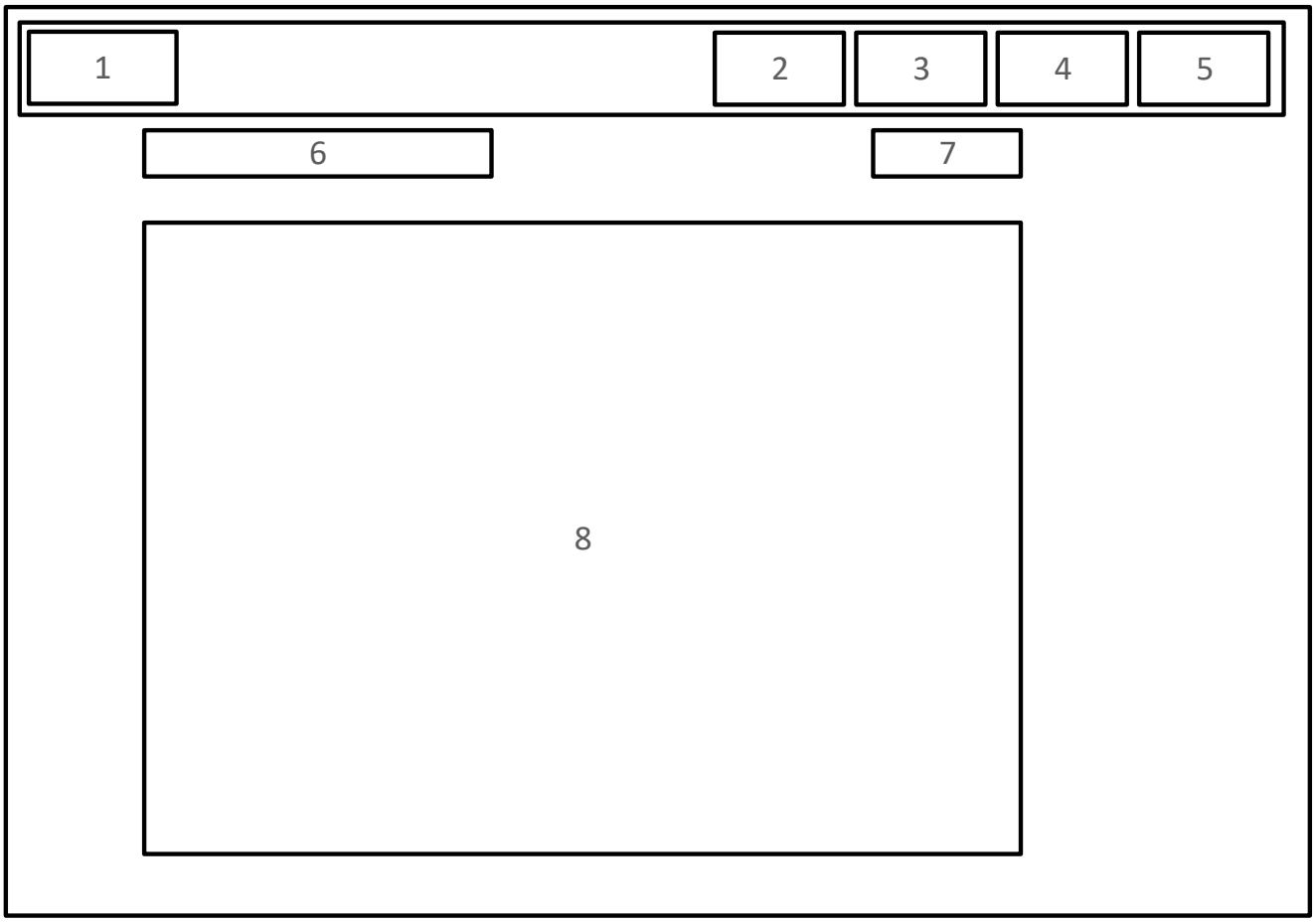


Рисунок 3.3 – Структурна схема головного вікна застосунку

Складові головного вікна:

1. Логотип.
2. Кнопка переходу до профілю.
3. Кнопка рейтингу.
4. Кнопка виходу з акаунту.
5. Кнопка зміни кольорової теми застосунку.

6. Панель зміни рівня складності гри.

7. Таймер.

8. Ігрова область.

Розроблено структурну схему вкладки профілю користувача. Вона дозволяє переглядати історію ігор, змінити ім’я користувача та переглянути місце користувача в рейтингу гравців. Її зображення наведено на рисунку 3.4.

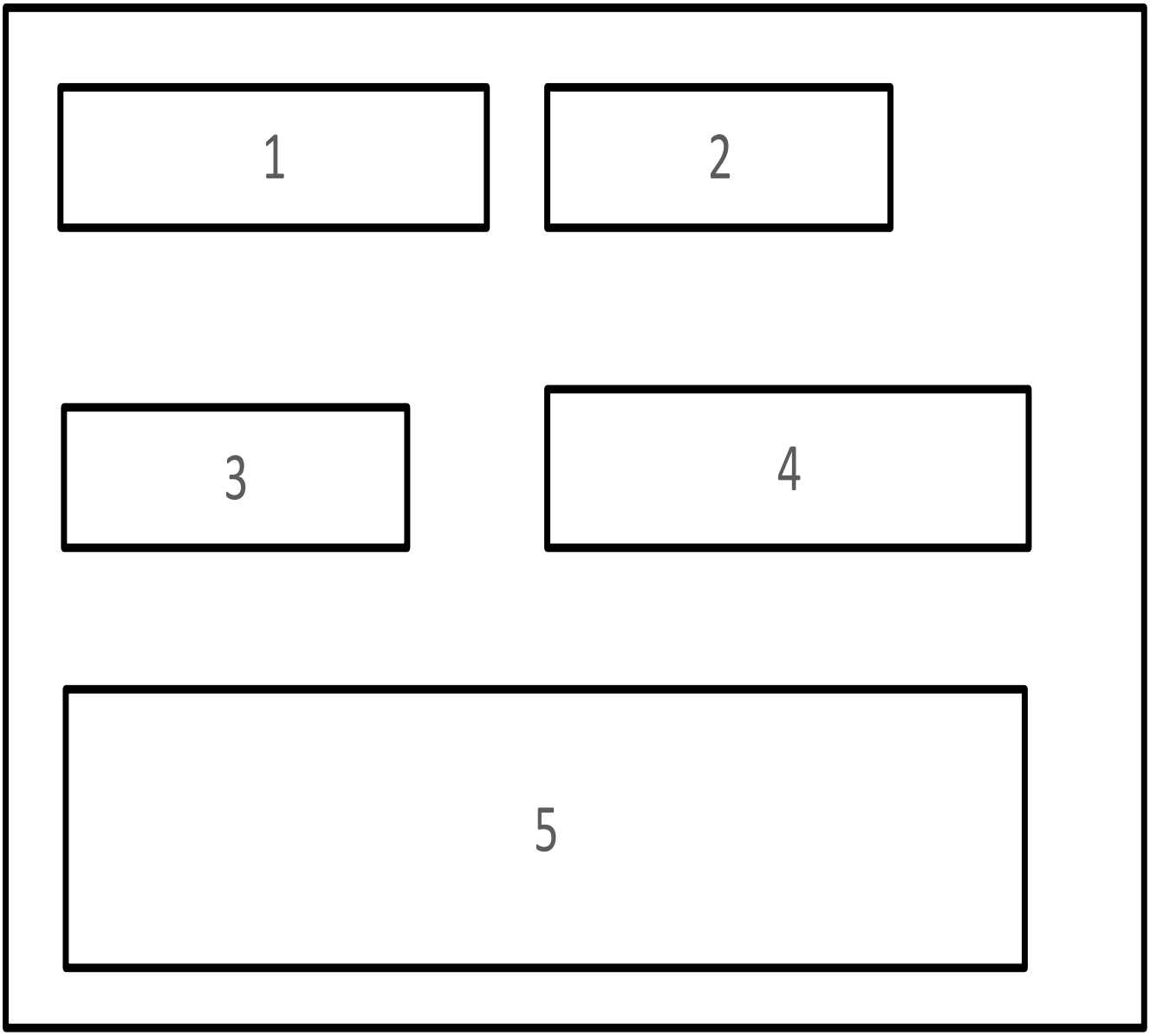


Рисунок 3.4 – Структурна схема профілю користувача

Складові вкладки профілю користувача:

1. Ім’я користувача

2. Місце користувача в рейтингу гравців.
3. Рейтинг користувача.
4. Кнопка зміни імені користувача.
5. Історія ігор.

При завершенні гри виводиться окреме вікно з інформацією про результат гри залежно від того, як зіграв користувач. Структурна схема вікна про завершення гри наведена на рисунку 3.5.

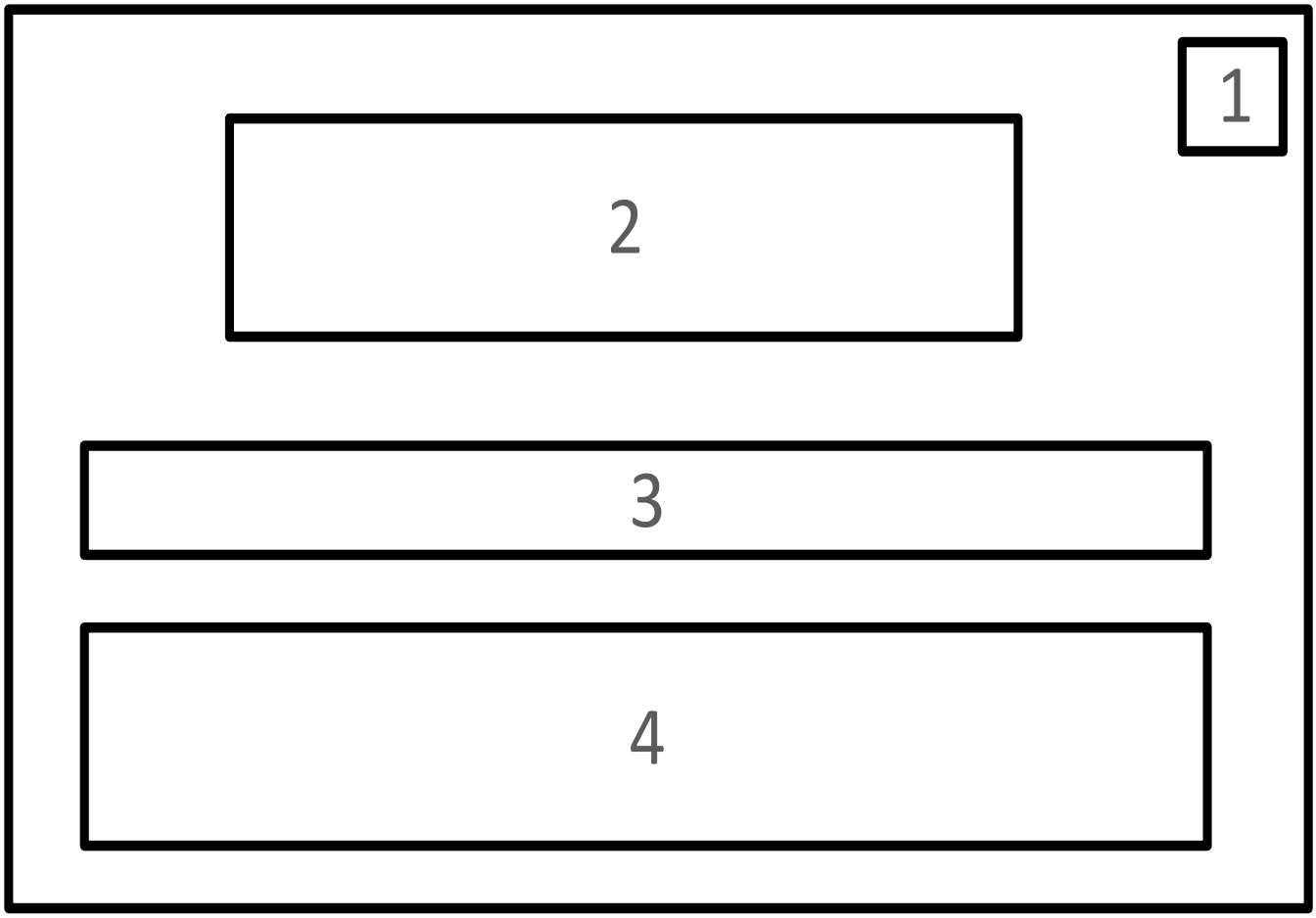


Рисунок 3.5 – Вікно інформації про завершення гри

Складові вікна інформації про завершення гри:

1. Кнопка закриття вікна.
2. Повідомлення про результат гри.
3. Зміна в рейтингу.
4. Кнопка авторизації для неавторизованих користувачів.

Вікно реєстрації та авторизації дозволяє користувачам отримати доступ до свого акаунту в застосунку, що дозволяє переглядати рейтинг гравців, власну історію ігор та зберігати свій ігровий прогрес. Структурна схема вікна реєстрації та авторизації наведена на рисунку 3.6.

Складові вікна реєстрації/авторизації:

1. Кнопка для закриття вікна.
2. Кнопка авторизації через Google.
3. Кнопка для режиму реєстрації.
4. Кнопка для режиму авторизації.
5. Поле введення електронної пошти.
6. Поле для паролю.
7. Кнопка для авторизації/реєстрації.

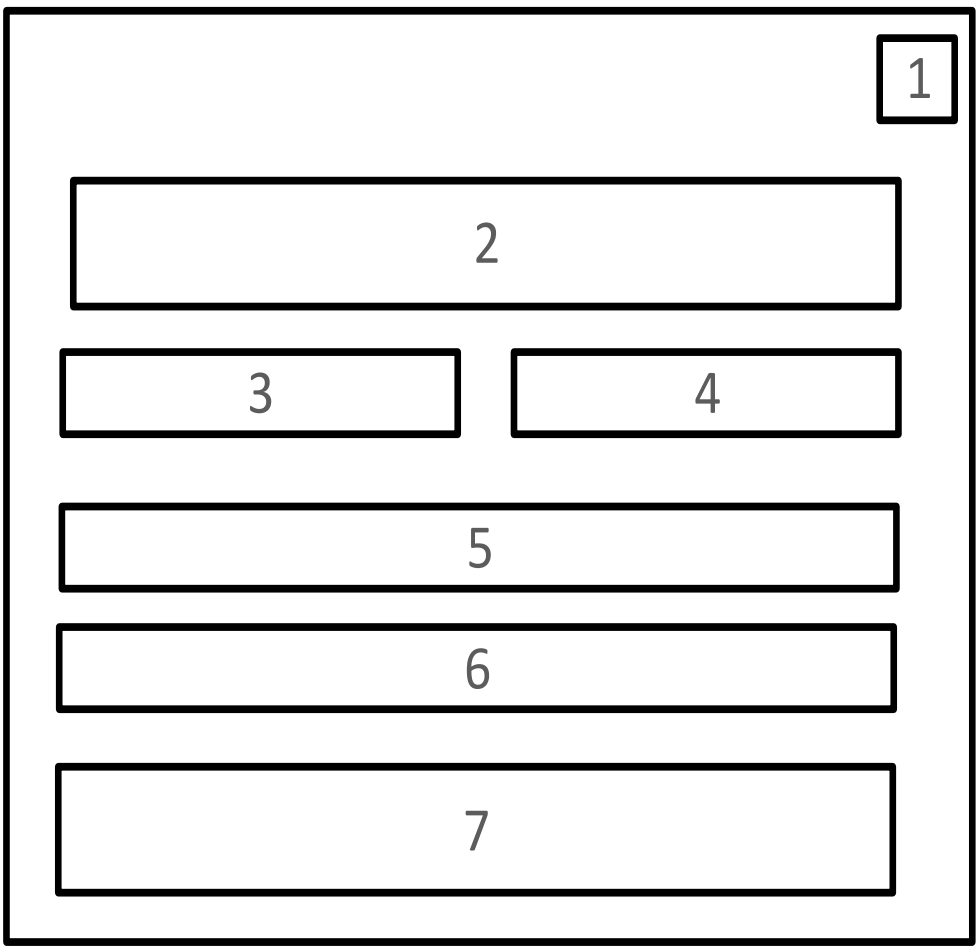


Рисунок 3.6 – Структурна схема вікна реєстрації/авторизації

Вікно рейтингу дозволяє переглядати рейтинг найуспішніших гравців та своє місце у ньому. Структурна схема цього вікна наведена на рисунку 3.7.

