НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

«На правах рукопису»
УДК _____

До захисту допущено:
Завідувач кафедри
_____ Сергій СТІРЕНКО
«___»_____20__ р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-професійною програмою «Комп'ютерні системи та мережі»

зі спеціальності 123 «Комп'ютерна інженерія»

на тему: «Онлайн-вікторина «Сто запитань»»

Виконав (-ла):
студент (-ка) ІІ курсу, групи ІО-11мп
Серебряков Роман Олександрович _____

Науковий керівник:
доц., к.т.н. доц.,
Долголенко Олександр Миколайович _____

Консультант:
проф., д.т.н. проф.,
Кулаков Юрій Олексійович _____

Рецензент:
доц., к.т.н. доц.,
Катін Павло Юрійович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2022 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Рівень вищої освіти – другий (магістерський)

Спеціальність – 123 «Комп'ютерна інженерія»

Освітньо-професійна програма «Комп'ютерні системи та мережі»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Сергій СТІРЕНКО
«___»_____20__ р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Серебрякову Роману Олександровичу

1. Тема дисертації «Онлайн-вікторина «Сто запитань»», науковий керівник дисертації к.т.н. доц. Долголенко Олександр Миколайович, затверджені наказом по університету від «09» листопада 2022 р. № 4115-с

2. Термін подання студентом дисертації _____

3. Об’єкт дослідження: інтелектуально-розважальна вікторина у вигляді веб-додатку_____

4. Вихідні дані: _____

5. Перелік завдань, які потрібно розробити: огляд існуючих рішень, вибір технологій для реалізації поставленого завдання, створення клієнтської та серверної частини програми_____

6. Орієнтовний перелік графічного (ілюстративного) матеріалу: 31 рисунок

7. Орієнтовний перелік публікацій: _____

8. Консультанти розділів дисертації:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	проф. Кулаков Ю.О.		

9. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів дисертації	Примітка
1	Огляд існуючих рішень	05.09.2022 - 11.09.2022	
2	Планування розробки	12.09.2022 - 18.09.2022	
3	Вибір технологій для розробки	18.09.2022 - 21.09.2022	
4	Реалізація системи	22.09.2022 - 30.10.2022	
5	Тестування системи	31.10.2022 - 05.11.2022	
6	Розрахунки стартапу	06.11.2022 - 12.11.2022	
7	Оформлення документації	13.11.2022 - 30.11.2022	

Студент Серебряков Роман Олександрович

Науковий керівник дисертації Долголенко Олександр Миколайович

РЕФЕРАТ

Об’єктом досліджень магістерської дисертації є інтелектуально-розважальна онлайн-вікторина у вигляді веб-додатку. Даний проект дозволить весело провести дозвілля у компанії друзів, швидко відповідаючи на питання різної тематики. А використовуючи зручний конструктор питань, до гри можна додати і свої цікаві запитання для майбутніх ігор.

В епоху розвитку галузі відеоігор та серед різноманіття масштабних проектів завжди свою аудиторію знаходять невеликі цікаві проекти, особливо ті, які мають мультиплеєрну складову. Онлайн-вікторини є саме такими проектами, тож їх розробку можна завжди вважати актуальною.

Онлайн-вікторина, що розроблена в даній магістерській дисертації, заснована на механіці гри “Jeopardy!”, яка з’явилася у 60-х роках у США та була локалізована у багатьох країнах світу. Різні її версії, зокрема і комп’ютерна, досі є популярними серед людей будь-якого віку.

Додаток створений за допомогою фреймворку Sails.js, який у свою чергу заснований на Express.js та Socket.io, а також використовує Waterline ORM для роботи з базою даних. Програмний код написаний мовою JavaScript у середовищі WebStorm.

Ключові слова: онлайн, вікторина, веб-додаток.

ABSTRACT

The object of research of the master's thesis is an intellectual and entertaining online quiz in the form of a web application. This project will allow you to have fun in a company of friends, quickly answering questions on various topics. And using the convenient question designer, you can add your interesting questions to the game for the future.

In the era of development of the video game industry and among the variety of large-scale projects, small and interesting projects always find their audience, especially those that have a multiplayer. Online quizzes are just such projects, so their development can always be considered relevant.

The online quiz developed in this master's thesis is based on the mechanics of the game "Jeopardy!", which appeared in the 60s in the USA and was localized in many countries around the world. Its various versions, including the computer version, are still popular among people of any age.

The application is built using Sails.js framework, which is based on Express.js and Socket.io, and uses Waterline ORM to work with the database. The program code is written in the JavaScript language in the WebStorm environment.

Keywords: online, quiz, web application.

ЗМІСТ

ВСТУП

Відеоігри вже більш як півстоліття є невід'ємною частиною поп-культури людства. Майже кожна людина хоч раз у житті мала справу з відеоіграми. Починаючи з передвстановлених в операційну систему сапера, косинки та маджонга і закінчуючи останніми ААА-проектами від іменитих студій. Тому галузь відеоігор з кожним роком розвивається вражаючи нас все більш деталізованою графікою, продвинутою поведінкою штучного інтелекту, а також фізикою персонажів та оточення.

Ще у далеких 1950-х роках минулого століття почали з’являтися перші відеоігри. А після широкого розповсюдження персональних комп’ютерів та ігрових консолей відеоігри почали розвиватися ще стрімкіше, та охопили всі можливі жанри та напрямки. Наразі можна обрати гру на будь-який смак. З сильним цікавим сюжетом або веселим мультиплеєром, з цінником у половину мінімальної зарплатні або безкоштовну, десктопну або браузерну. Переважна більшість женеться за останніми проектами, які зможуть вразити своїм візуальним інтерфейсом та наявністю передових технологій, але є і інша категорія людей, яка шукає щось більш просте, знайоме та легке, щоб можна було запустити на найпростішому комп’ютері. Одним з таких напрямків є настільні ігри. Монополія, шахи, шашки, різні карткові ігри, -всі вони були розроблені аби при відсутності самої настільної гри можна було одним кліком запустити повністю адаптовані для миші та клавіатури їх комп’ютерні версії.

Така сама історія відбулась і з телевізійними іграми. Але якщо настільні ігри можна було купити у найближчому магазині, то потрапити у телепередачу було трохи складніше. «Хто хоче стати мільйонером», «Своя гра», «Що? Де? Коли?», - кожен хотів прийняти участь у цих телевікторинах, а тим паче разом зі своїми друзями. Таким чином і виник попит на відеоігри по телевізійним вікторинам, в яких можна було у режимі онлайн зіграти зі своїми друзями та позмагатись за титул найерудованішої людини у компанії. Піддавшись ностальгії я також вирішив створити вікторину, у якій в режимі онлайн можна

буде разом з друзями весело та цікаво провести вечір, відповідаючи на питання. А для того, щоб дану гру міг запустити кожен на майже будь-якому комп’ютері вона повинна бути браузерним веб-додатком.

Причин для розробок таких проектів зараз вистачає. По перше, завжди знайдуться люди, які полюбляють даний жанр та шукають нові проекти для вечірніх ігор у компанії. По друге, хоч наразі розвиток апаратного забезпечення дозволяє відеоіграм бути дуже реалістичними, але сучасні апаратні компоненти, м’яко кажучи, не є дешевими. Тож я вважаю, що саме зараз нового витка розвитку потребують браузерні ігри, які не вимагають сучасного «заліза» та дозволяють весело провести час з друзями. Браузерні багатокористувацькі ігри дозволяють знаходити аудиторію серед тих, хто не має потужного апаратного забезпечення, а також серед тих, хто не хоче встановлювати додаткове програмне забезпечення. По третє, вікторини є інтелектуально-розважальними іграми, які змушують наш мозок трохи подумати, або хоча б просто згадати якусь інформацію, що позитивно впливає на людину.

За основу онлайн-вікторини «Сто запитань» я обрав телевізійну гру “Jeopardy!”, яка ще у 60-х роках минулого століття підкорила американських телеглядачів, а з деяким часом була локалізована у багатьох країнах світу. Саме ви, можливо, знаєте її під назвою «Своя гра». Механіка гри є дуже цікавою, а головною фішкою її вже існуючих комп’ютерних онлайн версій є конструктор запитань, який дозволяє кастомізувати гру під свою компанію, створивши запитання на цікаві для усіх теми. Тож оглянувши існуючі рішення даної вікторини, спираючись на їх переваги та не недоліки, а також зібравши низку нових ідей я почав розробку онлайн-вікторини «Сто запитань».

РОЗДІЛ 1

ОНЛАЙН-ВІКТОРИНА ЯК ПОПУЛЯРНИЙ ЖАНР ІНДУСТРІЇ ВІДЕОІГОР

1.1. Історія виникнення та розвитку перших відеоігор

Сьогодні індустрія відеоігор є однією з найбільших розважальних індустрій у світі. Вона розвивалася десятки років наразі присутня в повсякденному житті майже кожної людини, від простих додатків на мобільних телефонах до ігрових продуктів для шоломів віртуальної реальності. Але на мою думку, вже у 80-х роках минулого століття відеоігри ступили на найважливішу сходинку у своєму розвитку з появою першої мережевої комп'ютерної гри.

Перші відеоігри почали з'являтися після Другої світової війни, використовуючи перепрограмовані комп'ютери, які були призначені для військових [1]. У 1950-х роках, на початку розвитку галузі, ігри розроблялися лише під одне апаратне забезпечення, оскільки тоді ще не існувало єдиного стандарту. Портування гри на іншу платформу було дуже складним тому багато продуктів просто видаляли через деякий час після їх використання.