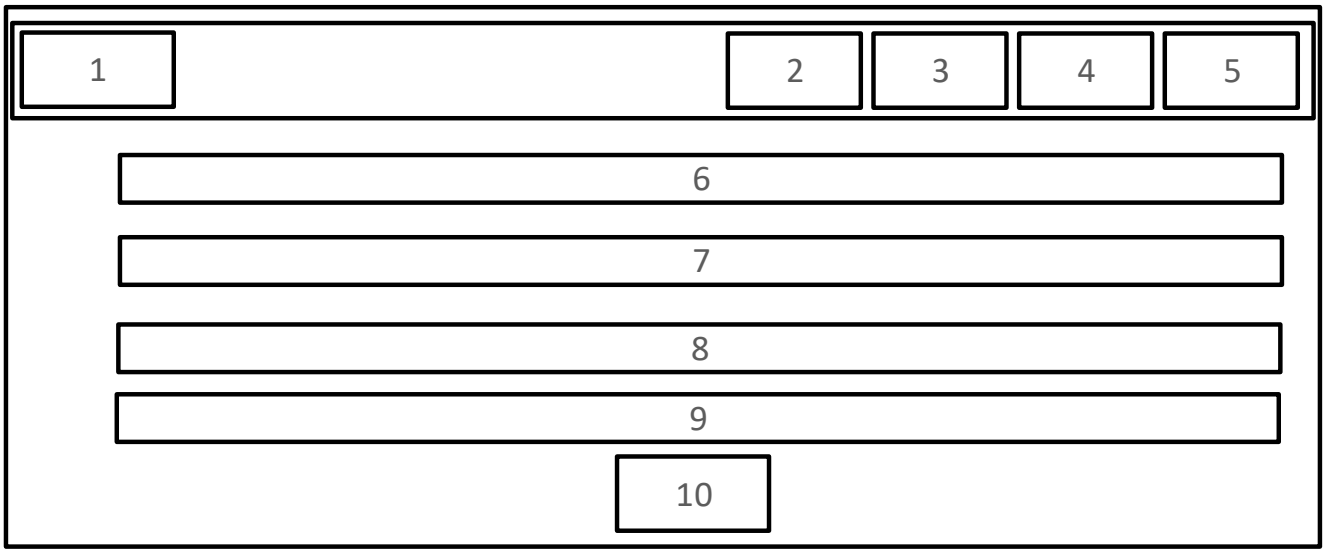


Рисунок 3.7 - Структурна схема вікна рейтингу

Складові вікна рейтингу:

1. Логотип.
2. Кнопка переходу до профілю.
3. Кнопка рейтингу.
4. Кнопка виходу з акаунту.
5. Кнопка зміни кольорової теми застосунку.

6-9. Перелік гравців

10. Меню для перемикання між сторінками рейтингу.

3.5 Розробка програмного інтерфейсу застосунку

Для розробки інтерфейсу застосунку було використано мову TypeScript. Вона добре інтегрується із бібліотекою React, яка використовувалась для розробки UI-компонентів.

Для навігації між сторінками додатку було використано React Router. Ця бібліотека дозволяє реалізувати маршрутизацію, створювати динамічні URL-

адреси та забезпечити плавний перехід між сторінками. Використання React Router допомогло організувати структуру додатку та підвищити його масштабованість.

Стилізація інтерфейсу здійснювалась за допомогою CSS modules. Цей підхід дозволяє ізолювати стильові правила та уникнути конфліктів імен класів. Таким чином, код CSS стає більш модульним та супроводжуваним.

Для авторизації та реєстрації користувачів було обрано інтерфейс користувача, який інтегрується з серверною частиною використовуючи REST API.

Особлива увага при розробці інтерфейсу приділялась забезпеченню адаптивності та кросплатформності застосунку. Оскільки цільова аудиторія може використовувати різні пристрої - від десктопних комп'ютерів до мобільних телефонів, було необхідно забезпечити коректне відображення та функціональність на всіх типах пристроїв.

Для вирішення цієї задачі використовувались такі підходи:

1. Застосування гнучких CSS-макетів з використанням медіа запитів для адаптації під різні розміри екранів.
2. Оптимізація графічних ресурсів (зображень, іконок тощо) для швидкого завантаження на мобільних пристроях.
3. Тестування інтерфейсу на різних пристроях та налаштування розміщення елементів для забезпечення зручності користування.
4. Використання адаптивної типографіки та масштабування елементів інтерфейсу відповідно до розміру екрану.

Окрім цього, для забезпечення кросплатформності було використано сучасні веб-технології, такі як HTML5 та CSS3. Це дозволило досягти високої сумісності та працездатності застосунку на різних операційних системах та браузерах.

Розробка інтерфейсу ігрового веб-додатку для розвитку логічного мислення є складним та комплексним процесом, що вимагає врахування багатьох факторів. Від вибору технологій та архітектури компонентів до забезпечення адаптивності та кросплатформності - кожен з цих аспектів відіграє важливу роль у створенні якісного та зручного для користувача інтерфейсу.

Застосування сучасних веб-технологій, модульної архітектури та адаптивного дизайну дозволило створити зручний та функціональний інтерфейс, який відповідає потребам та очікуванням цільової аудиторії.

3.6 Висновки

В цьому розділі, було проведено варіантний аналіз та обґрунтування вибору мови програмування розробки, середовища розробки та засобів реалізації інтерфейсу користувача. Було розроблено інтерфейс застосунку, описано результати розробки.

4 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Аналіз методів тестування

Тестування інтерфейсу веб-застосунків є невід'ємною частиною розробки веб-продуктів. Воно допомагає перевірити, чи відповідає інтерфейс вимогам користувачів, чи є він зручним і інтуїтивно зрозумілим [17].

Під час тестування інтерфейсу веб-застосунків перевіряються такі параметри, як навігація, розміщення елементів управління, зручність введення даних, швидкість завантаження сторінок, візуальне оформлення та інші.

Тестування ігрових веб-застосунків має свої особливості. Окрім стандартних параметрів, таких як функціональність і сумісність з різними браузерами, тут важливо перевірити й такі аспекти, як графіка, звук, швидкість реакції гри на дії користувача.

При тестуванні ігрових веб-застосунків також важливо перевірити, чи працює гра коректно при різних рівнях навантаження, чи не виникає проблем при тривалому використанні.

Веб-застосунки - це програми, які працюють в браузері користувача. Вони можуть бути як простими, так і дуже складними, залежно від призначення і функціональності [18].

Тестування веб-застосунків починається з перевірки їх функціональності. Перевіряється, чи працюють всі функції застосунку коректно, чи не виникає помилок при різних вхідних даних.

Після перевірки функціональності проводиться тестування сумісності. Перевіряється, чи працює веб-застосунок коректно в різних браузерах, на різних операційних системах і пристроях.

Важливим аспектом тестування веб-застосунків є перевірка безпеки. Перевіряється, чи немає в застосунку вразливостей, які можуть бути використані зловмисниками для отримання несанкціонованого доступу до даних користувачів.

Також важливо перевірити, чи відповідає веб-застосунок стандартам веб-

розробки. Це допоможе забезпечити його правильну роботу в різних браузерах і на різних пристроях.

Тестування веб-застосунків може проводитися як вручну, так і з використанням автоматизованих інструментів. Автоматизоване тестування дозволяє швидше і ефективніше перевірити велику кількість функцій застосунку.

При тестуванні інтерфейсу веб-застосунків важливо перевірити, чи є він зручним для користувачів з різними рівнями підготовки. Інтерфейс повинен бути інтуїтивно зрозумілим навіть для тих, хто вперше стикається з подібним застосунком.

Також важливо перевірити, чи відповідає інтерфейс вимогам доступності. Він повинен бути зручним для користування людьми з обмеженими можливостями, наприклад, з порушеннями зору або слуху.

При тестуванні ігрових веб-застосунків важливо перевірити, чи є гра цікавою і захоплюючою. Це можна зробити, проаналізувавши відгуки користувачів, які вже грали в цю гру.

Також важливо перевірити, чи є гра справедливою. Вона не повинна давати переваги деяким гравцям за рахунок інших, все має залежати лише від їхнього рівня майстерності.

При тестуванні веб-застосунків важливо перевірити, чи працює вони коректно при різних умовах навантаження. Це допоможе уникнути проблем при використанні застосунку великою кількістю користувачів.

Також важливо перевірити, чи працює веб-застосунок коректно при різних швидкостях Інтернет-з'єднання. Це допоможе забезпечити його правильну роботу для користувачів з різними умовами доступу до мережі.

При тестуванні інтерфейсу веб-застосунків важливо перевірити, чи є він зручним для користування на різних пристроях. Інтерфейс повинен коректно відображатися на екранах різного розміру і роздільної здатності.

Також важливо перевірити, чи працює веб-застосунок коректно при різних налаштуваннях браузера. Це допоможе уникнути проблем при використанні застосунку користувачами з різними налаштуваннями безпеки і конфіденційності.

При тестуванні ігрових веб-застосунків важливо перевірити, чи працює гра коректно при різних налаштуваннях графіки і звуку. Це допоможе забезпечити її правильну роботу на різних пристроях і в різних умовах.

Також важливо перевірити, чи працює гра коректно при різних налаштуваннях керування. Це допоможе забезпечити зручне керування для користувачів з різними вподобаннями і звичками.

При тестуванні веб-застосунків важливо перевірити, чи працює вони коректно при різних налаштуваннях регіональних параметрів. Це допоможе забезпечити їх правильну роботу для користувачів з різних країн і регіонів.

Також важливо перевірити, чи працює веб-застосунок коректно при різних налаштуваннях часових поясів. Це допоможе уникнути проблем з відображенням часу і дати для користувачів з різних країн.

При тестуванні інтерфейсу веб-застосунків важливо перевірити, чи є він зручним для користування на різних типах клавіатур. Інтерфейс повинен коректно відображатися і працювати на клавіатурах з різними розкладками і розмірами клавіш.

Також важливо перевірити, чи працює веб-застосунок коректно при використанні різних типів мишок. Це допоможе забезпечити зручне керування для користувачів з різними вподобаннями і звичками.

При тестуванні ігрових веб-застосунків важливо перевірити, чи працює гра коректно при використанні різних типів ігрових пристроїв. Це допоможе забезпечити зручне керування для користувачів з різними вподобаннями і звичками.

Також важливо перевірити, чи працює гра коректно при використанні різних типів екранів. Це допоможе забезпечити її правильну роботу на різних пристроях і в різних умовах.

При тестуванні веб-застосунків важливо перевірити, чи працює вони коректно при використанні різних типів мережевих з'єднань. Це допоможе забезпечити їх правильну роботу для користувачів з різними умовами доступу до мережі.

Також важливо перевірити, чи працює веб-застосунок коректно при використанні різних типів браузерів. Це допоможе забезпечити його правильну роботу для користувачів з різними вподобаннями і звичками.

При тестуванні інтерфейсу веб-застосунків важливо перевірити, чи є він зручним для користування на різних типах мобільних пристроїв. Інтерфейс повинен коректно відображатися і працювати на смартфонах, планшетах і інших мобільних пристроях.

Також важливо перевірити, чи працює веб-застосунок коректно при використанні різних типів операційних систем. Це допоможе забезпечити його правильну роботу для користувачів з різними вподобаннями і звичками.

4.2 Тестування інтерфейсу застосунку

Розпочнемо тестування інтерфейсу системи із вікна реєстрації. Зображення вікна реєстрації наведено на рисунку 4.1.

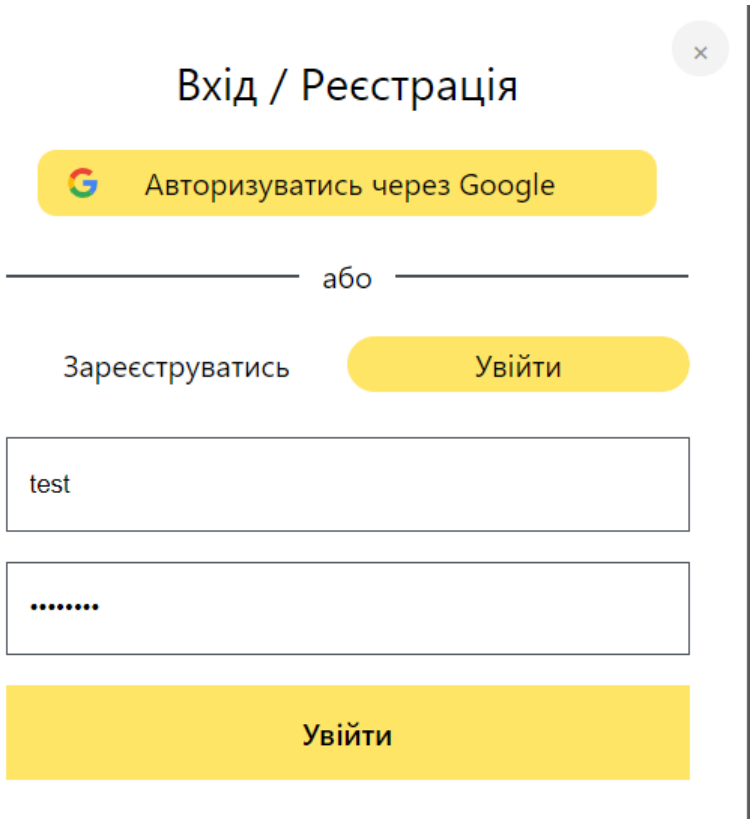


Рисунок 4.1 – Вікно реєстрації та авторизації

Введемо неправильний формат електронної адреси, наприклад, слово «test», що не є електронною адресою. В результаті, система виводить повідомлення про помилку та надає інформацію про зміст помилки. Зображення помилки наведено на рисунку 4.2.

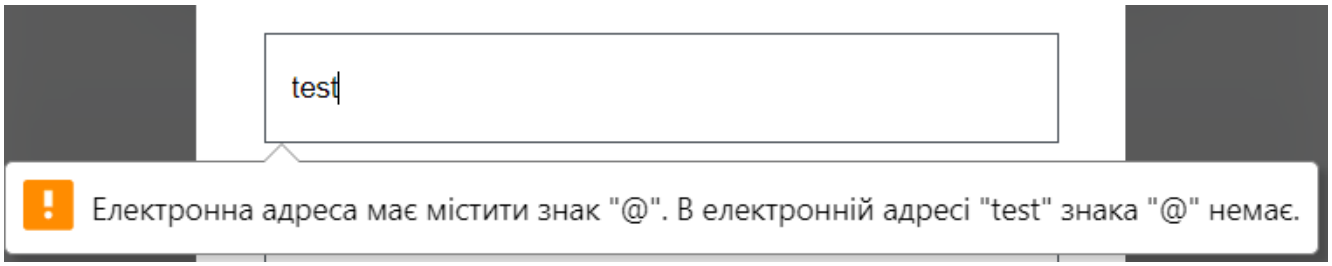


Рисунок 4.2 – Виведення помилки при введенні електронної адреси

Після введення правильних даних та успішної авторизації, можемо переглянути профіль користувача, зображення якого наведено на рисунку 4.3.

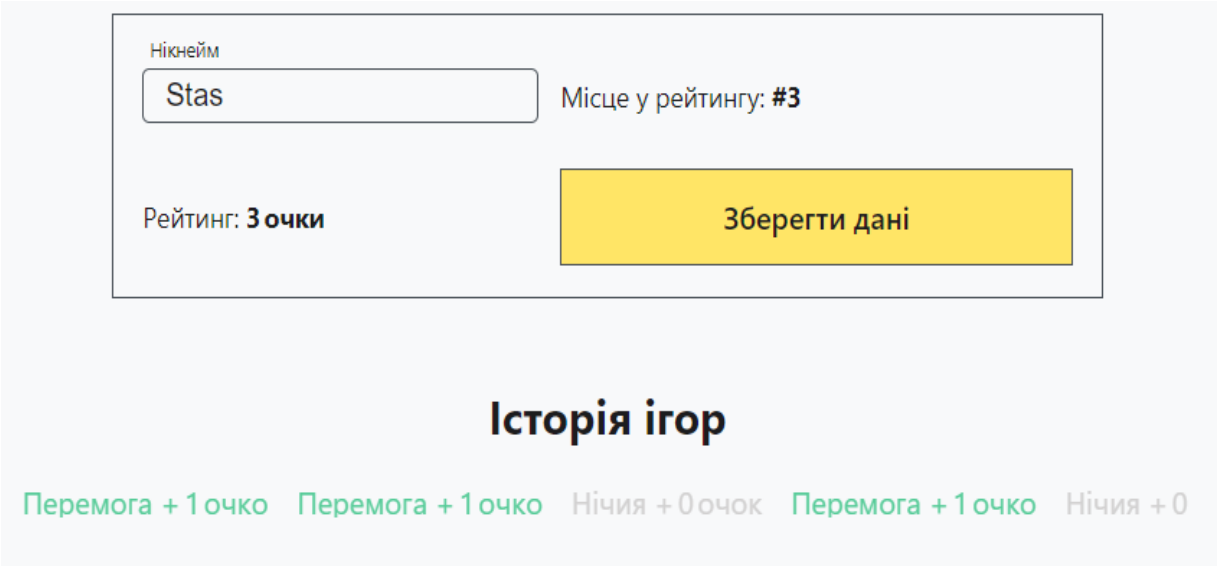


Рисунок 4.3 – Профіль користувача

Тут ми можемо побачити інформацію про нікнейм користувача, рейтинг, місце у рейтингу та історію ігор.

Оновимо ім’я користувача. Після успішної зміни, виводиться відповідне повідомлення, як на рисунку 4.4.

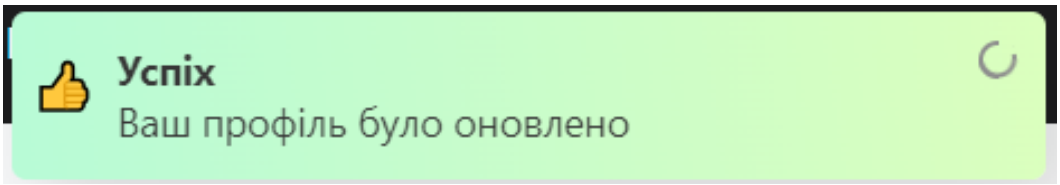


Рисунок 4.4 – Повідомлення про оновлення профілю

Відкриємо рейтинг користувачів. Його зображення подано на рисунку 4.5.

Рисунок 4.5 – Рейтинг користувачів

Перейдемо до вікна гри. Його зображення подано на рисунку 4.6.

Рисунок 4.6 – Ігрове вікно

Встановимо високу складність гри. Ігрове поле наприкінці гри наведено на рисунку 4.7.