Найпершу відеогру публічно продемонстрували на Канадській національній виставці у 1950 році. Нею стала "Bertie the Brain" - аркадна гра у хрестики-нулики. Її розробив канадський інженер Джозеф Кейтс на комп'ютері, висота якого становила 4 метри, а функціонал вичерпувався цією самою грою. Тому навіть після успіху на виставці, зібравши черги людей, які бажали хоч трохи пограти в цю гру, її довелося видалити для подальшого використання комп'ютера.

У 1951 році вже на Фестивалі Британії був показаний комп'ютер Nimrod від компанії Ferranti, який був 20 футів у ширину, 10 у глибину та 5 у висоту (рис. 1.1). Він був спроектований виключно для однієї гри Nim - математичної стратегії, в якій гравці по черзі видаляють предмети, які задані правилами гри. Оскільки головним завданням Ferranti було продемонструвати

свої досягнення у проектуванні комп’ютерних програм та апаратного забезпечення, а не розважити відвідувачів, то через три тижні даний комп’ютер вже було розібрано. Між тим все більше людей були зацікавлені в самому процесі гри, аніж у тому як був побудований даний апарат та як він працював. У тому ж 1951 році була розроблена комп’ютерна гра у шашки, особливістю якої було те, що вона була розроблена для комп’ютерів загального призначення, а не для конкретного апаратного рішення.

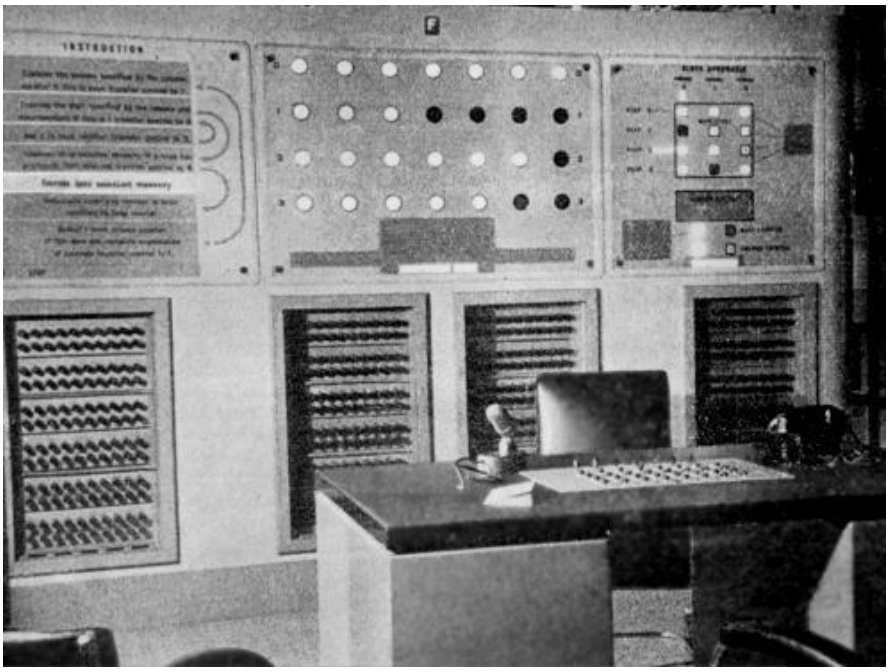

Рис. 1.1 – комп’ютер Nimrod від компанії Ferranti

У 1953 році вже почали з’являтися інтерактивні графічні ігри. Першою з них стала симуляція м’яча, який підстрибував, та яким потрібно було потрапити у спеціальний отвір. Автором гри став Олівер Альбер, студент Массачусетського технологічного інституту. А вже через рік у Мічиганському університеті була створена гра у більярд, в якій за допомогою маніпулятора потрібно було керувати кієм та забивати кулі до луз. Комп’ютер розраховував траєкторії руху куль до та після зіткнень, а при досягненні лузи вони зникали. Оскільки потрібно було демонструвати рух куль у реальному часі, комп’ютер 40 разів на секунду оновлював зображення на екрані. Гра була виконана на комп’ютері MIDSAC та була призначена для демонстрації потужності самого комп’ютера.

Першою грою, що не демонструвала можливості комп’ютера, а була створена як розважальний продукт, стала “Tennis for Two”, яку продемонстрували в 1958 році. Вона була створена Вільямом Хігінботамом, фізиком із Америки, на аналоговому комп’ютері Donner Model 30 для відвідувачів Брукгейвенської національної лабораторії. У перший же день вона стала найпопулярнішим “експонатом” лабораторії, який збирав біля себе черги з бажаючих пограти. Саме завдяки орієнтованості на розважальну складову дану гру вважають першим зразком тих відеоігор, якими вони є зараз.

Широкого поширення відеоігри зазнали у 1962 році з виходом “Spacewar!”. Головною особливістю даної гри була можливість до копіювання на інші комп’ютери. Завдяки цьому вона стала першою грою, яка могла бути розповсюджена на інші пристрої і використання якої не було обмежене апаратним забезпеченням, на якому вона була розроблена. Розробниками даного проекту стали співробітники Гарварду та MIT Уейн Війтанен, Мартін Гретц та Стів Рассел, а розробка велася на міні комп’ютері DEC PDP-1, придбаний MIT (рис. 1.2). Університет дозволив студентам та співробітникам використовувати даний апарат у вільний час для проектів, які не були безпосередньо пов’язані з роботою або навчанням.

Рис. 1.2 – комп’ютер DEC PDP-1

У кінці 60-х великої популярності стала набувати мова програмування BASIC, тож розробники ігор почали використовувати її для створення своїх продуктів. Головною перевагою такого кроку було те, що програму, створену на мові BASIC можна було запустити на різних типах апаратного забезпечення, що дозволяло розширити аудиторію. Саме в той період були випущені бінго, баскетбольний симулятор та “Space Travel”. Всі вони були написані саме мовою програмування BASIC.

З 1970-ми прийшла епоха аркадних ігор. Почали з’являтися перші аркадні автомати, а з ними і ігрові зали, досхочу набиті цими автоматами, на яких були встановлені ігри на будь який смак. Набував розвитку і домашній геймінг, оскільки саме тоді і з’явилась перша домашня ігрова консоль Magnavox Odyssey (рис. 1.3), яку як і сучасні ігрові консолі можна було підключати до телевізора. Альтернативою підключення до телевізора було використання растрового дисплею. Ціна даного виробу становила 100 доларів США, а за 3 роки було продано понад 350 тисяч екземплярів, чому посприяв порт аркадної гри “Pong”, яка стала хітом серед аркадних автоматів. Згодом домашні ігрові консолі набули ще більшої популярності з переходом на консолі, які передбачали використання картриджів. Дані консолі містили мікропроцесори, які зчитували інформацію з картриджів, тому просто змінюючи картриджі можна було запускати нескінченну кількість відеоігор.

Рис. 1.3 – перша домашня ігрова консоль Magnavox Odyssey

В той же час потроху почав зростати ринок персональних комп’ютерів. Наприкінці 1970-х років почали з’являтися відносно доступні моделі, такі як Commodore PET, Apple II, TCP-80 тощо. Вони мали вже передвстановлену низку різних відеоігор, а також мову програмування BASIC для створення власних продуктів. Програмісти поширювали написаний код за допомогою дискет, касет та картриджів, а ринок нестудійних проектів почав значно поповнюватися іграми від програмістів-любителів, які залюбки продавали власні ігри.

У 1980-х роках з’явилося нове покоління персональних комп’ютерів, серед яких найпопулярнішими були Commodore Amiga та Atari ST. У порівнянні з попереднім поколінням, вони вирізнялися більш якісними графічними та аудіозвуковими можливостями. Оскільки дані системи набули великої популярності, розробники ігор почали створювати проекти саме для цих систем. Але ці системи мали недоліки. Продукти випущені для даних систем було не так просто копіювати під інші системи, оскільки вони повинні були відповідати усім вимогам продукту. А вже у 1981 році IBM представила свій перший персональний комп’ютер (рис. 1.4), який використовував MS-DOS в якості операційної системи та відкриту апаратну архітектуру, завдяки чому у ньому можна було додавати, змінювати або оновлювати компоненти. Так з’явилось поняття сумісності з IBM PC, завдяки чому розробники відеоігор могли не створювати продукти під конкретну архітектуру окремого комп’ютера, а писати програми які відповідали специфікаціям сумісності з IBM PC. Тож протягом 80-х років почалося розширення здатностей даного комп’ютера. IBM представила відеоконтролери, які дозволяли відображення кольорової графіки, а після цього покращили і аудіозвукову складову за допомогою звукових карт, які забезпечували цифровий звук. Останнім етапом стала можливість підключення ігрових контролерів, ранніх версій сучасних геймпадів.

Рис. 1.4 – IBM Personal Computer

В той же час на ринку з’явився і Apple Macintosh. На відміну від IBM PC він мав більш закриту операційну систему, якою можна було керувати за допомогою графічного інтерфейсу. Apple Macintosh не мав такого значного успіху на ринку як сумісні з IBM PC комп’ютери, але налічував доволі велику кількість програмного забезпечення для Macintosh OS, серед яких були і відеоігри. А серед популярний студій розробників ігрових продуктів стали з’являтися відомі і по наш час Activision та Electronic Arts.