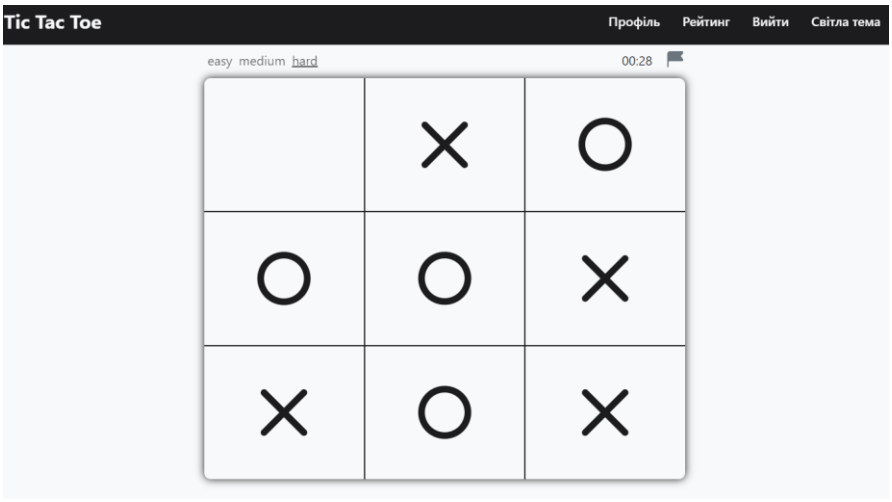


Рисунок 4.7 – Ігрове поле наприкінці гри

Гра проведена в нічию, у зв’язку з чим виводиться відповідне повідомлення (рисунок 4.8).

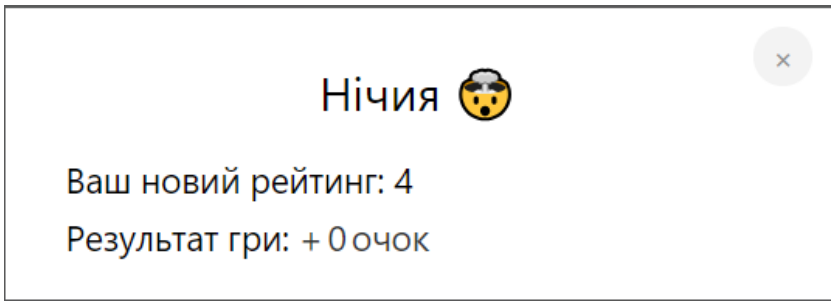


Рисунок 4.8 – Повідомлення про завершення гри

Для неавторизованих користувачів, це вікно виглядає так, як на рисунку 4.9.

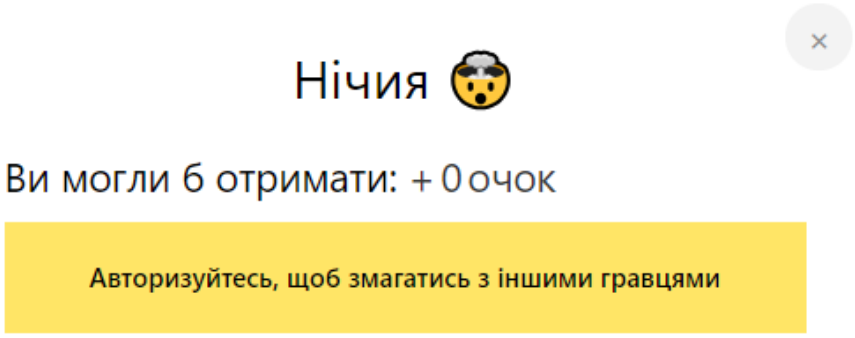


Рисунок 4.9 – Повідомлення про нічию для неавторизованих користувачів

Після зіграних успішних ігор, рейтинг користувачів змінився. Зміни в рейтингу зображені на рисунку 4.10.

Рисунок 4.10 – Зміни в рейтингу користувачів

Змінимо кольорову тему гри на темну тему. Результат зображено на рисунку 4.11.

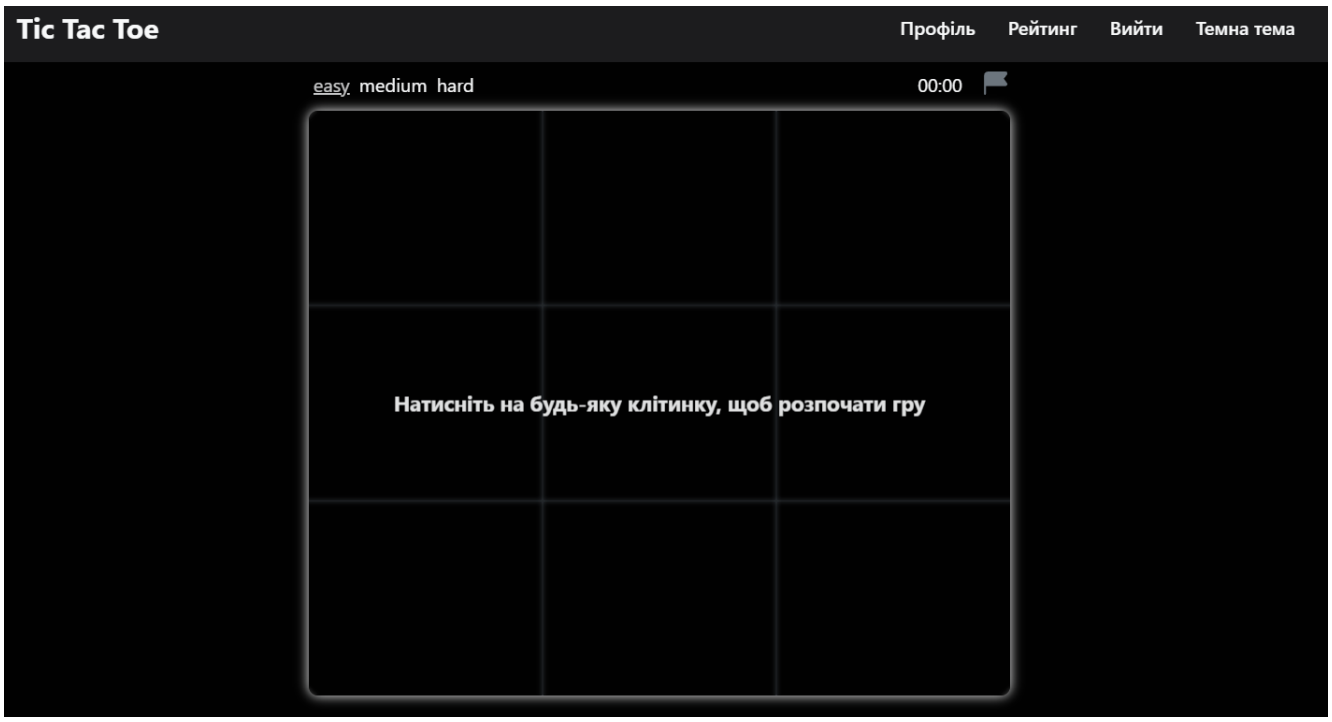


Рисунок 4.11 – Зміна кольорової теми на темну

Застосунок також адаптовано для мобільних пристроїв. Вигляд застосунку з мобільного пристрою наведено на рисунку 4.12.

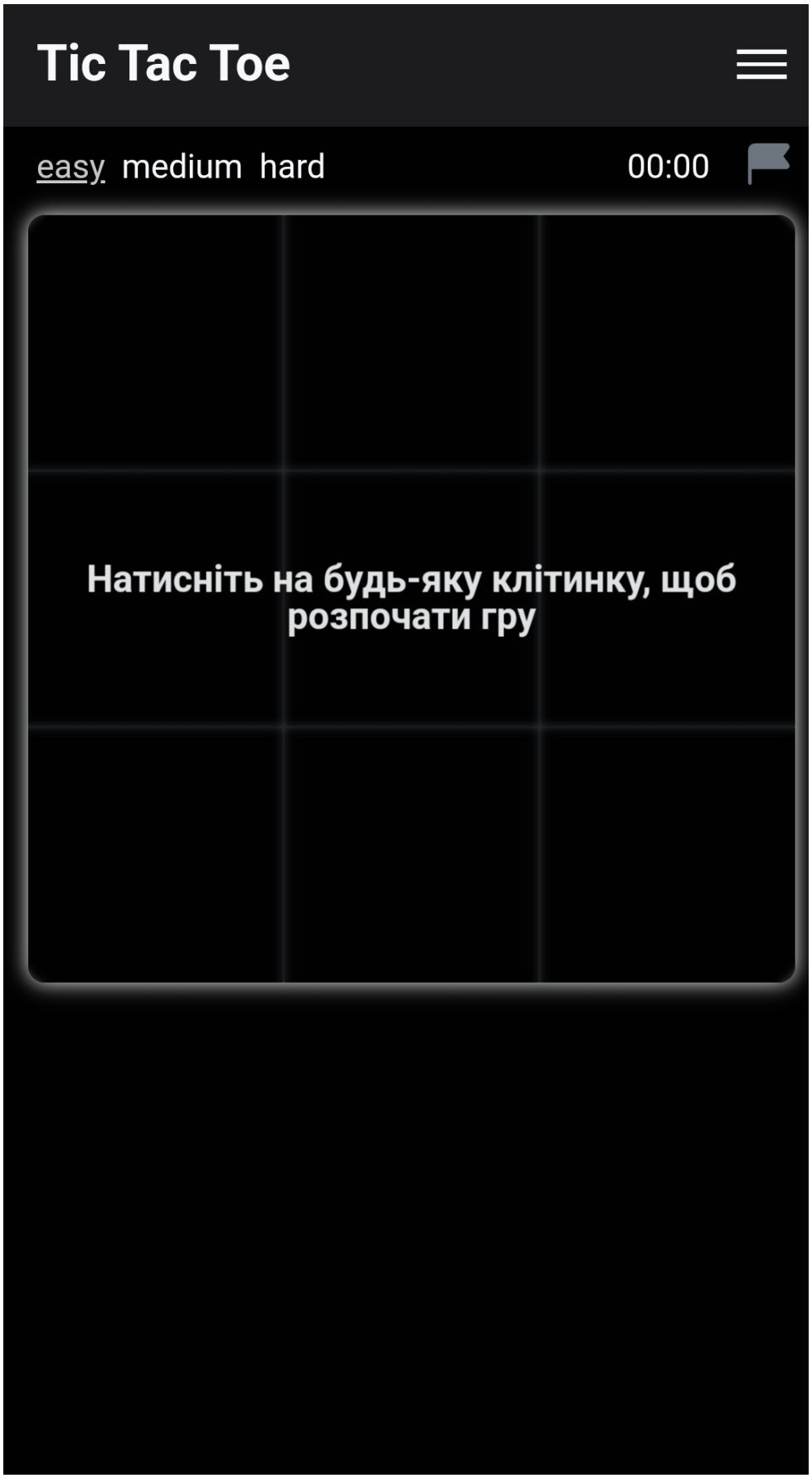


Рисунок 4.12 – Застосунок з мобільного пристрою

Навігацію в мобільній версії застосунку продемонстровано на рисунку 4.13.

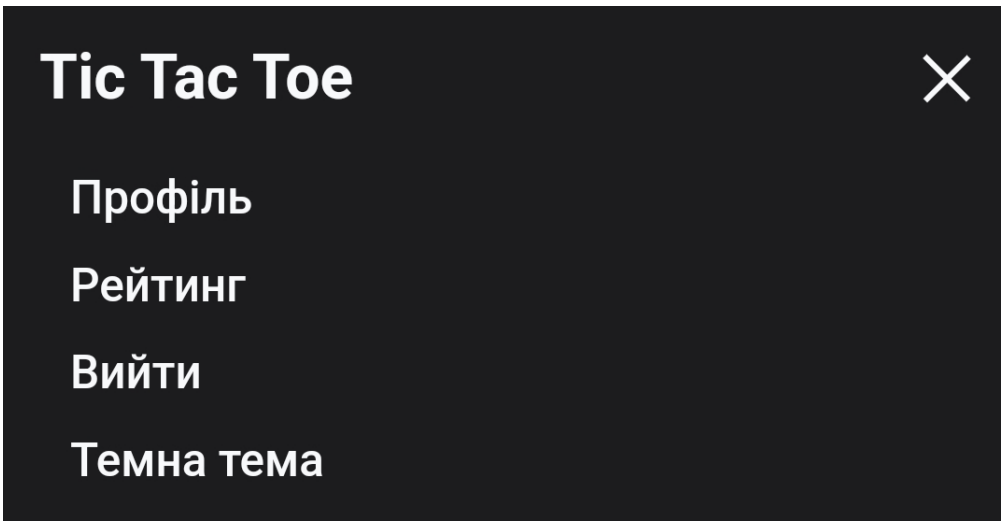


Рисунок 4.13 – Навігація в мобільній версії застосунку

Рейтинг гравців в мобільній версії наведено на рисунку 4.14.

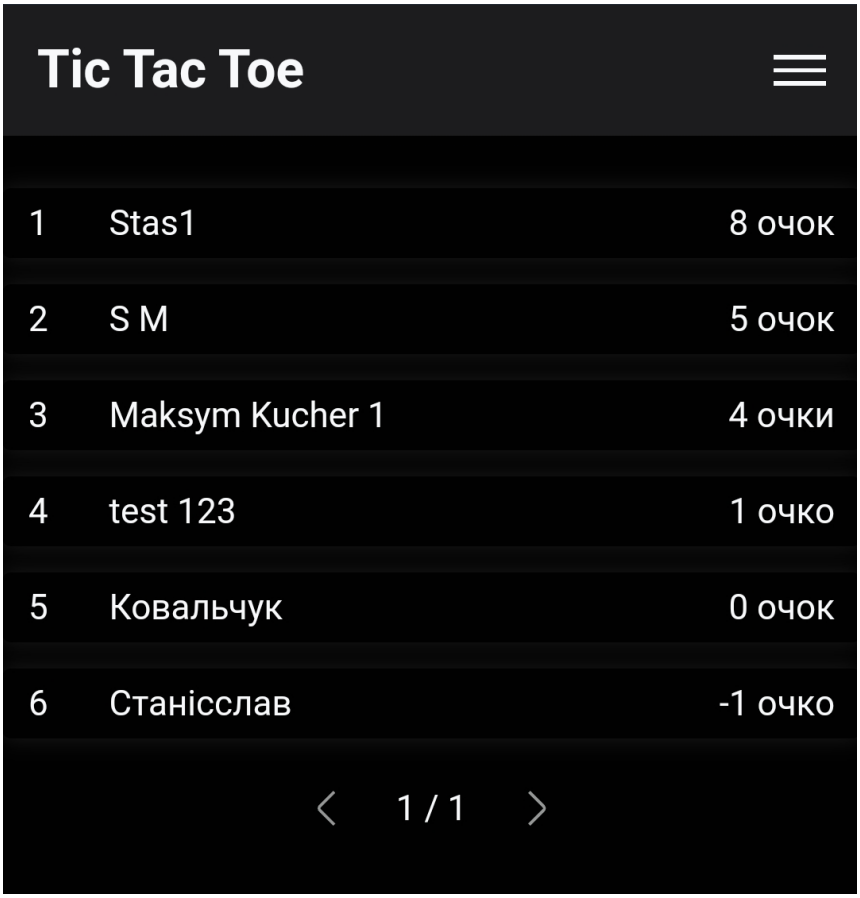


Рисунок 4.14 – Рейтинг гравців в мобільній версії

Профіль користувача в мобільній версії продемонстровано на рисунку 4.15.

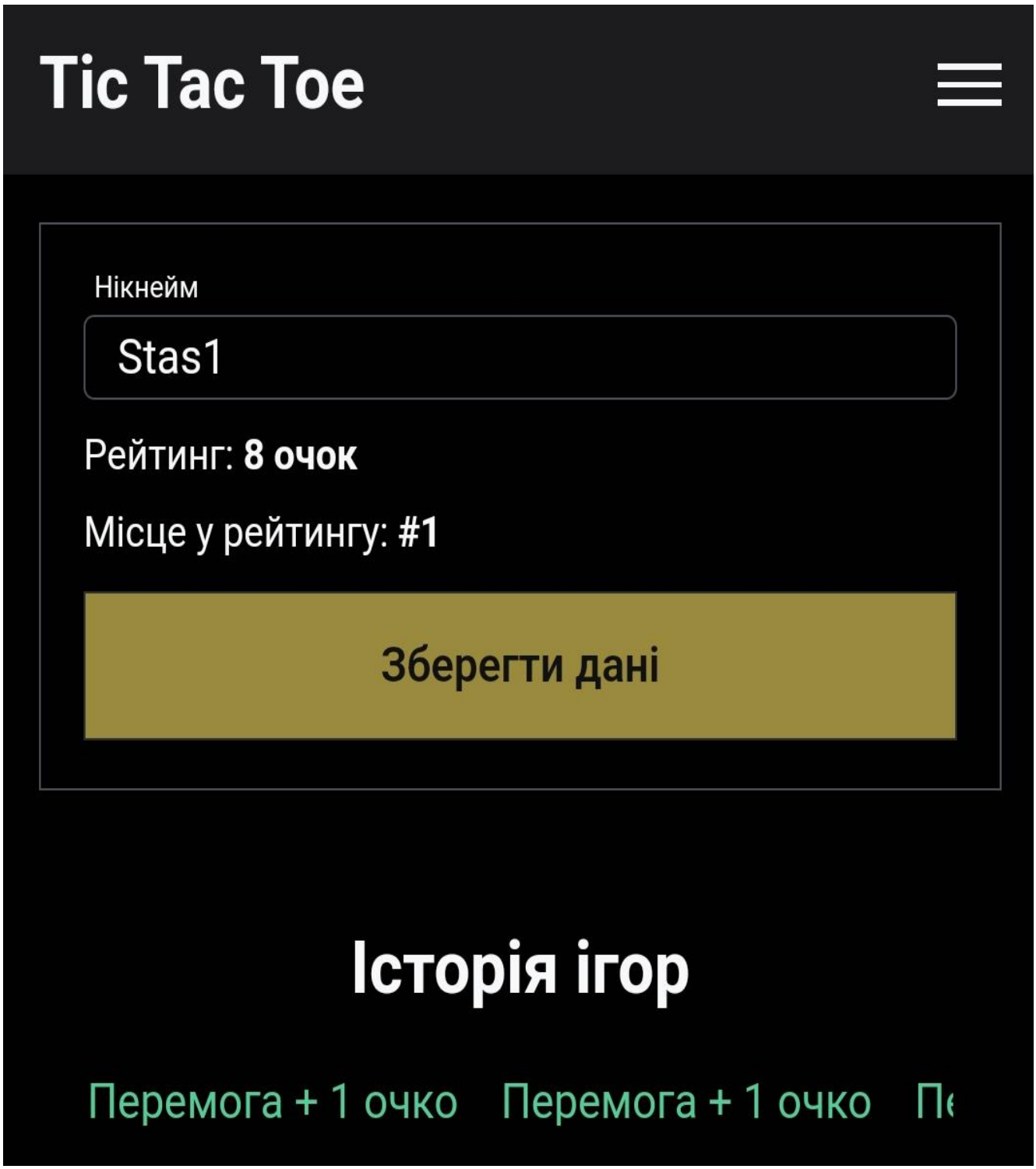


Рисунок 4.15 – Профіль користувача в мобільній версії

Результат гри в мобільній версії продемонстровано на рисунку 4.16.

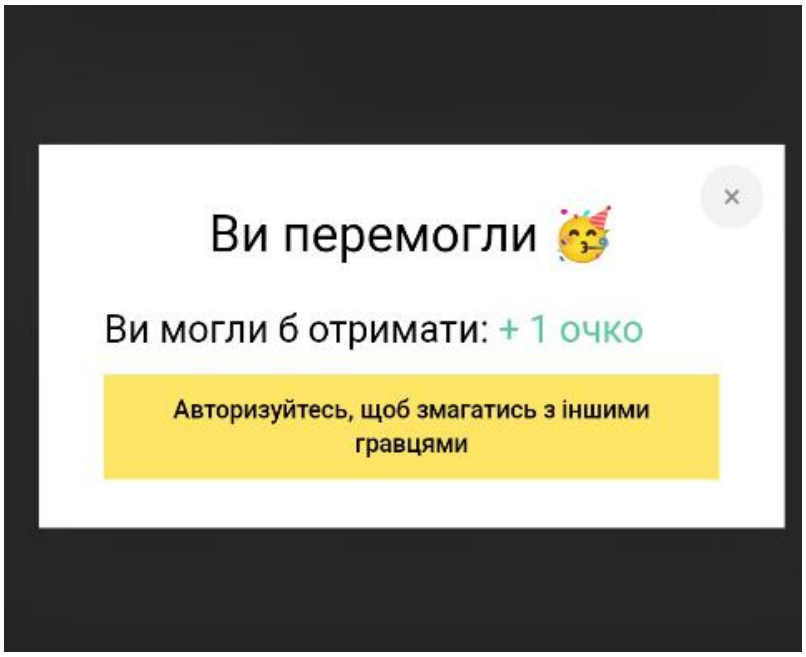


Рисунок 4.16 – Результат гри в мобільній версії

Для перегляду гри потрібно в профілі користувача обрати будь-яку із зіграних ігор. Вікно перегляду гри з першим ходом у ньому продемонстровано на рисунку 4.17.

Рисунок 4.17 – Перегляд гри (хід 1)

Так як це перший хід, кнопка переходу на попередній хід неактивна, тому можна переключитися лише на наступний хід (рисунок 4.18).

Рисунок 4.18 – Перегляд гри (хід 2)

Тепер активні кнопки переходу як на попередній, так і наступний ходи. Зробимо перехід на наступний хід (рисунок 4.19).

Рисунок 4.19 – Перегляд гри (хід 3)

Повернемося на попередній хід (рисунок 4.20).

Рисунок 4.20 – Повернення на хід 2

Тепер перейдемо до останнього ходу в грі (рисунок 4.21).

Рисунок 4.21 – Останній хід у грі

Як ми бачимо, після цього ходу гра завершилася, так як кнопка переходу на наступний хід неактивна. Гра завершилася тому, що була досягнута одна з умов перемоги, а саме – було виставлено 3 нулики по 2 вертикалі ігрового поля.

Таким чином, інтерфейс користувача працює так, як очікувалось та виконує

поставлені задачі розробки.

4.3 Висновки

В цьому розділі, було проведено аналіз методів тестування інтерфейсу веб-застосунку для розвитку логічного мислення. Було обрано метод ручного тестування, арґументовано такий вибір. Проведено тестування компонентів інтерфейсу застосунку, продемонстровано його працездатність та коректну роботу.

ВИСНОВКИ

В результаті виконання бакалаврської дипломної роботи, було спроектовано та візуалізовано інтерфейс web-застосунку для розвитку логічного мислення. Інтерфейс забезпечує взаємодію користувача із застосунком та його алгоритмами роботи. Він дозволяє грати у гру для розвитку логічного мислення, переглядати рейтинг гравців та власний рейтинг успішності гри.

Для розробки було використано середовище розробки Visual Studio Code, мову програмування TypeScript, фреймоворк React.

У першому розділі, було проведено аналіз стану питання розробки інтерфейсу застосунку, порівняльний аналіз аналогів, аналіз методів розв’язання задачі. На їх основі було доведено доцільність та актуальність власної розробки, поставлено задачі дослідження.

У другому розділі, було розроблено структурні схеми інтерфейсу застосунку, побудовано модель web-застосунку та розроблено алгоритми роботи реєстрації та авторизації і алгоритм процесу гри.

У третьому розділі, було проведено варіантний аналіз та обґрунтування вибору мови програмування розробки, здійснено вибір середовища розробки та інструменти розробки інтерфейсу застосунку. Розроблено та візуалізовано інтерфейс застосунку.

У четвертому розділі, було проведено аналіз методів тестування, проведено тестування розробленого інтерфейсу. Тестування довело працездатність та коректність роботи застосунку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. The Best Ways To Strengthen Your Logical Thinking Skills [Електронний ресурс] – Режим доступу до ресурсу: https://www.indeed.com/career-advice/career-development/strengthen-logical-thinking-skills

2. Кучер М.В., Мацедонський С.О. Розробка ігрового web-додатку для розвитку логічного мислення/М.В. Кучер, С.О. Мацедонський, В.П. Майданюк// Матеріали конференції «LIII Науково-технічна конференція підрозділів Вінницького національного технічного університету (2024) », Вінниця, 2024. [Електронний ресурс] – Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2024/paper/view/20790/17230

3. Elevate [Електронний ресурс] – Режим доступу до ресурсу: https://elevateapp.com/

4. Briangle [Електронний ресурс] – Режим доступу до ресурсу: https://www.braingle.com/

5. Cognito [Електронний ресурс] – Режим доступу до ресурсу: https://www.theguardian.com/technology/2016/mar/23/brain-training-apps-five-of-the-best

6. Firebase Authentication [Електронний ресурс] – Режим доступу до ресурсу: https://firebase.google.com/docs/auth

7. Firebase Realtime Database [Електронний ресурс] – Режим доступу до ресурсу: https://firebase.google.com/docs/database

8. HTML [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/ru/docs/Web/HTML

9. PHP [Електронний ресурс] – Режим доступу до ресурсу: https://www.php.net/

10. TypeScript [Електронний ресурс] – Режим доступу до ресурсу: https://www.typescriptlang.org/

11. Sublime Text [Електронний ресурс] – Режим доступу до ресурсу: https://www.sublimetext.com/

12. Visual Studio Code [Електронний ресурс] – Режим доступу до ресурсу: https://code.visualstudio.com/

13. IntelliJ IDEA [Електронний ресурс] – Режим доступу до ресурсу: https://www.jetbrains.com/idea/

14. Express.js [Електронний ресурс] – Режим доступу до ресурсу: https://expressjs.com/

15. Angular [Електронний ресурс] – Режим доступу до ресурсу: https://angular.io/

16. React [Електронний ресурс] – Режим доступу до ресурсу: https://legacy.reactjs.org/

17. Pradeep Soundararajan. Buddha in Testing. Ченнай: Notion Press, 2020. 132с.

18. Peter Yaworski. Real-World Bug Hunting. Сан-Франциско: No Starch Press, 2019. 264с.

58

ДОДАТОК А

(обов'язковий)

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О.Н.
«12» березня 2024 р.

Технічне завдання

на бакалаврську дипломну роботу «Розробка ігрового WEB-застосунку для розвитку логічного мислення. Частина 1. Проектування та візуалізація інтерфейсу додатку»

за спеціальністю

121 – Інженерія програмного забезпечення

Керівник бакалаврської дипломної роботи:
к.т.н., доц. В.П. Майданюк
«12» березня 2024 р.

Виконав:
студент гр. 2ПІ-20б Мацедонський С.О.
«12» березня 2024 р.

Вінниця ВНТУ – 2024 рік

1. Найменування та галузь застосування

Бакалаврська дипломна робота: «Розробка ігрового WEB-додатку для розвитку логічного мислення. Частина 1. Проектування та візуалізація інтерфейсу додатку».

Галузь застосування – веб-технології.

2. Підстава для розробки

Підставою для виконання бакалаврської дипломної роботи (БДР) є індивідуальне завдання на БДР та наказ № 80 від «11» березня 2024 ректора ВНТУ про закріплення тем БДР.

3. Мета та призначення розробки

Метою роботи є підвищення зручності використання застосунку для розвитку логічного мислення людини за рахунок розширення функціоналу клієнтської частини застосунку.

Основними задачами дослідження є:

- провести аналіз стану питання проектування та візуалізації інтерфейсу;
- розробити модель застосунку;
- розробити структуру клієнтської частини веб-застосунку;
- розробити алгоритм роботи програми;
- провести варіантний аналіз та обґрунтувати вибір мови програмування, середовища розробки та засобів реалізації інтерфейсу застосунку;
- розробити інтерфейс для реєстрації та авторизації користувача;
- розробити профіль користувача;
- розробити логіку перегляду зіграних ігор;
- розробити інтерфейс для гри;
- забезпечити адаптивність застосунку для різних видів пристроїв;
- провести тестування.

4. Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись БДР:

1. The Best Ways To Strengthen Your Logical Thinking Skills [Електронний ресурс] – Режим доступу до ресурсу: https://www.indeed.com/career-advice/career-development/strengthen-logical-thinking-skills

2 TypeScript [Електронний ресурс] – Режим доступу до ресурсу: https://www.typescriptlang.org/

3. Visual Studio Code [Електронний ресурс] – Режим доступу до ресурсу: https://code.visualstudio.com/

4. React [Електронний ресурс] – Режим доступу до ресурсу: https://legacy.reactjs.org/

5. Технічні вимоги

Середовище розробки – Visual Studio Code.

Мова програмування – TypeScript.

Вхідні дані – інформація про користувача.

Вихідні дані – рейтинг гравців, історія ігор.

6. Конструктивні вимоги

Конструкція пристрою повинна відповідати естетичним та ергономічним вимогам, повинна бути зручною в обслуговуванні та керуванні.

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред’являється по закінченню робіт:

- пояснювальна записка до бакалаврської дипломної роботи;
- технічне завдання;
- лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки:

№ з/п	Назва етапів дипломного проєкту	Строк виконання етапів роботи
1	Обґрунтування вибору методу розробки та постановка задач	14.03.24 –22.03.24
2	Розробка архітектури та алгоритмів програмного продукту	13.03.24 –22.03.24
3	Аналіз і вибір мови програмування та середовища розробки	25.03.24 –19.04.24
4	Розробка програмного продукту	22.04.24 –10.05.24
5	Тестування програми	13.05.24 – 24.05.24
6	Оформлення матеріалів до захисту БДР	27.05.24 – 30.05.24

10. Порядок контролю та прийняття

Порядок контролю і приймання роботи регламентується відповідними документами ВНТУ і державними стандартами.

62

ДОДАТОК Б

(обов'язковий)

ПРОТОКОЛ

ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Розробка ігрового WEB-застосунку для розвитку логічного мислення. Частина 1. Проектування та візуалізація інтерфейсу додатку.

Тип роботи: БДР

Підрозділ : кафедра програмного забезпечення, ФІТКІ

Науковий керівник: к.т.н., доц. каф. ПЗ Майданюк В.П..

Оригінальність	94,4
Схожість	5,6

Особа, відповідальна за перевірку Черноволик Г. О.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Аналіз звіту подібності

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

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

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

Автор роботи Мацедонський О.С..

Керівник роботи Майданюк В.П..

ДОДАТОК В

(довідниковий)

Текст коду клієнтської частини застосунку для розвитку логічного мислення

File: /src/index.tsx

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import { createBrowserRouter, RouterProvider } from 'react-router-dom';

// consts
import { routes } from './shared/consts/routes';

// providers
import { ThemeProvider } from './providers/theme/theme.provider';
import { GameProvider } from './providers/game/game.provider';
import { UserProvider } from './providers/user/user.provider';
import { AppProvider } from './providers/app/app.provider';
import { ToasterProvider } from './providers/toaster/toaster.provider';

// styles
import './shared/styles/colors.module.css';
import './shared/styles/reset.module.css';

const router = createBrowserRouter(routes);
const root = ReactDOM.createRoot(document.getElementById('root') as HTMLElement);

root.render(
  <React.StrictMode>
    <ToasterProvider>
      <AppProvider>
        <UserProvider>
          <ThemeProvider>
            <GameProvider>
              <RouterProvider router={router} />
            </GameProvider>
          </ThemeProvider>
        </UserProvider>
      </AppProvider>
    </ToasterProvider>
  </React.StrictMode>,
);
```

File: /src/shared/styles/colors.module.css

```
html {
  /* COLORS SAME ON EVERY THEME */
  --color-red: #e63946;
```

```
  --color-green: #57cc99;
  --color-accent: #ffe566;
  --color-backdrop: rgba(0, 0, 0, 0.15);
  --color-border: #495057;

  --color-dark: #1c1c1e;
  --color-dark-black: #010101;
  --color-light: #f8f9fa;
  --color-light-white: #f8f9fa;

  --color-gray-dark: #363636;
  --color-gray-middle: #8e8e8e;
  --color-gray-light: #d2d1d1;

  /* TOASTER COLORS */
  --color-gradient-success: linear-gradient(81.4deg, #b8fbd7 0%, #dbffbe 98.71%);
  --color-gradient-error: linear-gradient(74.06deg, #fbdcb8 0%, #feffbe 115.59%);
  --color-gradient-info: linear-gradient(236.63deg, #bff0ff 0%, #8bd5ff 92.41%);

  /* DARK THEME COLORS */
 &[data-theme='dark'] {
  --color-backgound: var(--color-dark-black);
  --color-loader: var(--color-light-white);
    --color-shadow-bg: #2d2d2dcc;

    --color-cell-hover: #1e201e;
    --color-surrender: #6c757d;

  /* SLIDER */
  --brand-primary: rgb(138, 180, 248);
  --brand-secondary: rgb(193, 168, 226);
  --brand-alternative: rgb(136, 186, 191);
  --background-site: rgb(0, 0, 0);
  --background-code: rgb(12, 12, 12);
  --text-body: rgb(222, 222, 222);
  --text-comment: rgb(170, 170, 170);
  --text-high-contrast: rgb(230, 230, 230);
  --text-medium-contrast: rgb(202, 202, 202);
  --text-low-contrast: rgb(170, 170, 170);
  --detail-high-contrast: rgb(101, 101, 101);
  --detail-medium-contrast: rgb(25, 25, 25);
  --detail-low-contrast: rgb(21, 21, 21);
  --admonition-note: rgb(138, 180, 248);
  --admonition-warning: rgb(253, 186, 116);
  --admonition-danger: rgb(220, 38, 38);
  --brand-primary-rgb-value: 138, 180, 248;
  --brand-secondary-rgb-value: 193, 168, 226;
  --brand-alternative-rgb-value: 136, 186, 191;
  --background-site-rgb-value: 0, 0, 0;
  --background-code-rgb-value: 12, 12, 12;
  --text-body-rgb-value: 222, 222, 222;
  --text-comment-rgb-value: 170, 170, 170;
```

```
--text-high-contrast-rgb-value: 230, 230, 230;
--text-medium-contrast-rgb-value: 202, 202, 202;
--text-low-contrast-rgb-value: 170, 170, 170;
--detail-high-contrast-rgb-value: 101, 101, 101;
--detail-medium-contrast-rgb-value: 25, 25, 25;
--detail-low-contrast-rgb-value: 21, 21, 21;
--admonition-note-rgb-value: 138, 180, 248;
--admonition-warning-rgb-value: 253, 186, 116;
--admonition-danger-rgb-value: 220, 38, 38;
}

/* LIGHT THEME COLORS */
&[data-theme='light'] {
  --color-backgound: var(--color-light-white);
--color-loader: var(--color-dark-black);
  --color-shadow-bg: #3b3b3bcc;

  --color-cell-hover: #e8ebee;
  --color-surrender: #6c757d;