У 1983 році була створена перша мережева комп’ютерна гра для персонального комп’ютера IBM. Поява онлайн ігор поклала початок тієї індустрії, яка існує зараз, оскільки мультиплеєр є дуже важливою складовою кожного проекту. Майже кожен AAA-продукт, який зараз з’являється в ігровій індустрії є мультиплеєрним, або має окрему мултиплеєрну частину, відокремлену від основної сюжетної кампанії. Це напряму пов’язано зі збільшенням кількості годин які користувач, проведе у тій чи іншій відеогрі, оскільки грати з друзями набагато веселіше ніж самому.

Тож на початку 1980-х років відеоігри пройшли своє становлення та набули тих характеристик, які вони мають і по цей день. Звичайно, надалі розвивалися технології, які покращували якість зображення та фізику поведінки предметів чи персонажів, але концепція, або так би мовити формула вдалої відеогри вже була знайдена, і головною її складовою був мультиплеєр.

На мою думку, навіть знехтувавши візуальною складовою можна створити продукт, який не буде поступатись у цікавості та обов’язково знайде свою аудиторію, якщо додати до нього мультиплеєрну складову. А оскільки відсутність графічної складової значно знижує потреби у великій кількості ресурсів пристрою, то такий проект можна створити у формі веб-додатка. При цьому він буде орієнтований на більшу кількість аудиторії, оскільки не потрібно враховувати особливості програмного та апаратного забезпечення пристрою. Головне щоб на ньому був встановлений браузер. Такі проекти є привабливими для користувачів, які не бажають встановлювати додаткове програмне забезпечення, або не мають такої змоги. Тож спираючись на приведені вище міркування я вирішив розробити власну онлайн-вікторину у вигляді веб-додатку.

1.2. Вікторина як розважальна складова нашого дозвілля

Вікториною називають інтелектуальну гру, головне завдання якої полягає в тому щоб надати максимально правильну кількість відповідей на поставлені запитання, при цьому запитання зазвичай відносяться до різноманітних тем [2]. Вікторини можуть бути використані як у навчальних цілях, як наприклад для проведення зрізу знань з вивченої теми, так і у розважальних, наприклад в якості телешоу.

Перші використання слова “вікторина”, а саме англійського “quiz”, датуються 1780 роком. Його походження невідоме, але є здогадки, що воно походить від студентського сленгу. Спочатку воно означало “дивну, ексцентричну особу” або “жарт”. Вже пізніше воно почало бути пов’язане з “інтенсивним навчанням”, в з середини 19-го століття стало означати “тест” або “екзамен”.

Вікторини можуть охоплювати найрізноманітніші теми. Від перевірки загальних знань до конкретних розділів конкретної теми. Вікторини поширені у пабах, на телебаченні, у навчальних установах, у вигляді настольних домашніх ігор, індивідуальних тестів на “який ти кіт” та онлайн ігор. До речі,

найбільшою вікториною, яка навіть зареєстрована у книзі рекордів Гіннеса є "Quiz for Life", що відбулася на Flanders Expo Halls у Генті у 2010 році. Тоді вона зібрала 2280 учасників.

У навчанні вікторини часто використовуються як розминка перед новою темою або для швидкого повторення вивченого матеріалу. Вона зазвичай містить низку нескладних запитань, на кожне з яких надається декілька варіантів вибору. Вікторини на визначення рис характеру або психологічних особливостей людини зазвичай супроводжуються ключем, який при підрахунку результатів показує які питання відображають конкретні якості людини. Такі тести дуже часто зустрічалися у журналах по типу “Cosmopolitan”. А з розвитком Google Forms та інших схожих програм створювати тести на різні буденні теми стало дуже просто і швидко.

1.3. Веб-додаток - найкращий варіант для онлайн вікторини

Серед найголовніших переваг веб-додатків можна виділити наступне:

- не потребують встановлення додаткового програмного забезпечення, для запуску потрібна тільки наявність браузера;
- не потребує оновлення;
- не займає додаткової пам’яті на диску;
- запустити веб додаток можна з будь-якого пристрою, якщо на цьому пристрої наявний доступ до Інтернету.

Ці переваги дозволяють веб-додаткам суттєво збільшувати аудиторію людей, які їх використовують, оскільки швидкість та простота доступу до них є максимально високими [3].

Оскільки онлайн-вікторина не потребує значних ресурсів комп’ютера, то найкращим варіантом я вважаю виконання даного продукту саме у вигляді веб-додатку.

1.4. Телевікторина “Jeopardy!” та огляд існуючих рішень

У 1964 році у США з’явилася одна з найпопулярніших вікторин у світі - “Jeopardy!”. Вона розпочала свій шлях на телебаченні, але згодом набула широкої популярності та була розповсюджена у найрізноманітніших формах [4]. Як телепроект “Jeopardy!” була локалізована у багатьох країнах по всьому світу. У 1994 році вона навіть з’явилась у пострадянському просторі під назвою “Своя гра”. Правила гри були доволі простими. Гравці по черзі обирали тему та номінал питання, після чого ведучий задавав питання, після чого гравці на швидкість намагались відповісти на задане питання.

Але дивитись телепередачу хоч і було весело, але багатьом хотілось пограти у неї самому. При цьому далеко не кожного запрошували на телебачення щоб взяти участь у грі. Тож з самого початку виходу телевікторини у продажу почали з’являтись настільні ігри на основі гри “Jeopardy!”, а у 1990-х роках з’явились книжки, які містили набори питань телевікторини.

Jeopardy! також була випущена у вигляді численних відеоігор, які охоплювали декілька поколінь апаратного забезпечення. Що ж там казати, ігри по цій франшизі виходять і по цей день. Наприклад, зараз у Playstation Store можна придбати Jeopardy! для Sony Playstation 4 за $19.99. Користувачі персональних комп’ютерів вперше побачили “Jeopardy!” у 1987 році ще на Apple II та Commodore 64.

Але якщо ми просто відкриємо браузер та введемо у пошуковий рядок “вікторина Jeopardy грати онлайн”? Відкинувши Android версію, оскільки нас цікавлять браузерні веб-додатки, ми побачимо продукт від JeopardyLabs, який фактично не є онлайн грою. Обравши тему гри, або створивши гру з власними питаннями у доволі зручному конструкторі питань (рис. 1.5) її можна увімкнути тільки на одному комп’ютері.

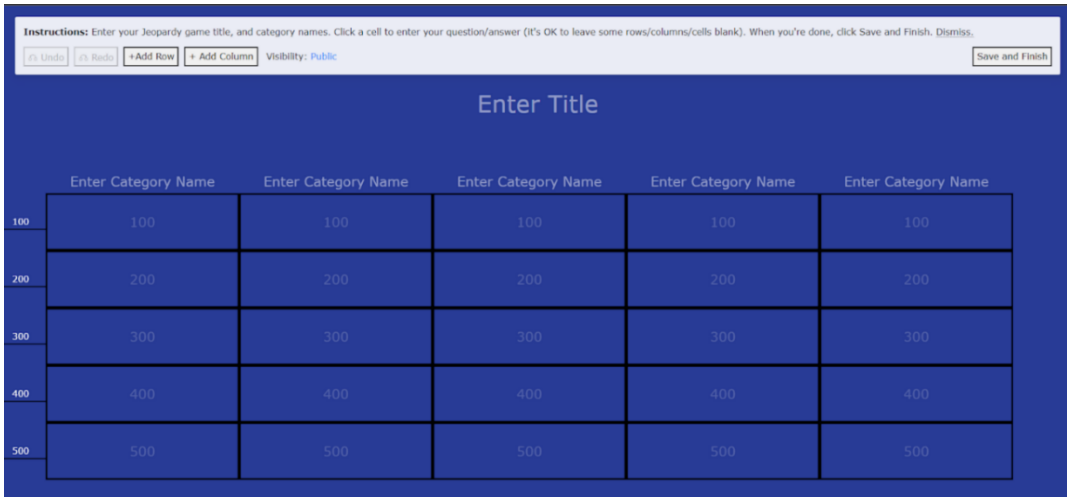

Рис. 1.5 – Конструктор питань “Jeopardy!”

Але увімкнувши саму гру, попередньо обравши кількість гравців, ми бачимо сітку питань, яка відкривається тільки на одному пристрої. Після цього є можливість обирати питання, зачекати поки гравець надасть відповість, а потім натиснувши на “space” ми можемо побачити правильну відповідь, після чого зарахувати або ні надану відповідь гравцю (рис. 1.6).