/* SLIDER */
--brand-primary: rgb(47, 112, 193);
--brand-secondary: rgb(116, 97, 195);
--brand-alternative: rgb(19, 120, 134);
--background-site: rgb(249, 249, 249);
--background-code: rgb(244, 244, 244);
--text-body: rgb(54, 49, 61);
--text-comment: rgb(99, 94, 105);
--text-high-contrast: rgb(49, 49, 49);
--text-medium-contrast: rgb(99, 94, 105);
--text-low-contrast: rgb(116, 109, 118);
--detail-high-contrast: rgb(192, 192, 192);
--detail-medium-contrast: rgb(234, 234, 234);
--detail-low-contrast: rgb(240, 240, 242);
--admonition-note: rgb(46, 109, 188);
--admonition-warning: rgb(255, 196, 9);
--admonition-danger: rgb(220, 38, 38);
--brand-primary-rgb-value: 47, 112, 193;
--brand-secondary-rgb-value: 116, 97, 195;
--brand-alternative-rgb-value: 19, 120, 134;
--background-site-rgb-value: 249, 249, 249;
--background-code-rgb-value: 244, 244, 244;
--text-body-rgb-value: 54, 49, 61;
--text-comment-rgb-value: 99, 94, 105;
--text-high-contrast-rgb-value: 49, 49, 49;
--text-medium-contrast-rgb-value: 99, 94, 105;
--text-low-contrast-rgb-value: 116, 109, 118;
--detail-high-contrast-rgb-value: 192, 192, 192;
--detail-medium-contrast-rgb-value: 234, 234, 234;
--detail-low-contrast-rgb-value: 240, 240, 242;
--admonition-note-rgb-value: 46, 109, 188;
--admonition-warning-rgb-value: 255, 196, 9;
```

```
  --admonition-danger-rgb-value: 220, 38, 38;
  }
}
```

File: /src/shared/styles/vars.module.css

```
@value system-font: -apple-system, BlinkMacSystemFont, "Segoe UI", "Roboto", "Oxygen", "Ubuntu", "Cantarell", "Fira Sans", "Droid Sans", "Helvetica Neue", sans-serif;

@value tablet: 768px;
@value tablet-min: 600px;
@value phone: 450px;
@value phone-max: 550px;

html {
 --transform-menu: translateY(0);
  --transition-duration: 0.3s;

  --transform-toast: 0;
  --toast-transition-duration: 0;
}
```

File: /src/shared/consts/routes.tsx

```
// pages
import { RouteObject } from 'react-router';
import { AppWrapper } from '../../components/app-wrapper/app-wrapper';
import { NotFoundPage } from '../../pages/404/404';
import { MainScreen } from '../../pages/main-screen/main-screen';
import { ProfileScreen } from '../../pages/profile/profile';
import { RatingScreen } from '../../pages/rating/rating';

export const MAIN_PAGE_ROUTE = '/';
export const PROFILE_PAGE_ROUTE = '/profile';
export const RATING_PAGE_ROUTE = '/rating';

export const routes: RouteObject[] = [
  {
    element: <AppWrapper />,
    children: [
      {
        index: true,
        element: <MainScreen />,
        errorElement: <NotFoundPage />,
      },
      {
        path: '/profile',
        element: <ProfileScreen />,
        errorElement: <NotFoundPage />,
      },
      {
        path: '/rating',
        element: <RatingScreen />,
        errorElement: <NotFoundPage />,
```

```
    },
   ],
  },
];
```

File: src/providers/toaster/toaster.provider.tsx

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

// context
import { toasterContext } from './toaster.context';

// types
import type { ToasterContextType, ToasterMessage, ToasterTypes } from './toaster.types';

type Props = {
 children: React.ReactNode;
};

export const ToasterProvider = ({ children }: Props) => {
 const [messages, setMessages] = useState<ToasterMessage[]>([]);

 const addToasterMessage = useCallback(
  (message: string, header: string, type: ToasterTypes): void => {
   setMessages(prev => [
    ...prev,
    {
     type,
     message,
     header,
     unique: Math.random().toString(36).substring(2) + Date.now().toString(36),
    },
   ]);
  },
  [],
 );

 const bug = useCallback(
  (rawError: Error | string, header = '') => {
   if (rawError instanceof Error) {
    addToasterMessage(rawError.message, header, 'error');
   } else {
    addToasterMessage(rawError, header, 'error');
   }
  },
  [addToasterMessage],
 );

 const success = useCallback(
  (message: string, header = '') => {
   addToasterMessage(message, header, 'success');
  },
  [addToasterMessage],
```

```
  );

  const info = useCallback(
    (message: string, header = '') => {
      addToasterMessage(message, header, 'info');
    },
    [addToasterMessage],
  );

  const warning = useCallback(
    (message: string, header = ''): void => {
      addToasterMessage(message, header, 'warning');
    },
    [addToasterMessage],
  );

  const removeToasterMessage = useCallback((unique: string) => {
    setMessages(prev => prev.filter(message => unique !== message.unique));
  }, []);

  const memoContext = useMemo<ToasterContextType>(
    () => ({
      addToasterMessage,
      removeToasterMessage,
      bug,
      info,
      warning,
      success,
      messages,
    }),
    [addToasterMessage, bug, info, messages, removeToasterMessage, success, warning],
  );

  return <toasterContext.Provider value={memoContext}>{children}</toasterContext.Provider>;
};
```

File: src/providers/toaster/toaster.context.ts

```
import React from 'react';

// types
import { ToasterContextType } from './toaster.types';

export const toasterContext = React.createContext<ToasterContextType>({
  bug: () => {},
  info: () => {},
  warning: () => {},
  success: () => {},
  removeToasterMessage: () => {},
  messages: [],
});

export const useToasterContext = () => {
```

```
  const context = React.useContext(toasterContext);
  if (!context) {
    throw new Error('useToaster must be used within a ToasterProvider');
  }
  return context;
};
```

File: src/providers/toaster/toaster.consts.ts

```
export const TOASTER_DURATION = 4000;
```

File: src/providers/theme/theme.provider.tsx

```
import { useLayoutEffect, useMemo, useState } from 'react';

// types
import { ThemeContextType, ThemesType } from './theme.types';

// context
import { themeContext } from './theme.context';

// consts
import { LIGHT_THEME, DARK_THEME } from './theme.consts';

export const ThemeProvider = ({ children }: { children: React.ReactNode }) => {
  const getThemeFromBrowser = () => {
    const datasetTheme = document.documentElement.dataset.theme as ThemesType | undefined;

    const isOsDarkMode = window.matchMedia('(prefers-color-scheme: dark)');
    const osMode = isOsDarkMode.matches ? DARK_THEME : LIGHT_THEME;

    return datasetTheme ?? osMode;
  };

  const [theme, setTheme] = useState<ThemesType>(getThemeFromBrowser);

  // On theme value change set theme in html dataset attributes
  useLayoutEffect(() => {
    if (document.documentElement && document.documentElement.dataset) {
      document.documentElement.dataset.theme = theme;
    }
  }, [theme]);

  const memoThemeContextValue = useMemo<ThemeContextType>(
    () => ({
      theme,
      toggleTheme: () => setTheme(theme => (theme === LIGHT_THEME ? DARK_THEME : LIGHT_THEME)),
    }),
    [theme],
  );
  return <themeContext.Provider value={memoThemeContextValue}>{children}</themeContext.Provider>;
};
```

File: src/providers/theme/theme.context.ts

```
import React from 'react';

// consts

// types
import { ThemeContextType } from './theme.types';

export const themeContext = React.createContext<ThemeContextType>({
  theme: 'light',
  toggleTheme: () => {},
});

export const useThemeContext = () => {
  const context = React.useContext(themeContext);

  if (!context) throw new Error('themeContext should be used within ThemeProvider');

  return context;
};
```

File: src/providers/theme/theme.consts.ts

```
export const LIGHT_THEME = 'light';
export const DARK_THEME = 'dark';
```

File: src/components/toaster/toaster-message.tsx

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

// context
import { useToasterContext } from '../../providers/toaster/toaster.context';

// consts
import { TOASTER_DURATION } from '../../providers/toaster/toaster.consts';

// types
import type { ToasterMessage } from '../../providers/toaster/toaster.types';

// styles
import styles from './toaster.module.css';
import classNames from 'classnames';

type Props = {
  message: ToasterMessage;
};

export function ToasterMessage({ message }: Props): JSX.Element {
  const { removeToasterMessage } = useToasterContext();

  const [isPausedToast, setPausedToast] = useState(false);

  const [touchStart, setTouchStart] = useState(0);
  const [touchEnd, setTouchEnd] = useState(0);
  const [needClose, setNeedClose] = useState(false);
```

```
const isError = message.type === 'error';
const isSuccess = message.type === 'success';
const isInfo = message.type === 'info';
const isWarning = message.type === 'warning';

const swipeSensativeAmountPixels = 70;

const handleTouchStart = (e: React.TouchEvent) => {
  setTouchStart(e.targetTouches[0].clientX);
};

const handleTouchMove = (e: React.TouchEvent) => {
  setTouchEnd(e.targetTouches[0].clientX);
};

const resetStyles = () => {
  document.documentElement.style.setProperty('--transform-toast', '0');
  document.documentElement.style.setProperty('--toast-transition-TOASTER_DURATION', '0');
};

const handleClickToast = () => {
  if (!isPausedToast) setPausedToast(true);
};

useEffect(() => {
  let toastTimeout: number | null = null;
  if (needClose) {
    document.documentElement.style.setProperty('--transform-toast', '110%');
    document.documentElement.style.setProperty('--toast-transition-TOASTER_DURATION', '0.3s');
    toastTimeout = window.setTimeout(() => {
      removeToasterMessage(message.unique);
    }, 300);
  }
  return () => {
    if (toastTimeout) window.clearTimeout(toastTimeout);
  };
}, [needClose]);

useEffect(() => {
  resetStyles();
  document.documentElement.style.setProperty(
    '--toaster-half-time-animation',
    `${TOASTER_DURATION / 2}ms`,
  );
  document.documentElement.style.setProperty('--toaster-time-animation', `${TOASTER_DURATION}ms`);

  return () => {
    resetStyles();
  };
}, []);
```

```
useEffect(() => {
  if (touchStart && touchEnd) {
    const deltaX = `${touchEnd - touchStart}px`;
    document.documentElement.style.setProperty('--transform-toast', `${deltaX}`);
  }
}, [touchStart, touchEnd]);

useEffect(() => {
  let toastTimeout: number | null = null;
  if (isPausedToast) {
    if (toastTimeout) window.clearTimeout(toastTimeout);
  } else {
    toastTimeout = window.setTimeout(() => {
      removeToasterMessage(message.unique);
    }, TOASTER_DURATION);
  }
  return () => {
    if (toastTimeout) window.clearTimeout(toastTimeout);
  };
}, [isPausedToast]);

const handleTouchEnd = (unique: string) => {
  if (touchEnd && touchStart - touchEnd > swipeSensativeAmountPixels) {
    document.documentElement.style.setProperty('--transform-toast', '-110%');
    document.documentElement.style.setProperty('--toast-transition-TOASTER_DURATION', '0.3s');

    removeToasterMessage(unique);
  } else if (touchEnd && touchStart - touchEnd < -swipeSensativeAmountPixels) {
    document.documentElement.style.setProperty('--transform-toast', '110%');
    document.documentElement.style.setProperty('--toast-transition-TOASTER_DURATION', '0.3s');
    removeToasterMessage(unique);
  }
  resetStyles();
};

return (
  <div
    role="button"
    onClick={handleClickToast}
    onKeyDown={handleClickToast}
    onTouchStart={handleTouchStart}
    onTouchMove={handleTouchMove}
    onTouchEnd={() => handleTouchEnd(message.unique)}
    tabIndex={0}
    className={classNames(styles['toaster-message'], styles[message.type], {
      [styles['toaster-open']]: isPausedToast,
    })}
  >
    <div className={styles['toaster-header']}>
      {message.header
        ? message.header
        : isError
```

```
          ? 'Помилка'
          : isSuccess
          ? 'Успіх'
          : isInfo
          ? 'Інформація'
          : isWarning
          ? 'Увага'
          : null}
      </div>
      <div className={styles['toaster-body']}>{message.message}</div>
      {isPausedToast ? (
        <button onClick={() => setNeedClose(true)} className={styles['toaster-close']}>
          ×
        </button>
      ) : (
        <div className={styles['toaster-loader']} />
      )}
    </div>
  );
}
```

File: src/components/toaster/toaster-popup.tsx

```
// context
import { useToasterContext } from '../../providers/toaster/toaster.context';

// components
import { ToasterMessage } from './toaster-message';

// styles
import styles from './toaster.module.css';

export function ToasterMessages(): JSX.Element | null {
  const { messages } = useToasterContext();

  return (
    <>
      {messages.length ? (
        <div className={styles['toaster-wrap']}>
          {messages.map(message => {
            if (!message.message) return null;

            return <ToasterMessage key={message.unique} message={message} />;
          })}
        </div>
      ) : null}
    </>
  );
}
```

File: src/components/toaster/toaster.module.css

```
@value vars: "../../shared/styles/vars.module.css";
@value tablet from vars;
```

```
.toaster-wrap {
  position: fixed;
  top: 20px;
  right: 12px;
  z-index: 1041;
  transform: var(--transform-menu);
  transition: transform linear;
  transition-duration: var(--transition-duration);

  @media (max-width: tablet) {
    transform: unset;
    top: 5px;
    right: 3px;
    left: 3px;
  }
}

.toaster-message {
  min-height: 60px;
  width: 370px;
  box-sizing: border-box;
  padding: 10px 22px 12px 38px;
  font-size: 14px;
  line-height: 18px;
  position: relative;
  color: var(--color-gray-dark);
  border-radius: 5px;
  box-shadow: 10px 10px 30px var(--color-backdrop);
  animation-duration: 0.3s;
  animation-name: appearance;
  opacity: 0;
  pointer-events: none;
  margin: 0 auto;

  &::before {
    position: absolute;
    top: 16px;
    left: 8px;
    margin-right: 8px;
    font-size: 18px;
    text-align: center;
  }

  &:first-child,
  &:nth-child(2),
  &:nth-child(3) {
    opacity: 1;
    pointer-events: auto;
    cursor: pointer;
  }
```

```
 &:first-child {
  z-index: 4;
  transition: transform var(--toast-transition-duration) linear, width 0.2s linear;
  transform: translateX(var(--transform-toast));
 }

 &:nth-child(2) {
  width: 346px;
  transform: translateY(calc(12px - 100%));
  z-index: 3;
 }

 &:nth-child(3) {
  width: 333px;
  transform: translateY(calc(-36px - 100%));
  z-index: 2;
 }

 &:focus {
  outline: none;
 }

 &.toaster-open {
  height: auto;

  .toaster-body {
   overflow: unset;
   text-overflow: unset;
   white-space: unset;
  }
 }

&.error,
&.warning {
 background: var(--color-gradient-error);

 &::before {
  content: '';
 }
}

&.success {
 background: var(--color-gradient-success);

 &::before {
  content: '';
 }
}

&.info {
 background: var(--color-gradient-info);
```

```
 &::before {
  content: '';
 }
}

.toaster-close {
 position: absolute;
 top: 0;
 right: 0;
 width: auto;
 height: auto;
 padding: 5px 10px;

 svg {
  fill: var(--color-gray-dark);
  margin-right: 0;
 }
}

.toaster-header {
 font-weight: 700;
 line-height: 20px;
}

.toaster-body {
 overflow: hidden;
 text-overflow: ellipsis;
 white-space: nowrap;
}

.toaster-loader {
 top: 10px;
 right: 10px;
 width: 14px;
 height: 14px;
 position: absolute;
 clip: rect(auto, auto, auto, auto);

 --toaster-half-time-animation: unset;
 --toaster-time-animation: unset;

 animation-duration: 0.01s;
 animation-delay: var(--toaster-half-time-animation);
 animation-name: close-wrapper;

 &::before, &::after {
  content: '';
  display: block;
  width: 10px;
  height: 10px;
  border: 2px solid var(--color-gray-middle);
  border-radius: 7px;
```

```
    position: absolute;
    clip: rect(0px, 7px, 14px, 0px);
    animation-iteration-count: 1;
    animation-fill-mode: forwards;
    animation-timing-function: linear;
   }

   &::before {
    animation-duration: var(--toaster-half-time-animation);
    animation-name: left-spin;
    animation-delay: var(--toaster-half-time-animation);
   }

   &::after {
    animation-duration: var(--toaster-time-animation);
    animation-name: right-spin;
   }
  }
}

@keyframes appearance {
  from {
    transform: translateX(110%);
  }

  to {
    transform: translateX(0);
  }
}

@keyframes right-spin {
  from {
    transform: rotate(-180deg);
  }
  to {
    transform: rotate(180deg);
  }
}

@keyframes left-spin {
  from {
    transform: rotate(0deg);
  }
  to {
    transform: rotate(180deg);
  }
}

@keyframes close-wrapper {
  to {
    clip: rect(0px, 7px, 14px, 0px);
  }
```

```
}
```

File: src/components/timer/timer.tsx

```
import { useState, useEffect } from 'react';
import classNames from 'classnames';

// styles
import styles from './timer.module.css';

type Props = {
  deadline: number;
  redTimer?: number;
  showDays?: boolean;
  showHours?: boolean;
  showMinutes?: boolean;
  showSeconds?: boolean;

  onEnd?: () => void;
};
export const Timer = ({
  deadline,
  showDays,
  showHours,
  showMinutes,
  showSeconds,
  redTimer,
  onEnd,
}: Props) => {
  const [days, setDays] = useState(0);
  const [hours, setHours] = useState(0);
  const [minutes, setMinutes] = useState(0);
  const [seconds, setSeconds] = useState(0);
  const [msLeft, setMsLeft] = useState(0);

  useEffect(() => {
    const interval = setInterval(() => {
      const time = deadline - Date.now();

      if (deadline && time <= 0) onEnd?.();

      setDays(Math.floor(time / (1000 * 60 * 60 * 24)));
      setHours(Math.floor((time / (1000 * 60 * 60)) % 24));
      setMinutes(Math.floor((time / 1000 / 60) % 60));
      setSeconds(Math.floor((time / 1000) % 60));
      setMsLeft(time);
    }, 100);
    return () => clearInterval(interval);
  }, [deadline]);
  return (
    <div
      className={classNames(styles.timer, {
        [styles['alert']]: deadline && redTimer && msLeft >= 0 && msLeft <= redTimer,
```