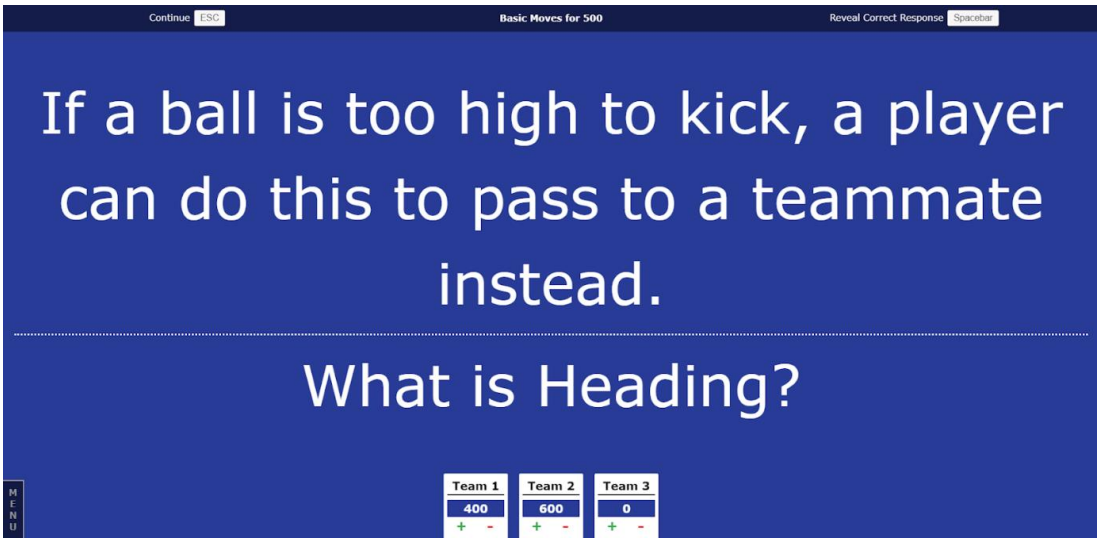

Рис. 1.6 – Процес гри “Jeopardy!”

Але головною проблемою стає те, що екран гри потрібно поширювати між гравцями, якщо вони не знаходяться в одній кімнаті. Також конструктор питань має тільки текстовий вид запитання, тобто до нього не можна додати ні зображень, ні відео. Також гра ведеться одним гравцем, тобто інші гравці ніяким чином не допущені до геймплею. Тільки один гравець або ведучий виконує перевірки наданих відповідей, обирає хто має відповідати та нараховує або знімає бали.

Конструктор питань хоч і не має варіативності у виборі типу запитання, але є доволі зрозумілим та має всі інструменти для побудови сітки запитань. А сама гра, на мою думку, являє собою ідеальний приклад настільної гри в електронному варіанті з найпростішим функціоналом, що включає всі необхідні інструменти для створення та проведення гри.

Якщо ж пошуковий запит трохи змінити та пошукати саме “Свою гру” замість “Jeopardy!”, то можна знайти повноцінну онлайн вікторину “SIGame”. Вона має лобі, в якому можна створити окрему кімнату, або приєднатися до вже створеної. В цій кімнаті і запускається гра, до якої можуть доєднатися реальні люди. Принцип гри є наступним. Спочатку потрібно обрати ведучого, який буде перевіряти правильність наданої відповіді. У випадку якщо цього не зробити, ведучим стане комп’ютер, а гравцям доведеться у текстовому форматі вводити відповідь. При цьому зарахування відповіді відбудеться лише при повному співпадінні. Тож перед кожною грою потрібно обирати гравця, який стане ведучим, і який замість того щоб грати, буде сидіти позіхати, перевіряючи відповіді інших гравців. Це очевидний мінус. Конструктор питань у даній грі є окремим додатком (рис. 1.7), який потрібно встановлювати додатково, але кількість запропонованих наборів питань є доволі великою для вибору.

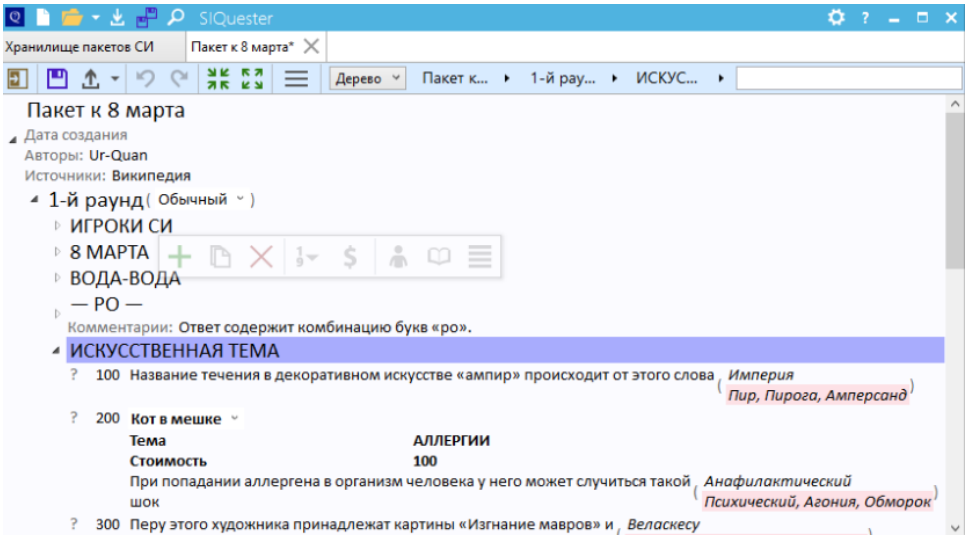

Рис.1.7 - Конструктор питань “SIGame”

Інтерактивності додає надана кожному гравцю кнопка, яку потрібно натиснути для того щоб відповісти на питання. Адже перший, хто натисне

буде відповідати, а інші гравці матимуть шанс це зробити тільки якщо перший гравець помилиться. Після перевірки відповіді, гравцю нараховуються бали за питання при правильній відповіді, або ж знімаються, якщо відповідь була невірною. Також гравці під час гри можуть вести розмову у текстовому чаті. Сам процес гри наведений на рис. 1.8.

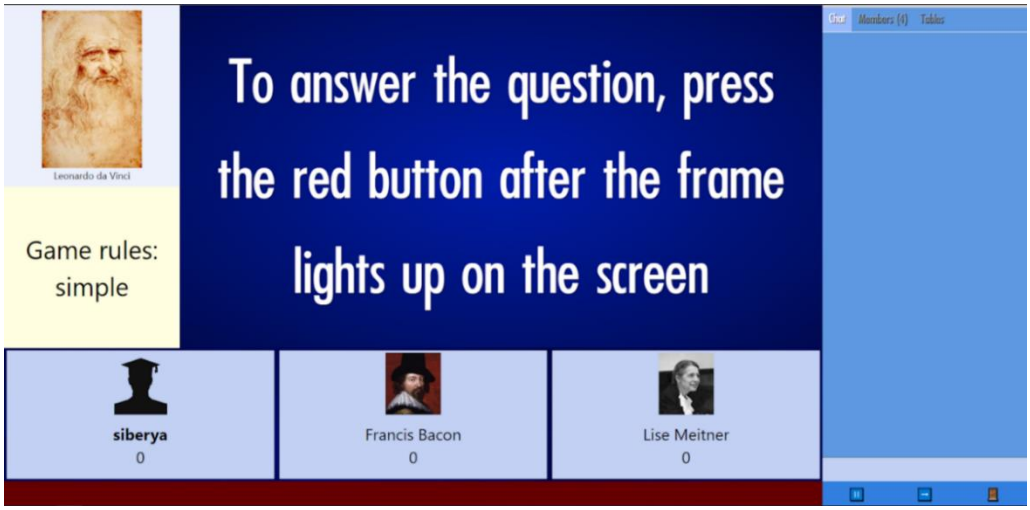

Рис. 1.8 – Процес гри “SIGame”

Підсумувавши вищесказане отримаємо наступну таблицю:

Таблиця 1.1

Недоліки існуючих рішень

Гра	Недоліки
Jeopardy	Потребує гравця, який буде самостійно взаємодіяти з додатком та адмініструвати весь процес гри.
	Немає мультиплеєрної складової, усі користувачі повинні знаходитись в одному приміщенні або ж ведучий має транслювати процес гри через сторонні додатки.
	Питання складаються лише з текстової частини, не можна додати до запитання медіа файли.
	Гравці можуть відповідати лише по черзі. Немає ігрової механіки, яка відповідала б за визначення гравця, що буде першим відповідати на питання.
	Немає індивідуального профілю гравця

Продовження таблиці 1.1

Гра	Недоліки
	Після надання неправильної відповіді інші гравці не зможуть відповісти на те саме питання, оскільки правильна відповідь буде показана на екрані для усіх гравців одночасно.
SiGame	Потребує ведучого при виборі усної гри. На початку гри завжди потрібно обирати людину, яка не буде грати, буде сидіти і перевіряти правильність наданих відповідей та адмініструвати гру.
	Гра без людини-ведучого потребує письмової відповіді, яка прописана для кожного питання як вірна і не може відрізнятися ні на один символ.
	Набори питань потрібно шукати окремо, стандартні набори не є дуже цікавими, а також їх не можна розширити.
	Конструктор питань є окремим додатком, до того ж його потрібно встановлювати.

ВИСНОВОК ДО ПЕРШОГО РОЗДІЛУ

Починаючи з 80-х років минулого століття багатокористувацькі онлайн-відеоігри є дуже популярними, незалежно від жанру. Серед інтелектуально-розважальних ігор великим попитом користуються онлайн-вікторини. Для даного жанру відеоігор найкращим варіантом виконання є веб-додаток.

Оглянувши існуючі рішення веб-додатків, заснованих на вікторині “Jeopardy!” та проаналізувавши переваги та недоліки даних проектів, було вирішено, що програмний продукт, який буде розроблятися в рамках даної магістерської дисертації повинен мати:

- онлайн складову, тобто мультиплеєр;
- вбудований та варіативний конструктор питань;
- автоматичне адміністрування гри, що включає перевірку наданих відповідей та нарахування балів, що відкине необхідність кожного разу обирати ведучого;
- варіативний вибір окремих категорій питань для гри, а не заздалегідь скомпонований набір питань;
- єдину базу питань, яку зможуть адмініструвати окремі гравці, що матимуть доступ до неї.