```
    })}
  >
    {(days > 0 || showDays) && (
      <span className={classNames(styles.days)}>{`0${Math.max(days, 0)}`.slice(-2)}:</span>
    )}
    {(hours > 0 || showHours) && (
      <span className={classNames(styles.hours)}>{`0${Math.max(hours, 0)}`.slice(-2)}:</span>
    )}
    {(minutes > 0 || showMinutes) && (
      <span className={classNames(styles.minutes)}>{`0${Math.max(minutes, 0)}`.slice(-2)}:</span>
    )}
    {(seconds > 0 || showSeconds) && (
      <span className={classNames(styles.seconds)}>{`0${Math.max(seconds, 0)}`.slice(-2)}</span>
    )}
  </div>
 );
};
```

File: src/components/timer/timer.module.css

```
.timer {
 display: flex;
 align-items: center;

 span {
  display: flex;
  align-items: center;
 }

 &.alert {
  span {
   animation: alert 1.5s infinite;
  }
 }

 .days {

 }

 .hours {

 }

 .minutes {

 }

 .seconds {

 }
}

@keyframes alert{
```

```
0% {
 color: var(--color-red)
}
/* 25% {
 color: rgba(230, 57, 70, 0.8);
} */
50% {
 color: rgba(230, 57, 70, 0.6);
}
/* 75% {
 color: rgba(230, 57, 70, 0.8);
} */
100% {
 color: var(--color-red);
}
}

@-webkit-keyframes alert{
 0% {
 color: var(--color-red)
}
/* 25% {
 color: rgba(230, 57, 70, 0.8);
} */
50% {
 color: rgba(230, 57, 70, 0.6);
}
/* 75% {
 color: rgba(230, 57, 70, 0.8);
} */
100% {
 color: var(--color-red);
}
}
```

File: src/components/theme-toggler/theme-toggler.tsx

```
import React from 'react';

import styles from './theme-toggler.module.css';
import { useThemeContext } from '../../providers/theme/theme.context';
import { DARK_THEME } from '../../providers/theme/theme.consts';

export const ThemeToggler = () => {
 const { theme, toggleTheme } = useThemeContext();

 return (
  <div className={styles['toggle-wrapper']}>
   <input
    id="themeSwitcher"
    className={styles['toggle']}
    type="checkbox"
    onChange={toggleTheme}
```

```
      checked={theme === DARK_THEME}
    />
  </div>
 );
};
```

File: src/components/theme-toggler/theme-toggler.module.css

```
.toggle {
 appearance: none;
  width: 62px;
  height: 32px;
  display: inline-block;
  position: relative;
  border-radius: 50px;
  overflow: hidden;
  outline: none;
  border: none;
  cursor: pointer;
  transition: background-color ease 0.3s;

 border: 1px solid var(--color-border);

 transform: scale(0.65);

  &::before {
    content: url('https://shivanarrthine.com/public/images/icons/moon.svg');
    display: block;
    position: absolute;
    z-index: 2;
    width: 24px;
    height: 24px;
    background: var(--color-gray-dark);
    left: 4px;
    top: 3px;
    border-radius: 50%;
    text-indent: 4px;
    line-height: 32px;
    word-spacing: 37px;
    color: #fff;
    white-space: nowrap;
    transition: all cubic-bezier(0.3, 1.5, 0.7, 1) 0.3s;
  }

  &:checked{
    &::before {
      left: 32px;
      content: url('https://shivanarrthine.com/public/images/icons/sun.svg');
      background: var(--color-gray-light);
   }
  }
 }
```

File: src/components/plural/plural.tsx

```
type Props = {
  count: number;
  one?: string;
  few?: string;
  many?: string;
  other?: string;
};

export const Plural = ({ count, many, few, one, other }: Props) => {
  if (count === 1 || count === -1) return <span>{one}</span>;

  if ((count >= 2 && count <= 4) || (count <= -2 && count >= -4)) return <span>{few}</span>;

  if (count > 4 || count < -4) return <span>{many}</span>;