РОЗДІЛ 2

ВИБІР ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ

2.1. Вибір фреймворку для створення проекту

В наш час вже існує доволі велика кількість інструментів, які допомагають програмістам у створенні власних продуктів. При цьому їхня кількість з кожним роком все росте. З'являються фреймворки для різних мов програмування з різним функціоналом в залежності від потреб. Вони максимально спрощують та надають додаткові можливості при побудові програм. Але який з них обрати для виконання даного проекту?

Звичайно, обирати фреймворк потрібно той, який орієнтований на мову програмування, якою ви володієте і якою маєте намір створювати проект. Для створення веб-додатків найкращим варіантом, на мою думку, є мова Javascript з використанням програмної платформи Node.js. Вона дозволяє підключати різноманітні зовнішні бібліотеки для розробки програмного забезпечення, використовувати Javascript як на серверній, так і на клієнтській стороні проекту. А висока продуктивність та крос-платформеність роблять Node.js найкращим варіантом для розробки веб-додатків.

Але виконання проекту лише за допомогою Node.js є дуже затратним в плані часу, тому для спрощення роботи з ним існує велика кількість веб-фреймворків [5]. Найпопулярнішими з них є:

- AdonisJS;
- Nest.js;
- Express.js;
- Koa.js;
- Sails.js.

Тож розглянемо кожен з них більш детально.

1. AdonisJS - популярний Node.js фреймворк, доступний для роботи на всіх операційних системах [6]. По своїм можливостям він є дуже схожим на Laravel, одного із найпопулярніших PHP-фреймворків. Через це AdonisJS

користується попитом серед PHP-розробників, які бажають перейти на платформу Node.js. Серед особливостей AdonisJS можна виділити потужну ORM яка дозволяє виконувати SQL-запити, підтримку No-SQL баз даних, простий конструктор для створення швидких запитів до баз даних та доволі гарну систему безпеки. AdonisJS має багатофункціональний рівень маршрутизації, оскільки можна створити маршрут для груп, а також для субдоменів. Контролери винесені у окремі файли, зв'язок маршруту з якими можна вказати за допомогою роутера. Даний фреймворк має вбудований завантажувач файлів, валідатор схем, а також шаблонізатор AdonisJS (рис. 2.1), який відкидає потребу шукати та встановлювати сторонній.

Рис. 2.1 – Приклад використання вбудованого шаблонізатора AdonisJS

2. Nest.js - інструмент для розробки масштабованих професійних серверних додатків [7]. Будучи написаним на Typescript, він поєднує у собі елементи об'єктно-орієнтованого та функціонального програмування. Nest.js також має модулі для підключення різноманітних баз даних, таких як MySQL, PostgreSQL, MongoDB. Готові додатки на Nest.js легко тестувати, завдяки дуже чіткій та структурованій архітектурі, яка складається з контролерів, модулів та провайдерів. Тому на початку розробки додаток можна розділити на мікросервіси та працювати з кожним з них окремо, а вже потім об'єднати все в одну систему. Завдяки модульній архітектурі даний фреймворк дозволяє

використовувати різноманітні бібліотеки. Всередині Nest.js використовує такі фреймворки як Express, надає безпосередньо API цих фреймворків розробнику, при цьому забезпечуючи набагато вищий рівень абстракції. Це дає розробникам свободу використовувати незліченну кількість сторонніх модулів, які доступні для основної платформи.

3. Express вже давно користується довірою серед розробників [8]. Він, якщо так можна сказати, є одним с дідів-першопрохідців Node.js фреймворків. Express.js був та досі залишається одним з найшвидших та найпростіших фреймворків та ідеально підходить для створення невеликих додатків, оскільки має на початку невеликий об’єм базового функціоналу. Побудувати найпростіший веб-додаток за допомогою Express.js можна буквально у декілька строк, виконавши імпорт фреймворку, створити за його допомогою об’єкт додатку, прописати базовий маршрут та дати команду додатку прослуховувати порт (рис. 2.2). Для більш складних додатків розробник може встановлювати різноманітні додаткові модулі для вирішення тієї чи іншої проблеми. Що стосується баз даних, то Express.js має функціонал роботи з No-SQL базами даних. Розробники зазвичай використовують разом з Express.js такі інструменти як MongoDB, Angular.js, ну і звичайно ж Node.js, який часто називають MEAN software stack як абревіатуру по першим літерам цих інструментів. Express.js надає невеликий надійний функціонал для створення односторінкових програм або веб-сайтів, при цьому фреймворк не прив’язаний до використання конкретної ORM або системи шаблонів. Завдяки відкритим можливостям у використанні будь-яких інструментів можна побудувати структуру додатку, яка буде ідеально підходити розробнику.

```
const express = require('express')
const app = express()
const port = 3000

app.get('/', (req, res) => {
  res.send('Hello World!')
})

app.listen(port, () => {
  console.log(`Example app listening on port ${port}`)
})
```

Рис. 2.2 – Приклад побудови найпростішого додатку за допомогою Express.js

4. Koa.js є доволі сучасним продуктом від розробників Express.js [9]. Він повинен був усунути всі недоліки свого старшого брата та пропонувати більш широкий спектр можливостей для розробки програмних продуктів. Головною особливістю Koa.js стало використання генераторів, тобто функцій які могли бути призупинені на будь-якому етапі роботи, що дозволяє продуктивно тестувати готові додатки. Також даний фреймворк працює без використання функцій зворотнього виклику та має потужну систему обробки помилок. Додаток, створений за допомогою Koa складається з масиву middleware-функцій, що виконуються у вигляді стеку за запитом. Koa є схожим на такі фреймворки як Ruby's Rack та Connect та включає методи для типових завдань, таких як узгодження вмісту, швидке оновлення кешу, підтримка проксі-сервера та редіректів. Незважаючи на вміст достатньо великої кількості корисних функцій, Koa не займає багато місця, оскільки більшість middleware-функцій є опціональними.

5. Sails.js - популярний фреймворк, основою для якого є Express.js та Socket.io [10]. Він використовує шаблон MVC та може бути використаним для створення додатків як малого, так і великого розміру, а також обробляти великі об'єми даних. Завдяки вбудованому функціоналу для роботи з сокетами, даний фрейморк є ідеальним варіантом для створення веб-чатів у

реальному часі, багатокористувацьких ігрових продуктів та будь-яких інших real-time додатків. Також Sails.js має вбудований функціонал, який зв'язує між собою MVC компоненти та будує структуру додатку. Розробнику просто необхідно структурувати окремі елементи та розкласти їх у призначені директорії. Архітектура Sails.js є схожою на Ruby on Rails - легендарний фреймворк для мови Ruby, який вже багато років підтримується та розширюється спільнотою. Sails.js також має вбудовану технологія Waterline ORM, яка використовується для зв'язку з різними базами даних. Тобто використовуючи один і той самий синтаксис Waterline дозволяє працювати з будь-якою із підтримуваних нею базами даних.

Виконавши детальний огляд популярних Node.js фреймворків можна зробити висновок, що для виконання даної магістерської дисертації найкраще всього підходить Sails.js. Він має вбудований функціонал для роботи з сокетами, які безперечно будуть використовуватися у проекті, та зручну систему для створення компонентів MVC-шаблону, на основі якого і буде функціонувати проект. Вкупі зі зручним вбудованим Waterline ORM та докладною документацією по використанню Sails.js є найкращим варіантом серед інших Node.js фреймворків для виконання поставленого завдання.

2.2. Веб-сокети як засіб комунікації у режимі реального часу

Веб-сокети вперше з'явилися у 2010 році в Google Chrome 4 та наразі є найбільш поширеним способом комунікації у режимі реального часу [11]. Вони використовуються у веб-чатах, мультиплеєрних відеоіграх, у документах для одночасного редагування, а також додатках, що працюють з геолокацією. Веб-сокети являють собою дуплексний канал для спілкування між клієнтом та сервером, який додається до вже встановленого TCP-з'єднання. Така технологія дозволяє встановити зв'язок між клієнтом та сервером з мінімальною затримкою, що дозволяє створювати так звані "real-time communication" додатки. Встановлене за допомогою веб-сокетів з'єднання можна тримати будь-який час для очікування дій від клієнта або

сервера. Встановлення зв’язку за допомогою сокетів відбувається у два кроки за допомогою так званого «рукостискання» між сервером та клієнтом (браузером). Браузер, за допомогою спеціальних заголовків надсилає запит на сервер про встановлення з’єднання за допомогою протоколу WebSocket і якщо приходить відповідь, таке з’єднання стає дійсним (рис.2.3).

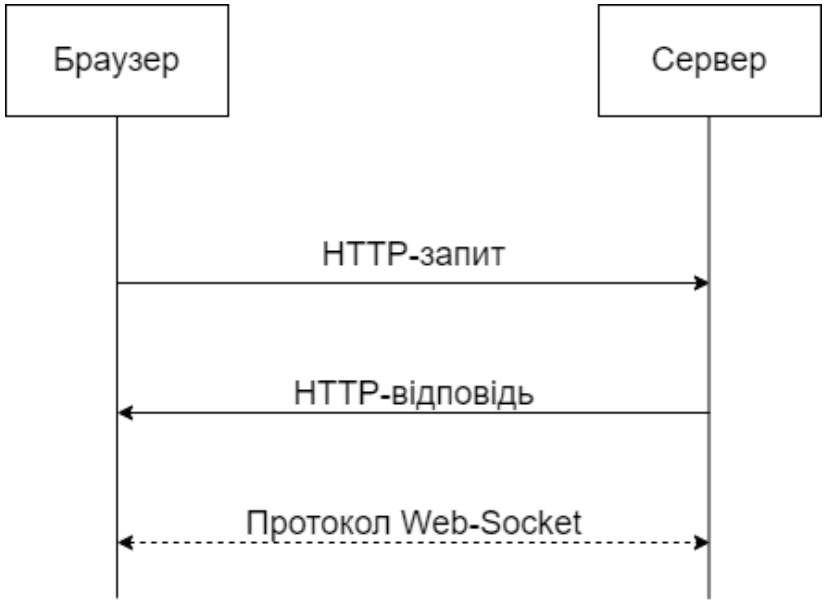

Рис. 2.3 – Встановлення з’єднання за допомогою протоколу Web-Socket

Для взаємодії гравців у розроблюваній онлайн-вікторині також будуть використовуватися веб-сокети. Обраний у попередньому підрозділі фреймворк Sails.js використовує для роботи з сокетами бібліотеку Socket.io. Дана бібліотека включає в себе весь функціонал веб-сокетів, обернений user-friendly обгорткою у вигляді простого зрозумілого синтаксису. Socket.io також має дуже чітку та зрозумілу документацію, яка допомагає не тільки знайти необхідний функціонал, а фактично навчає працювати з сокетами з нуля на конкретних прикладах, а також надає низку завдань для кращого освоєння. Серед функціоналу, який надає бібліотека Socket.io головними перевагами є:

- розсилка повідомлень одразу всім під’єднаним користувачам, у випадку з веб сокетами доводиться відправляти повідомлення кожному користувачу окремо;
- має вбудований балансувальник навантаження;

- автоматично перепідключає з’єднання у разі його розриву;
- максимально зрозуміле API для роботи.

Головним же недоліком Socket.io є досить значне використання трафіку при роботі, тому що всі вищевказані переваги потребують додаткових ресурсів. Якщо ж використовувати технологію веб-сокетів окремо, без бібліотеки Socket.io, маємо значну економію ресурсів, але ж і значну трату часу на написання базових операцій. Тож, на мою думку, використання бібліотеки Socket.io в проекті є досить виправданим.

2.3. MVC як шаблон для побудови додатку

Архітектурний шаблон, який буде використовуватись для проектування та розробки додатку є MVC – Model View Controller (рис. 2.4).

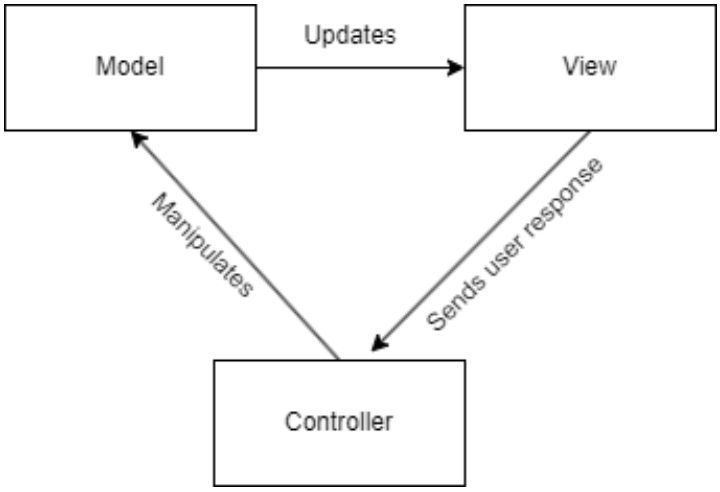

Рис. 2.4 – Шаблон MVC (Model View Controller)

Даний шаблон передбачає поділ додатку на 3 складові [12]:

- модель - складова, яка задає структуру даних;
- контролер - складова, яка маніпулює даними за допомогою моделей;
- представлення - складова, яка відповідає за візуальну частину, тобто відображення даних, які передав їй контролер.

Обраний у підрозділі 2.1 фреймворк Sails.js максимально чітко використовує даний шаблон. Він навіть надає директорії, куди потрібно додавати окремі компоненти для побудови додатку. Так, наприклад, у корені

проекту початково знаходиться директорія “views”, а директорія “api” має піддиректорії “models” та “controllers”. Завдяки цьому зручність у побудові проекту значно збільшується, оскільки не потрібно явно задавати зв’язки між цими компонентами, вони умовно задані системою директорій, а практично за допомогою вбудованого функціоналу самого фреймворку.

Основною перевагою використання трикомпонентного шаблону MVC є якраз таки трикомпонентність, тобто розмежованість. Саме вона дозволяє легко орієнтуватися у системі і знаходити помилки за дуже короткий час. Також дана розмежованість дозволяє розробляти та налагоджувати окремі компоненти незалежно один від одного, і, наприклад після тестування, об’єднувати все в одне ціле.

2.4. Клієнт-серверна архітектура

Веб-додаток, який розробляється у даній магістерській дисертації, буде побудований на архітектурі клієнт сервер. Дана архітектура передбачає поділ на 2 складові [13]:

- клієнта, який передає запити на сервер для подальшої обробки;
- сервера, який отримує запити від клієнта, обробляє їх, та відправляє відповідь.

У ролі клієнта зазвичай виступає користувач, а у якості сервера системне обладнання у вигляді одного або цілої системи комп’ютерів, яка є у багато разів потужнішою за клієнта. Така архітектура дозволяє виносити усю логіку додатку для подальшої обробки на сервер, тим самим знімаючи навантаження з клієнта. Якщо клієнтів, які одночасно звертаються до сервера буде декілька, то сервер сформує з них чергу та буде обробляти запити один за одним або по пріоритету. Серверна частина додатку повинна бути досить потужною, якщо передбачається велика кількість клієнтів, оскільки вона повинна витримувати навантаження і якнайшвидше відповідати на кожен запит. Тож потрібно постійно слідкувати за навантаженням та при потребі збільшувати ресурс серверної складової.

Детально оглянувши клієнт-серверну архітектуру можна виділити її переваги та недоліки.

Таблиця 2.1

Переваги та недоліки клієнт-серверної архітектури

Переваги	Недоліки
Виконання основної частини роботи на більш потужній серверній частині, тим самим знімаючи навантаження з клієнта.	При виході сервера з ладу система перестає працювати
Зберігання даних відбувається на серверній стороні, що є більш надійним ніж зберігання на клієнті.	Компоненти для створення потужної серверної частини є досить дорогими
Кросплатформеність – клієнт може працювати з сервером незважаючи на операційну систему яку той використовує	

Для даного проекту архітектура клієнт-сервер є правильним вибором, оскільки веб-додаток не повинен сильно навантажувати систему клієнта, тому логіку додатку потрібно виносити на серверну частину. При цьому якщо буде відбуватись збільшення кількості клієнтів, можна буде також поступово збільшувати потужність сервера.

2.5. Функціональні вимоги до веб-додатку

В даному підрозділі буде сформульований перелік конкретних задач, які повинен виконувати веб-додаток, що розробляється в рамках магістерської дисертації. Він повинен забезпечувати аутентифікацію користувача та мати усі ігрові елементи в рамках онлайн-вікторини. Повний перелік функціональних вимог вказаний у табл. 2.2:

Таблиця 2.2

Функціональні вимоги до додатку

Функція	Реалізація
Аутентифікація	Надати можливість користувачу увійти до свого профілю використавши логін та пароль. Також надати можливість створити гостьовий профіль з випадково згенерованим нікнеймом.
Реєстрація	Надати можливість користувачу створити профіль, вказавши свої персональні дані, додати аватар тощо.
Редагування профілю	Надати можливість користувачу змінити задані при реєстрації персональні дані.
Перегляд правил гри	Створити сторінку, де докладно описані правила вікторини.
Конструктор питань	Надати можливість окремим користувачам додавати, редагувати та видаляти питання та категорії питань, тобто маніпулювати даними у сховищі. Для цього адміністратор має надати права на конкретні дії, що може виконувати той чи інший користувач. За замовчуванням користувач не має доступу до сховища питань.

Продовження таблиці 2.2

Функція	Реалізація
Конфігурація питання	Надати можливість користувачу з правами доступу до розділу питань додавати медіа елементи до окремого питання. Також питання має мати до 6-ти варіантів відповіді, і тільки один з яких повинен бути вірним.
Конфігурація категорії питань	Надати можливість користувачу з правами доступу до розділу категорій питань додавати чи редагувати окремі категорії, якщо виявиться, що питання не підходить під одну із вже існуючих категорій.
З’єднання гравців	Надати користувачу можливість створити або доєднатися до вже створеної гри.
Забезпечення безпосереднього процесу гри	Надати всі необхідні функції для забезпечення процесу гри згідно з правилами вікторини.
Вихід з профілю	Надати можливості користувачу вийти з профілю.

2.6. Технології використані у проекті

Як і зазначалось у підрозділі 2.1, розробка даного проекту буде виконуватись за допомогою програмної платформи Node.js та фреймворку Sails.js, які надають можливість імпортувати різноманітні необхідні для розробки модулі та бібліотеки, а також пришвидшують побудову самого

проекту. В якості бази даних, на стадії розробки буде використовуватися файлова база даних sails-disk, яка вбудована у фреймворк Sails.js. Клієнтська частина буде виконана мовою HTML [14] з використанням набору інструментів Bootstrap [15], з використанням вставок JavaScript-коду за допомогою шаблонізатора EJS. Взаємодія гравців у реальному часі буде відбуватись за допомогою методів sails.sockets, вбудованих у Sails.js. Також у розробці клієнтської частини буде використовуватись бібліотека jQuery для продуктивної роботи з елементами HTML-розмітки.