  return <span>{other}</span>;
};
```

ДОДАТОК Г

(обов'язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

РОЗРОБКА ІГРОВОГО WEB-ЗАСТОСУНКУ ДЛЯ РОЗВИТКУ ЛОГІЧНОГО МИСЛЕННЯ. ЧАСТИНА 1. ПРОЕКТУВАННЯ ТА ВІЗУАЛІЗАЦІЯ ІНТЕРФЕЙСУ ЗАСТОСУНКУ

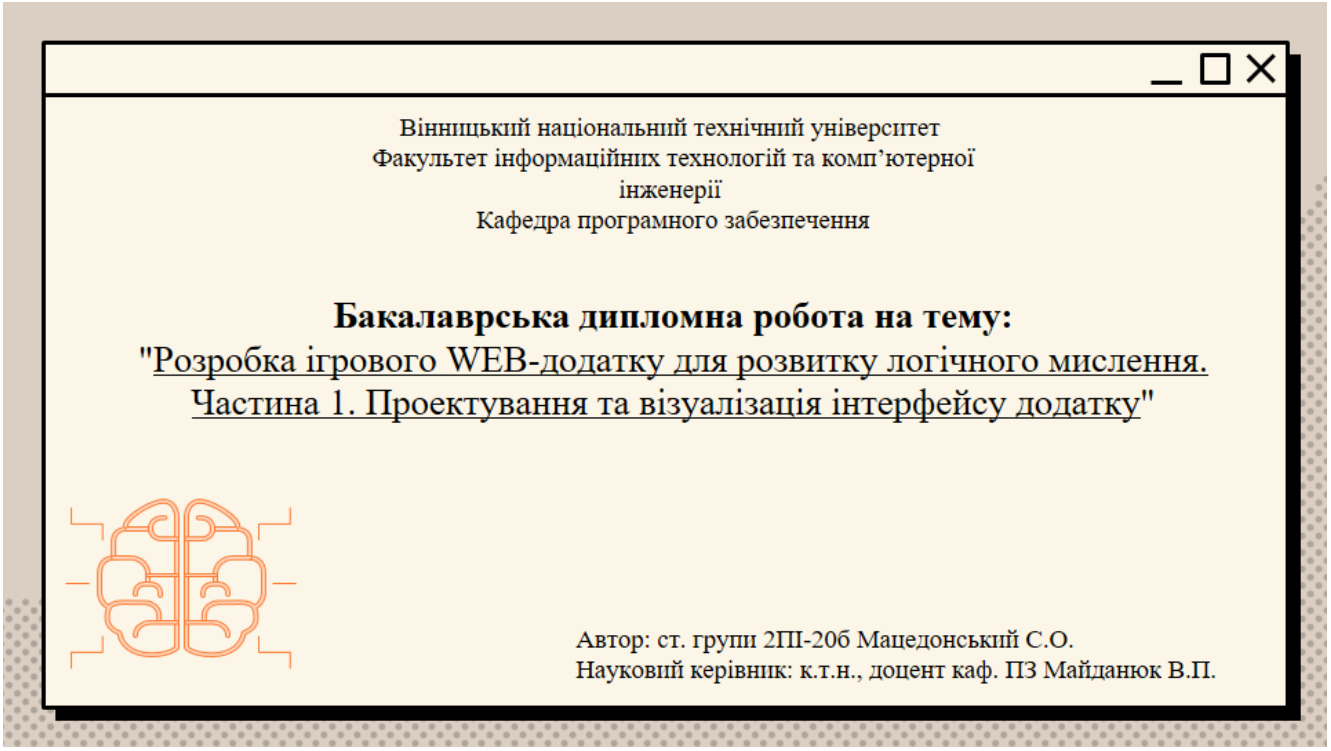


Рисунок Г.1 – Титульний слайд

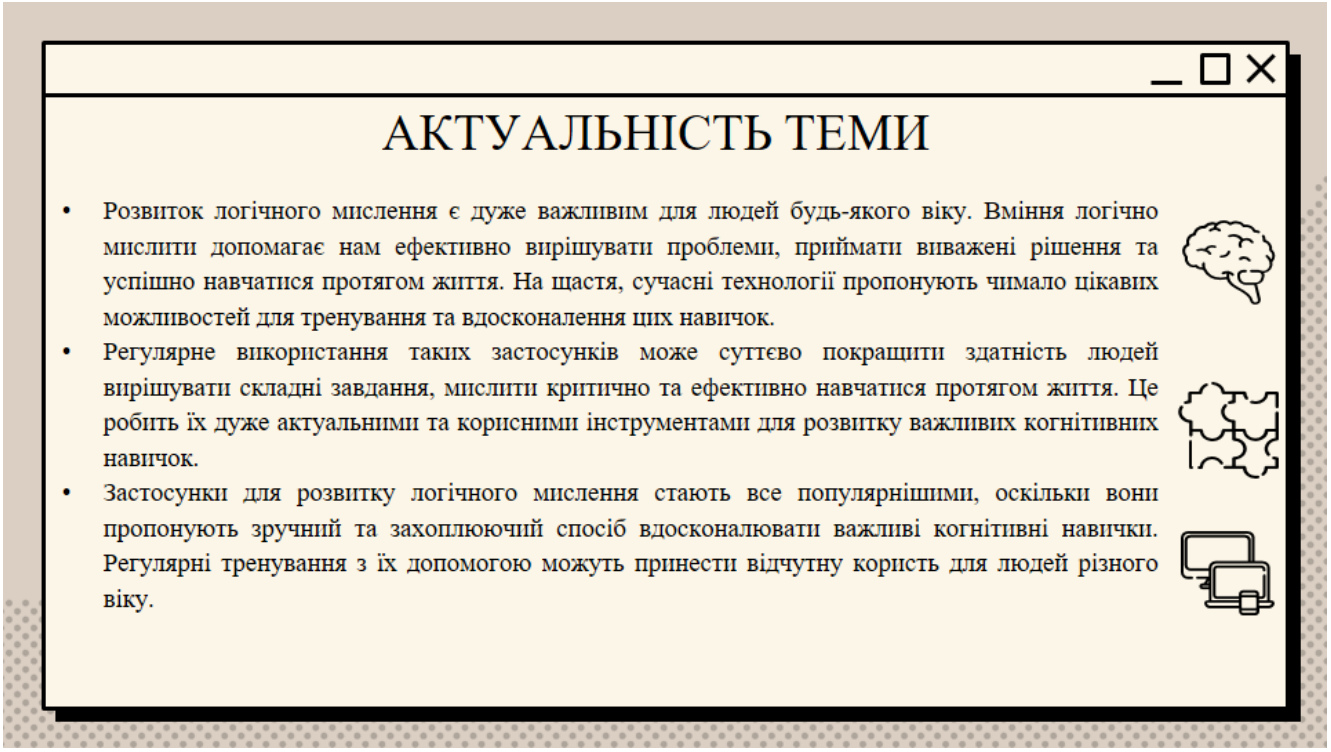


Рисунок Г.2 – Актуальність теми

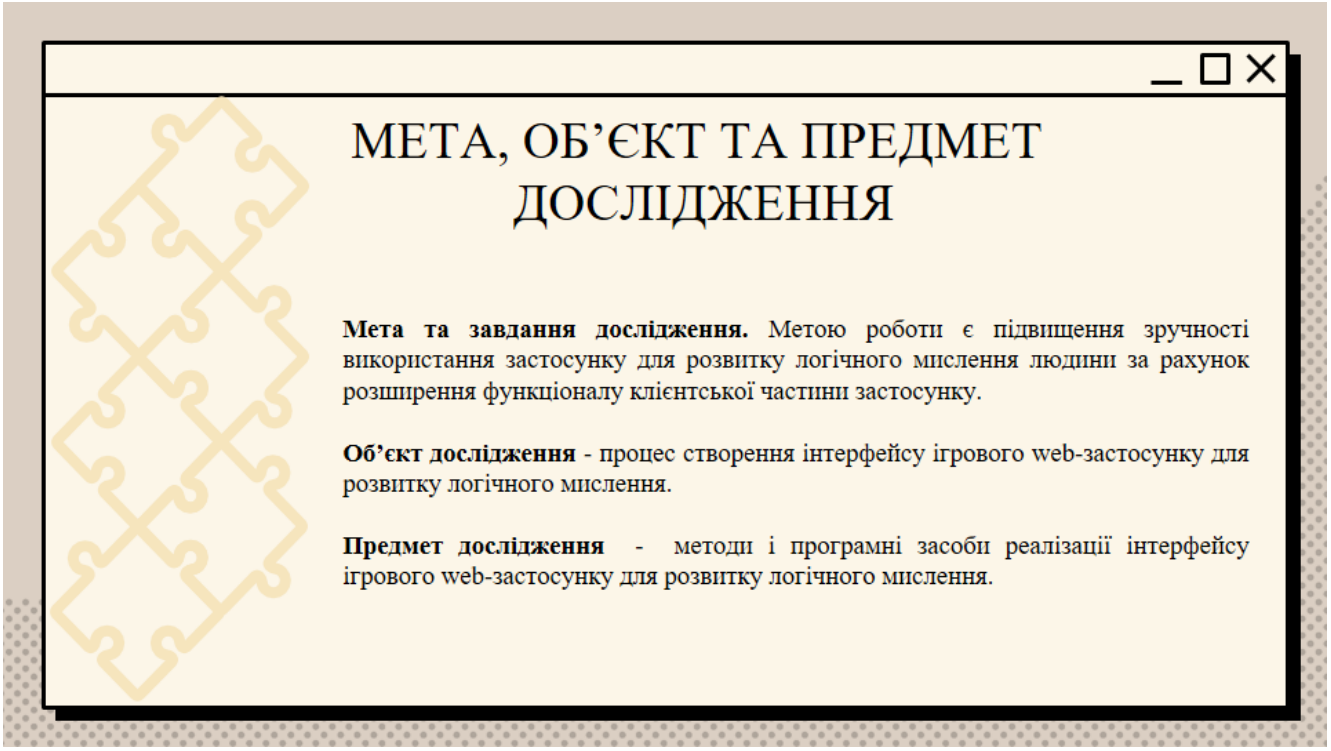


Рисунок Г.3 – Мета, об'єкт та предмет дослідження

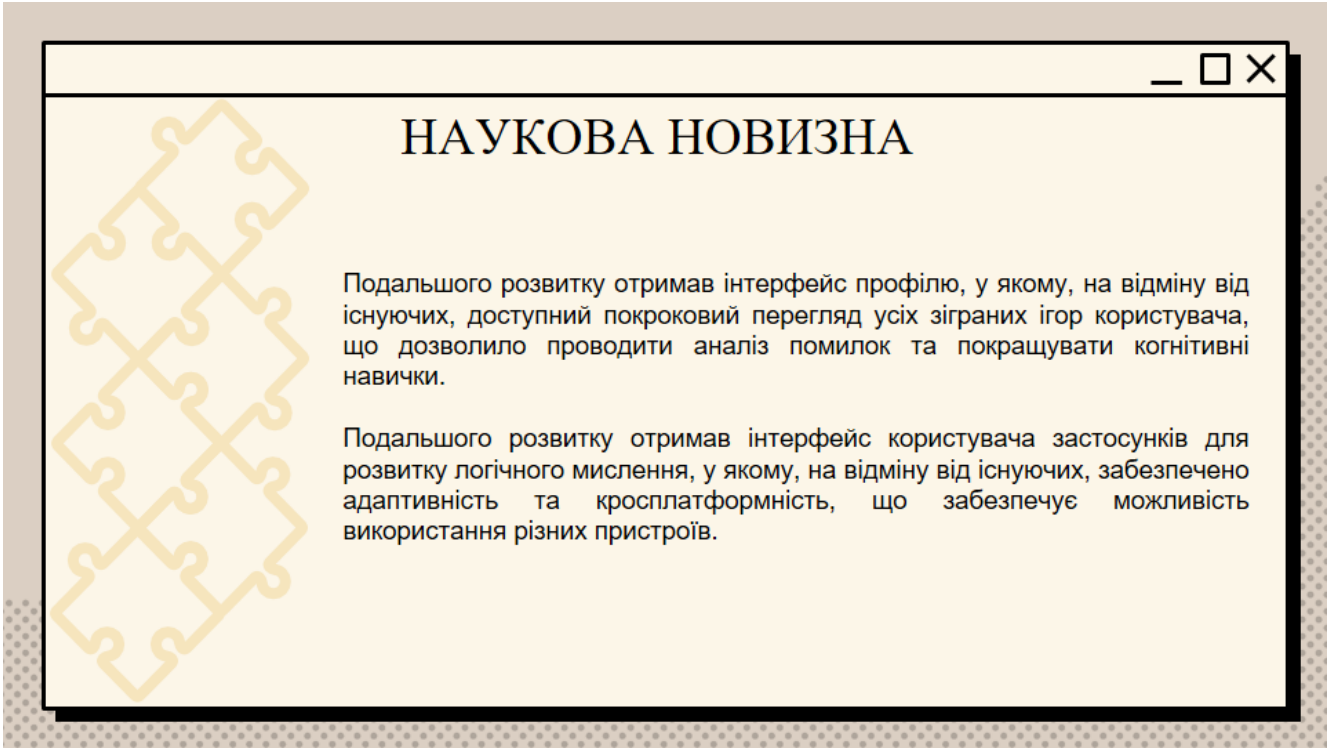


Рисунок Г.4 – Наукова новизна

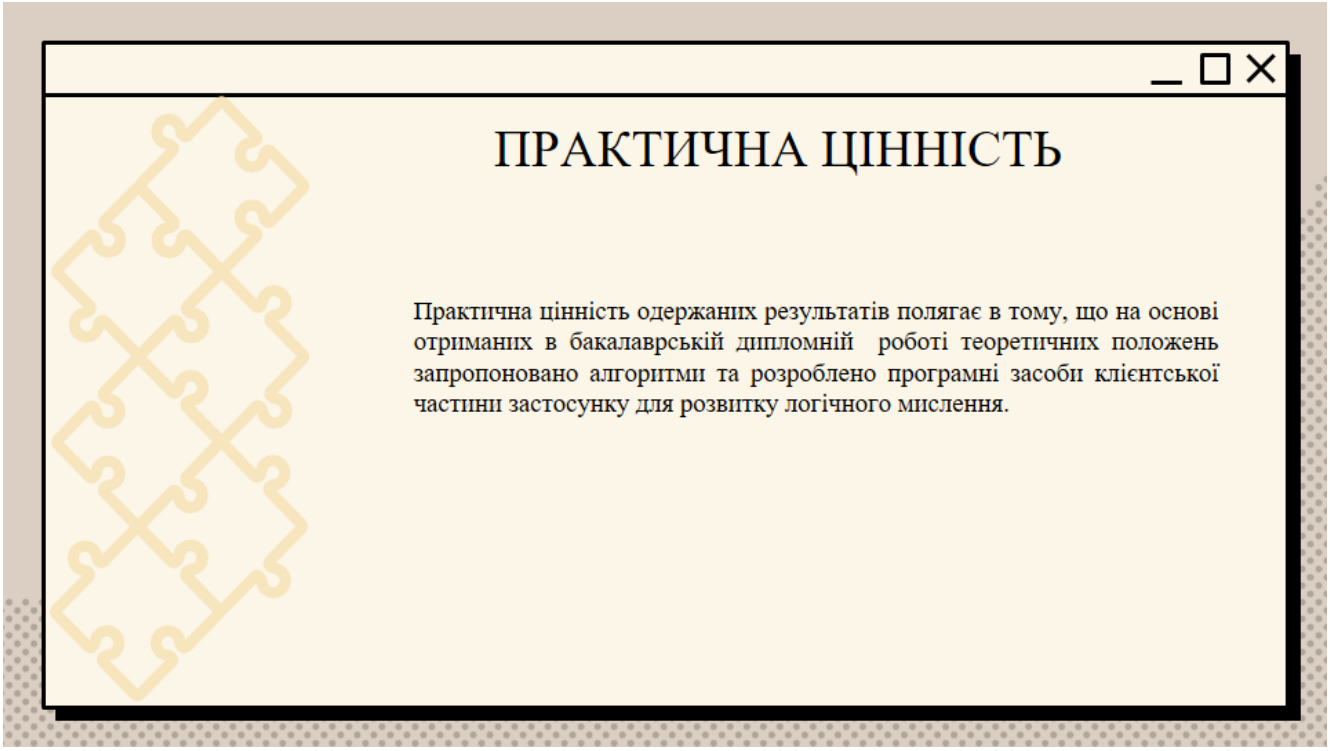
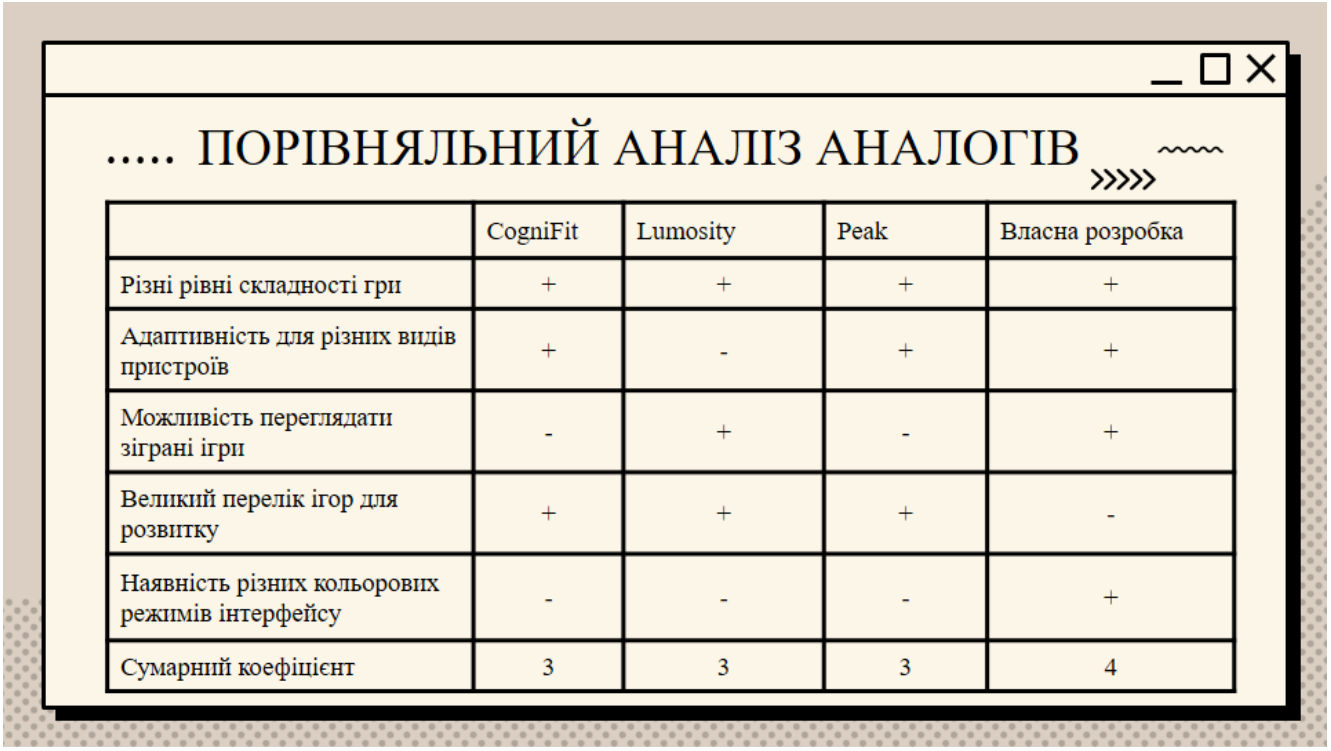


Рисунок Г.5 – Практична цінність

Рисунок Г.6 – Аналоги



	CogniFit	Lumosity	Peak	Власна розробка
Різні рівні складності гри	+	+	+	+
Адаптивність для різних видів пристроїв	+	-	+	+
Можливість переглядати зіграні ігри	-	+	-	+
Великий перелік ігор для розвитку	+	+	+	-
Наявність різних кольорових режимів інтерфейсу	-	-	-	+