Написання програмного коду буде відбуватись у середовищі WebStorm від компанії JetBrains. Завдяки швидкому аналізу коду та зручному автодоповненню у даному середовищі розробки дуже легко шукати необхідні частини коду, виконувати рефакторинг, налагоджування та імпортувати сторонні бібліотеки. Також даний інструмент має інтеграцію з різними системами керування версіями.

Усі перелічені інструменти допоможуть максимально ефективно виконати розробку поставленої задачі.

ВИСНОВОК ДО ДРУГОГО РОЗДІЛУ

У даному розділі після детального огляду були обрані технології для розробки веб-додатку. Основними інструментами для розробки будуть:

- середовище розробки WebStorm;
- програмна платформа Node.js;
- фреймворк Sails.js;
- веб-сокети.

Додаток буде виконаний дотримуючись шаблонів MVC (Model View Controller) та архітектури клієнт-сервер.

Також були сформульовані функціональні вимоги до додатку та складена таблиця з описом кожного пункту. Головною вимогою була визначена необхідність наявності базових інструментів для відтворення механіки гри у повному обсязі, а саме:

- авторизація, створення та редагування профілю;
- зручний та функціональний конструктор питань;
- онлайн гра з обраними гравцями у режимі реального часу.

РОЗДІЛ 3

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1. Загальна структура проекту та правила гри.

Розробку даного проекту можна поділити на 3 основні частини:

1. Розробка інструментів для реєстрації та інших маніпуляцій з профілем користувача.
2. Розробка конструктора запитань.
3. Розробка безпосереднього процесу гри та взаємодії користувачів у ній.

Правила гри можна охарактеризувати наступним чином:

1. Гравці по черзі обирають запитання із заздалегідь сформованої сітки питань. Запитання мають вартість від 100 до 1000 балів.
2. Після вибору запитання, воно з’являється на екрані у всіх гравців. Також на екрані з’являється таймер та велика червона кнопка, натискаючи на яку гравець свідчить про свою готовність відповідати. Відповідатиме на питання тільки той гравець, який швидше за всіх натисне червону кнопку. Якщо ніхто не натиснув кнопку для відповіді, то по закінченні таймера поточне питання пропускається.
3. Якщо хтось вирішив відповідати та першим встиг натиснути червону кнопку, у нього на екрані з’являться варіанти для відповіді та таймер. Обравши правильний варіант, гравцю нараховуються бали за дане питання, після чого гра знову переходить до сітки запитань. Якщо гравець не встигає відповісти за вказаний час, відповідь вважається невірною.
4. Якщо обраний варіант відповіді виявився невірним, гравець втрачає бали у розмірі вартості поточного питання, а інші гравці матимуть можливість відповісти на нього натиснувши червону кнопку.
5. Виграє той, хто набере найбільшу кількість балів на протязі гри.

Подальша розробка проекту буде виконана таким чином, щоб забезпечити виконання вищевказаних правил гри.

3.2. Створення моделей додатку.

Даний проект для роботи використовує 3 основні моделі:

- модель користувача;
- модель запитання;
- модель категорії запитань.

Взаємозв'язок цих моделей описаний на рис. 3.1:

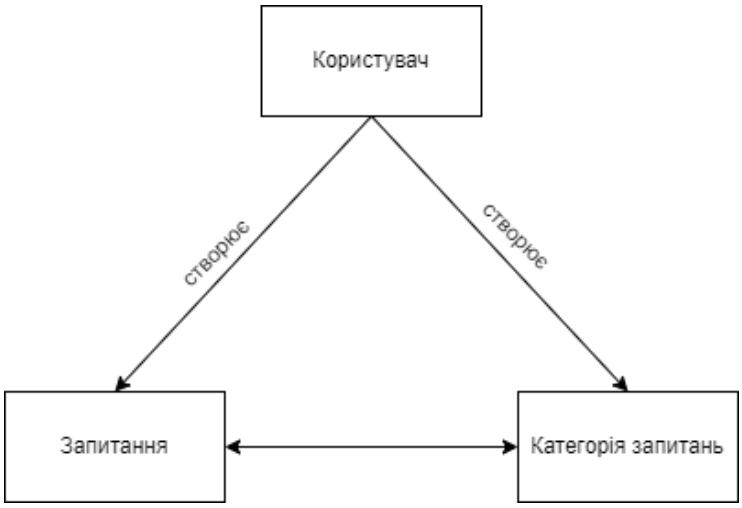

Рис. 3.1 – Схема взаємодії моделей додатку

Модель користувача містить персональні дані зареєстрованих користувачів. Модель запитань містить текст питання, варіанти відповідей, посилання на мультимедійні файли, а також ідентифікатор категорії питань, до якої відноситься. Модель категорії питань містить назву та всі ідентифікатори запитань, які підпадають під дану категорію. Зареєстровані користувачі з належними правами доступу можуть додавати або редагувати моделі запитання та категорії запитань.

Повний перелік полів даних моделей наведений у табл. 3.1 – 3.3.

Таблиця 3.1

Модель користувача (User.js)

Поле	Тип	Вимоги до валідації	Опис
id	number	unique: true required: true	Ідентифікатор користувача

Продовження таблиці 3.1

Поле	Тип	Вимоги до валідації	Опис
nickname	string	required: true	Нікнейм користувача
login	string	unique: true required: true	Логін користувача
password	string	required: true	Пароль
avatar	json		Аватар

Таблиця 3.2

Модель запитання (Question.js)

Поле	Тип	Вимоги до валідації	Опис
id	number	unique: true required: true	Ідентифікатор запитання
categories	collection: 'category',	required: true	Перелік категорій, до яких відноситься запитання
image	json		Зображення
video	string		Посилання на відео
text	string	required: true	Текст запитання
answerOptions	json	required: true	Варіанти відповідей
cost	number	required: true	Вартість запитання

Таблиця 3.3

Модель категорії запитань (Category.js)

Поле	Тип	Вимоги до валідації	Опис
id	number	unique: true required: true	Ідентифікатор категорії запитань
name	string	unique: true required: true	Назва категорії
question	collection: 'question'	required: true	Запитання, які відносяться до даної категорії

3.3. Розробка інструментів реєстрації та авторизації користувачів.

Авторизуватися у онлайн-вікторині «Сто запитань» можна двома способами (рис. 3.2):

- увійти за допомогою створеного раніше персонального профілю;
- увійти за допомогою гостьового профілю.

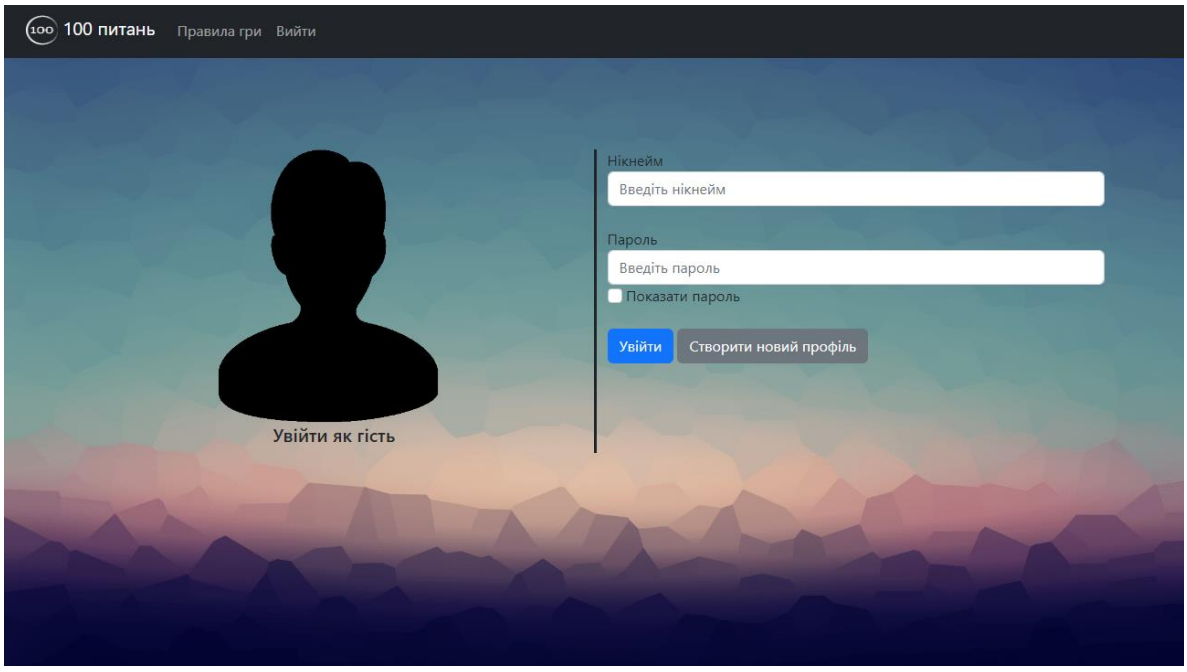

Рис. 3.2 – Сторінка авторизації

Почнемо з реалізації першого способу. Для цього створюємо форму авторизації, яка має поля «логін» та «пароль», а також прапорець, який відповідає за відображення на екрані символів, з яких складається пароль. При вводі своїх персональних даних та натисканні кнопки «Вхід» користувач відправляє свої персональні дані на сервер за допомогою ajax-запиту, а сервер, в свою чергу, виконує пошук профілю користувача у базі даних. Якщо профіль було знайдено, додаток перенаправляє користувача на сторінку головного меню програми. Якщо ні, користувач отримує на екран повідомлення про те, що профіль не було знайдено.