Сумарний коефіцієнт	3	3	3	4

Рисунок Г.7 – Порівняльний аналіз аналогів

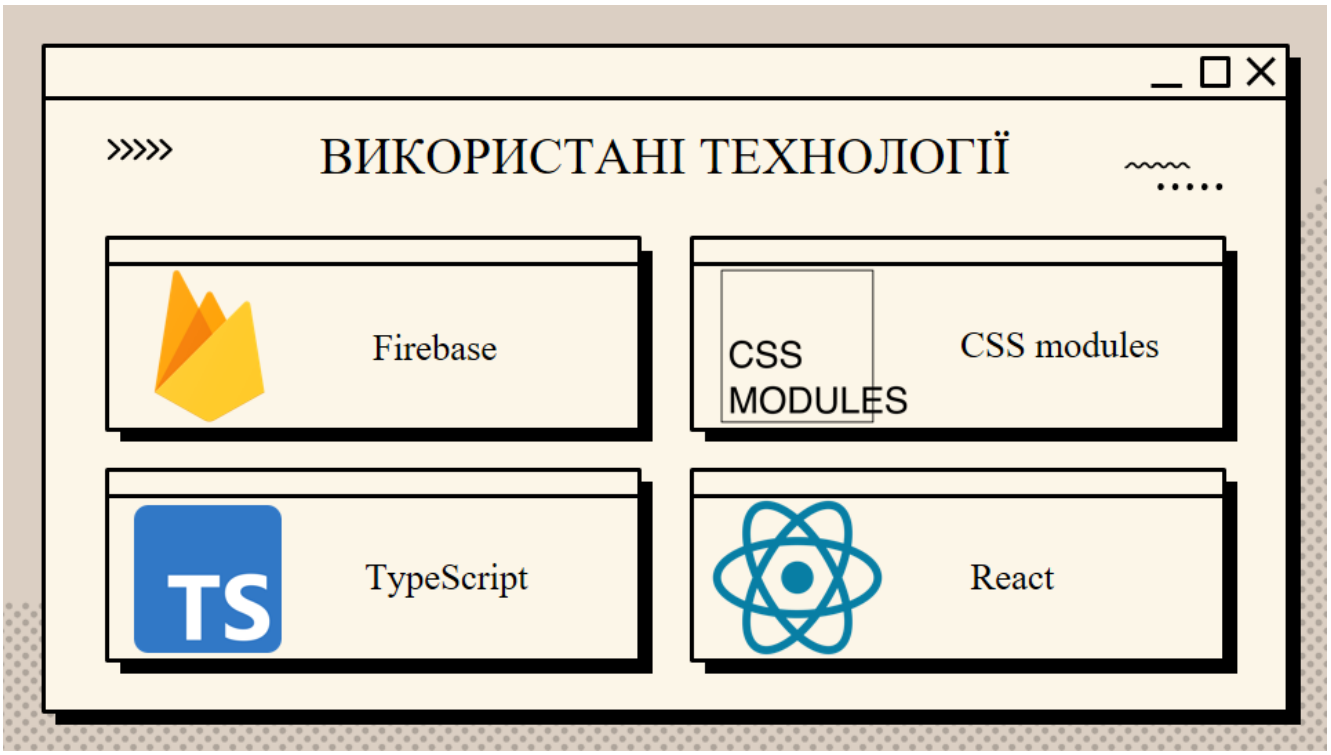


Рисунок Г.8 – Використані технології

Рисунок Г.9 – Модель застосунку

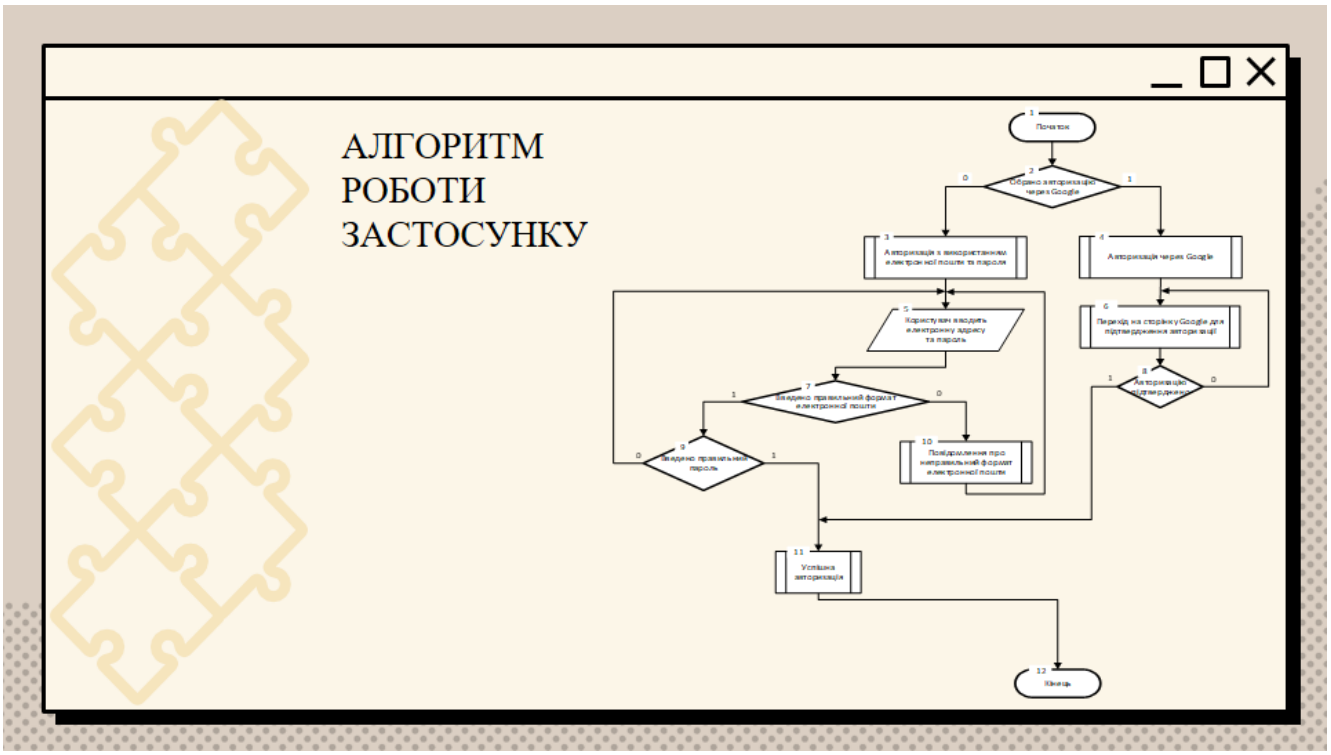


Рисунок Г.10 – Алгоритм роботи застосунку

Рисунок Г.11 – Блок-схема алгоритму перегляду гри

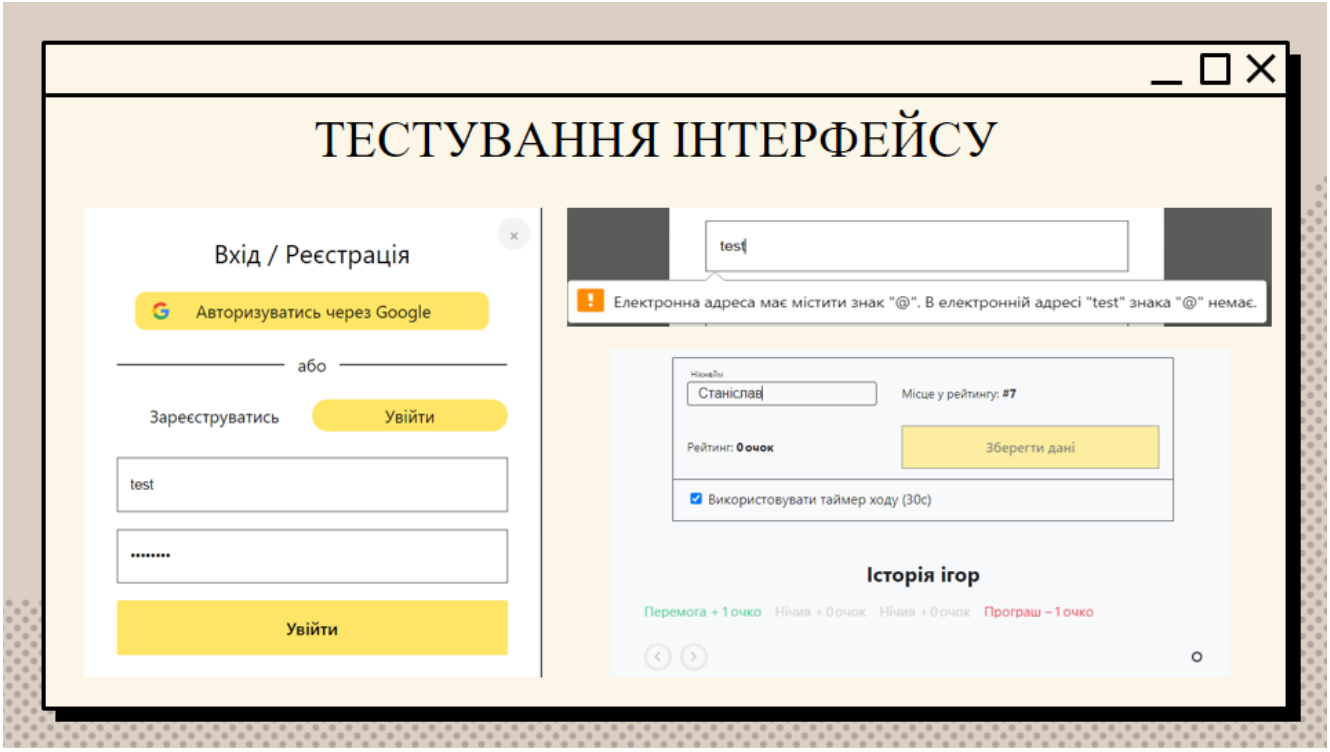


Рисунок Г.12 – Тестування інтерфейсу

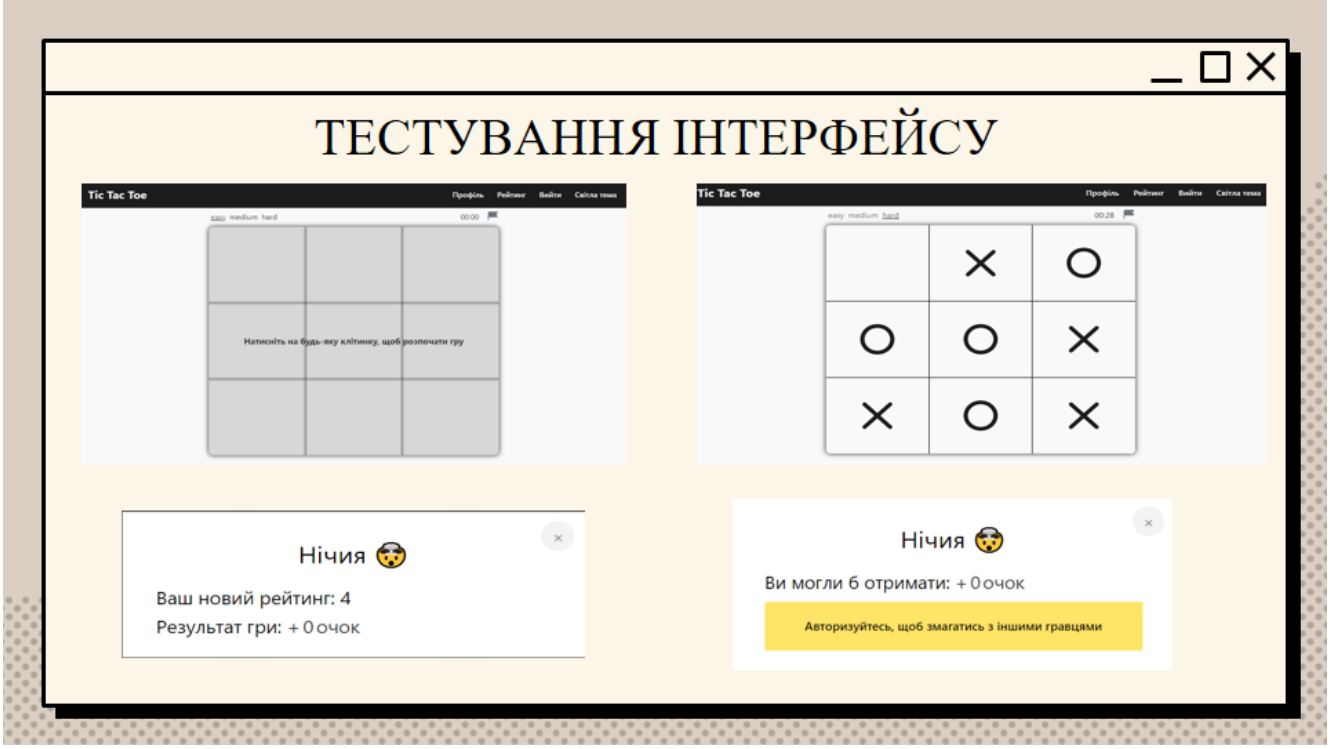


Рисунок Г.13 – Тестування інтерфейсу

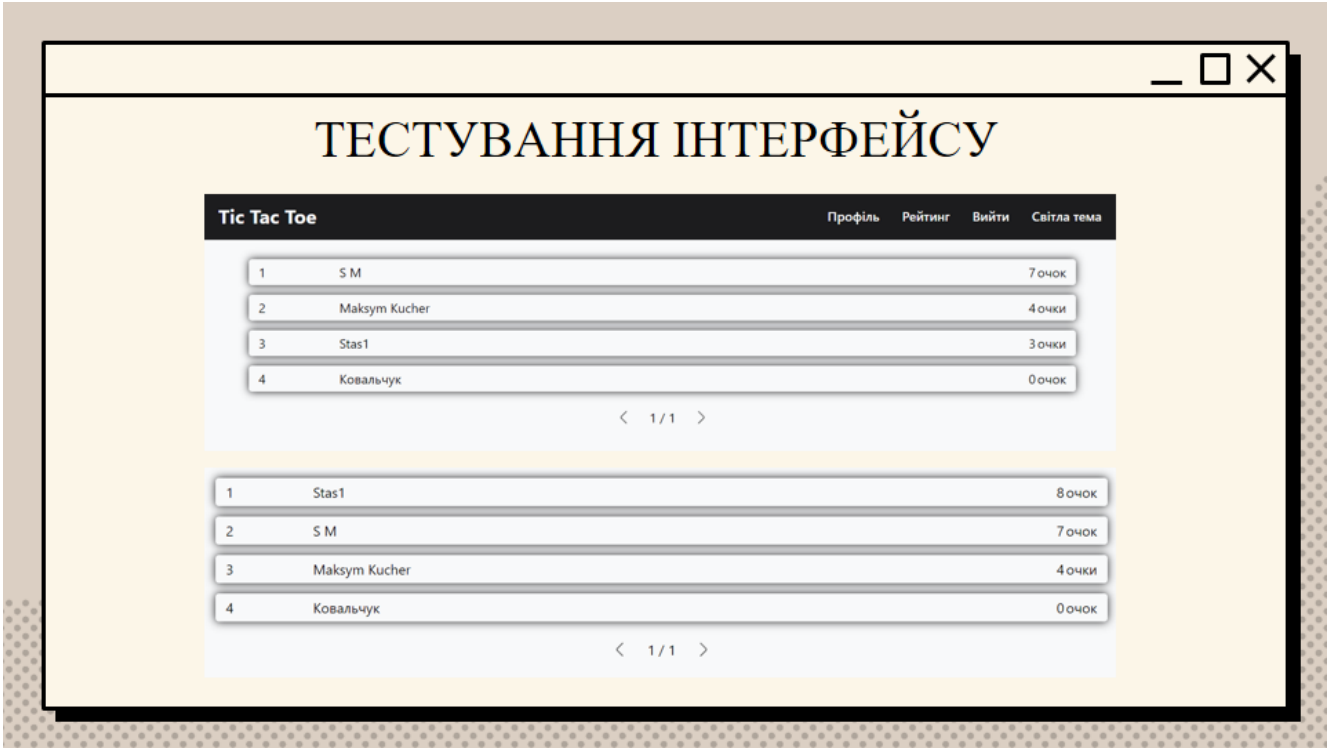


Рисунок Г.14 – Тестування інтерфейсу

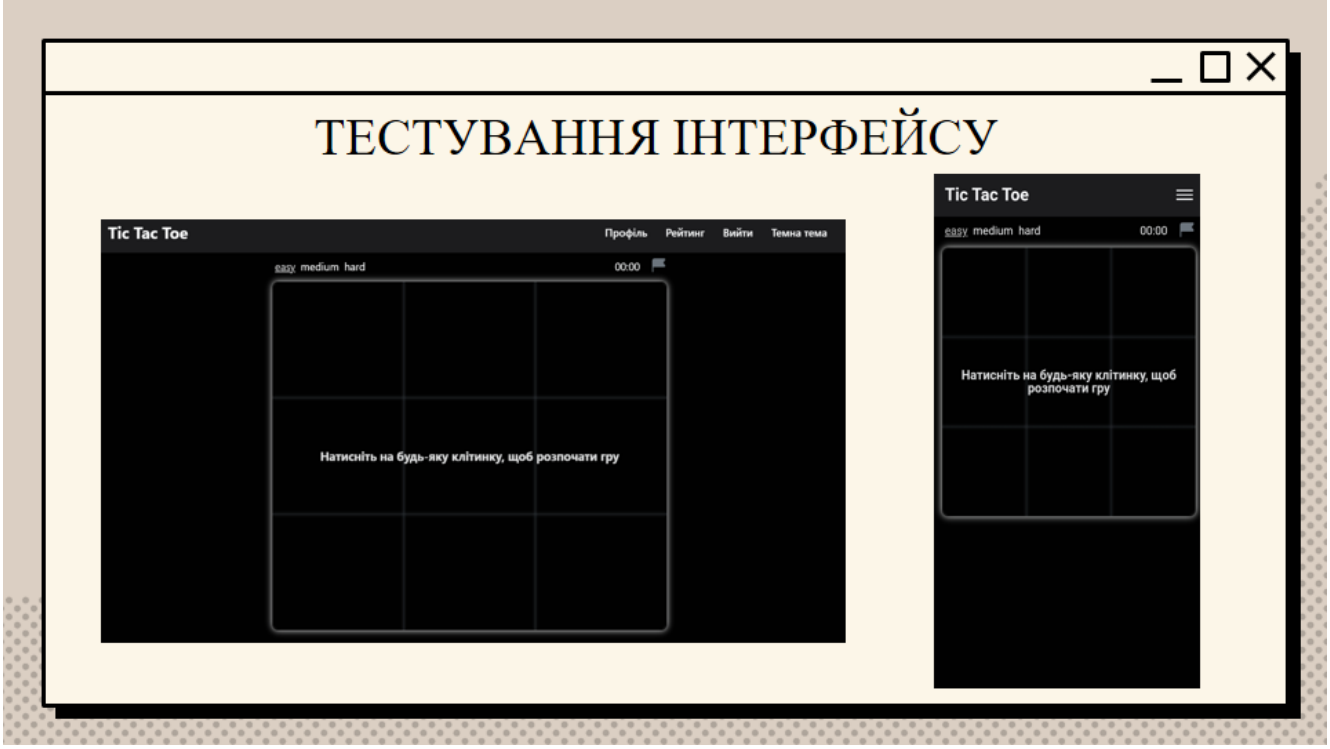


Рисунок Г.15 – Тестування інтерфейсу

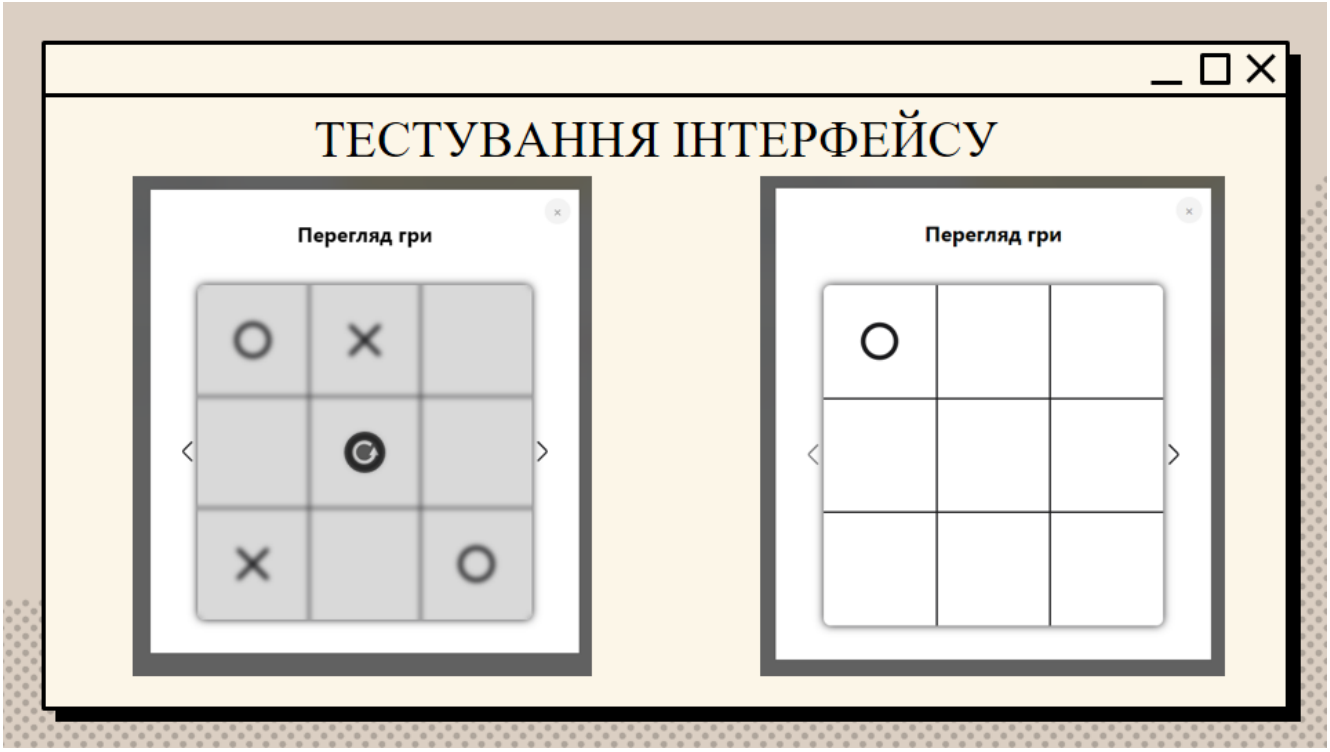


Рисунок Г.16 – Тестування інтерфейсу

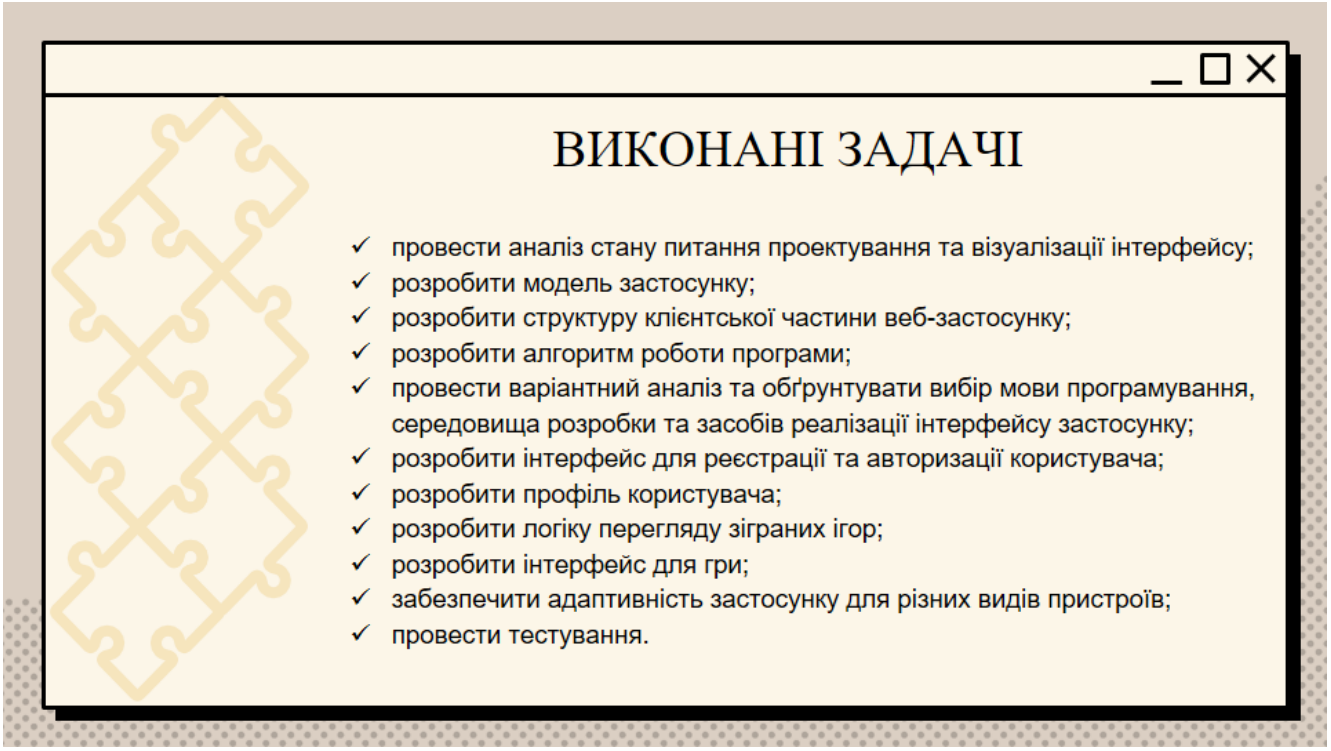


Рисунок Г.17 – Виконані задачі

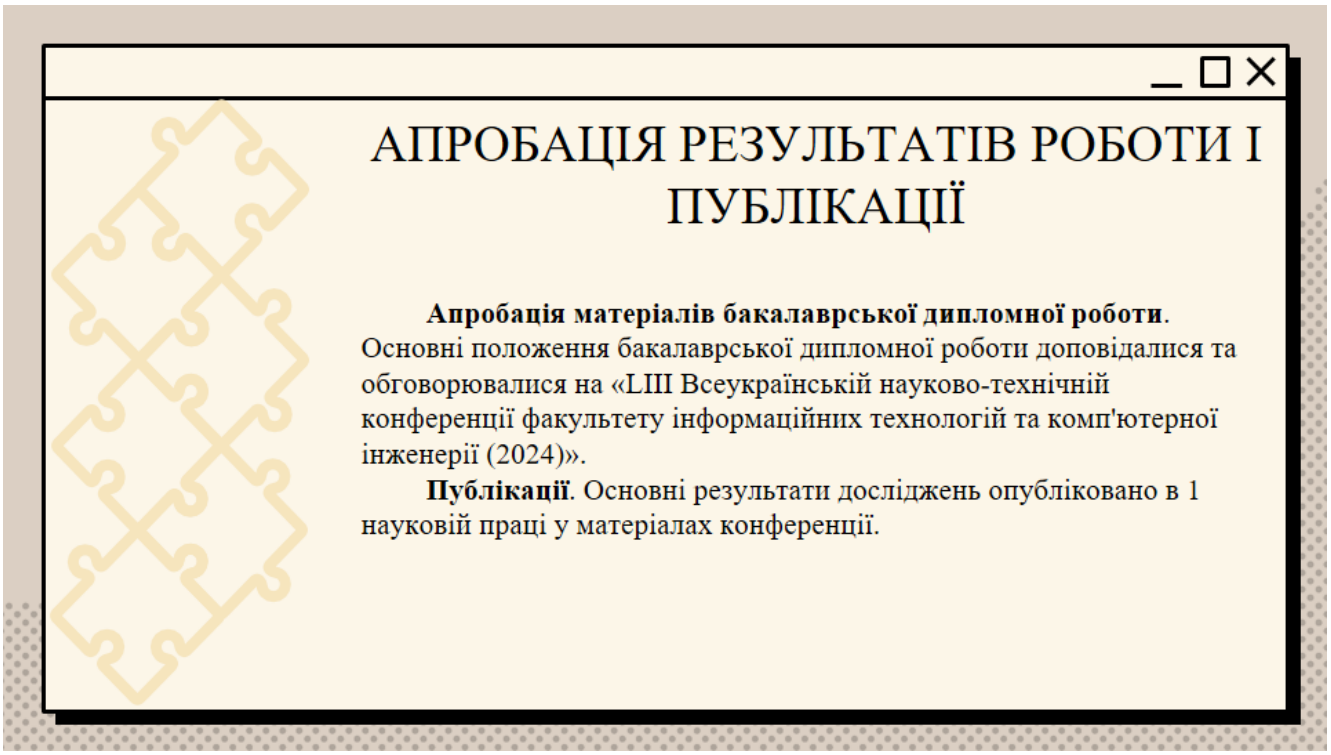


Рисунок Г.18 – Апробація результатів роботи і публікації

Рисунок Г.19 – Завершальний слайд