Якщо ж користувач досі не має власного профілю, він може натиснути кнопку «Створити новий профіль» та потрапити на сторінку реєстрації (рис. 3.3), яка складається з форми, що включає в себе наступні поля:

- нікнейм – ім’я користувача, під яким він буде відображатись іншим користувачам;

- унікальний логін користувача, який буде використовуватись для пошуку профілю;

- пароль користувача;

- аватар профілю користувача, який можна завантажити зі свого девайсу, обравши відповідне зображення.

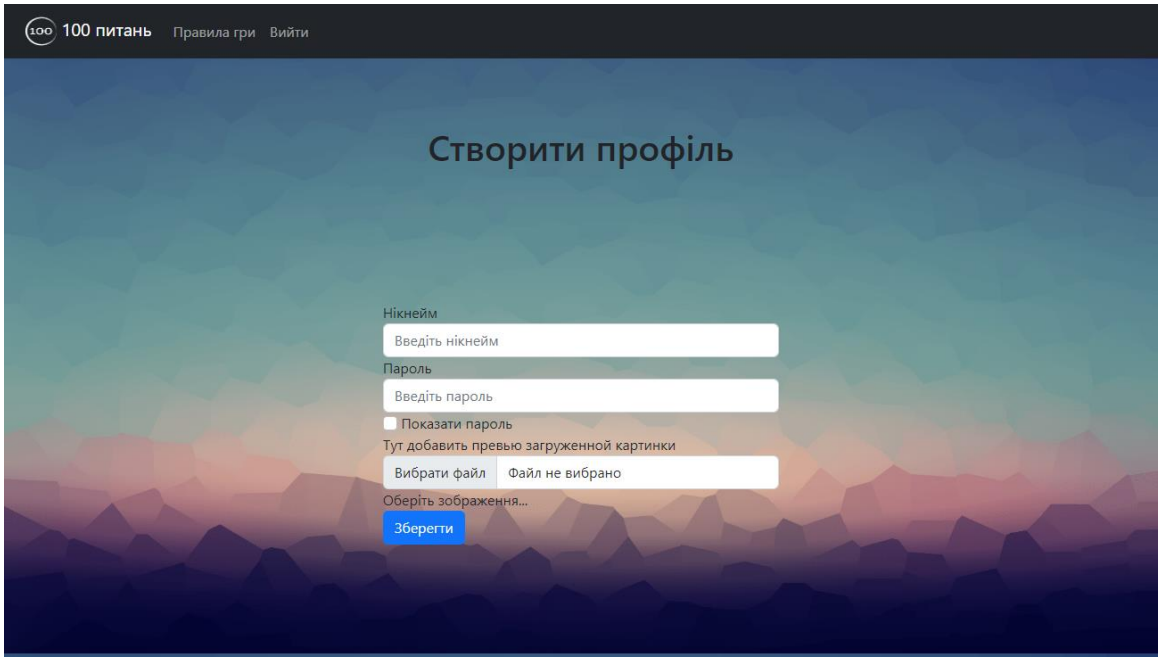

Рис. 3.3 – Сторінка реєстрації

Повний алгоритм проходження авторизації вказаний на рис. 3.4:

Рис. 3.4 – Алгоритм проходження реєстрації користувача

Після створення профілю користувач автоматично авторизується за допомогою введених персональних даних.

Другим способом авторизації є вхід за гостьовим профілем. Для цього на сторінці авторизації потрібно обрати саме цей варіант, після чого для користувача буде автоматично згенероване унікальне ім’я, під якими його

будуть бачити інші користувачі та додано стандартний аватар. Генерація нікнейму виконується за допомогою бібліотеки “nickname-generator”, яка при стандартних параметрах створює рядок з двох випадково згенерованих слів та двох цифр.

Повний алгоритм проходження авторизації показаний на рис. 3.5:

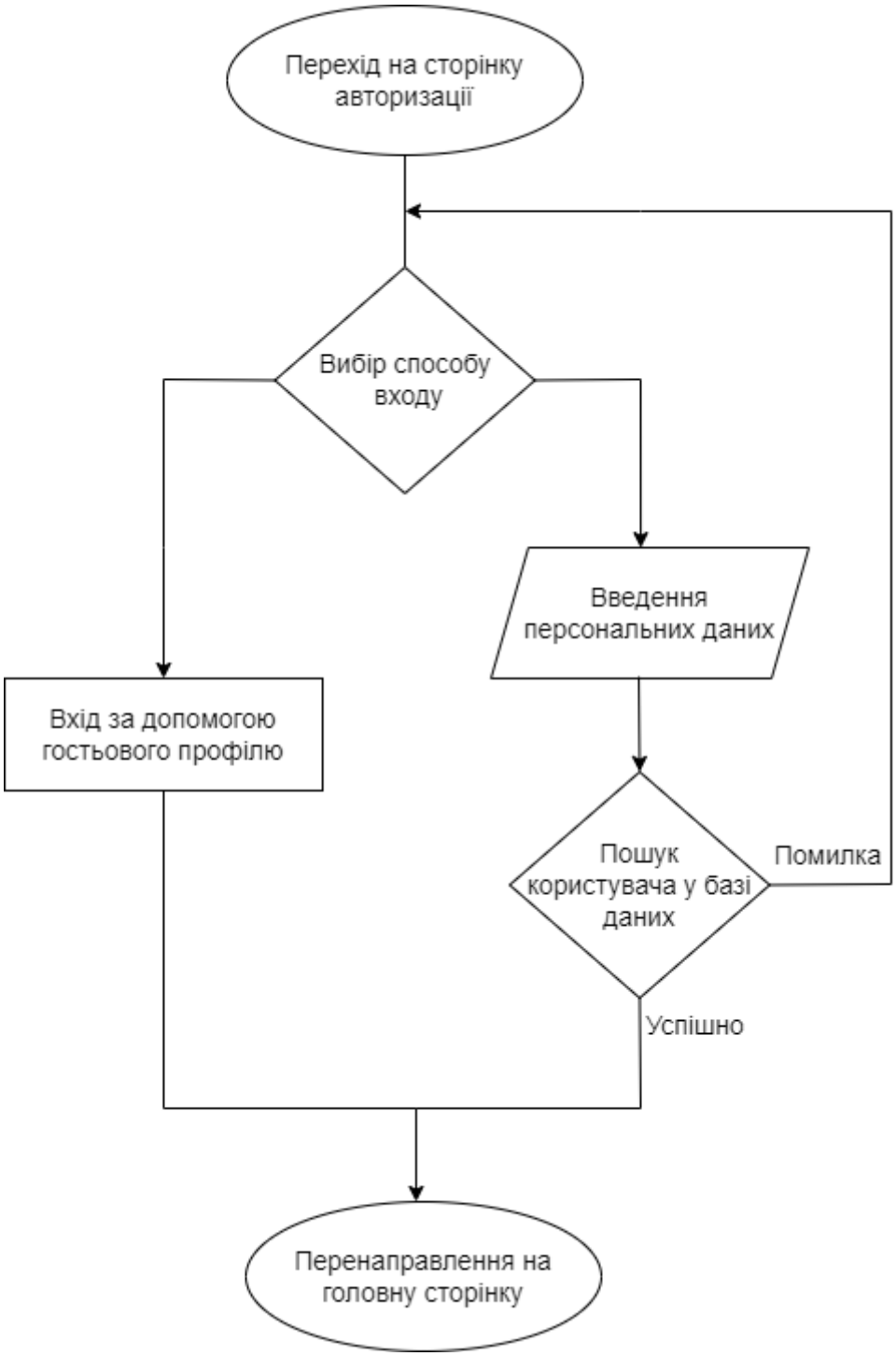

Рис. 3.5 – Алгоритм проходження авторизації користувача

Після успішної авторизації сервер записує дані користувача у об’єкт req.session.user для подальшої роботи з ними. Структура даного об’єкту описана у таблиці 3.4:

Таблиця 3.4

Структура об’єкту req.session.user

Властивість	Тип	Опис
type	string	Спосіб авторизації користувача. Може набувати значень “guest” та “authorized”
nickname	string	Ім’я користувача
avatar	json	Об’єкт, який містить шлях до персонального аватару, користувача

Авторизувавшись у системі користувач потрапляє до головного меню (рис. 3.6), яке складається з 5-ти розділів:

1. «Почати гру» – виконує перехід до лобі, з якого можна буде створити гру, або доєднатися до вже створеної.
2. «Конструктор запитань» – виконує перехід до конструктора запитань, в якому при наявності потрібних прав доступу можна додавати та редагувати окремі запитання або категорії запитань.
3. «Редагувати профіль» – виконує перехід на сторінку редагування профіля, на якій можна змінити персональні дані гравця.
4. «Правила гри» – виконує перехід на сторінку, на якій описані правила вікторини.
5. «Інформація про гру» – виконує перехід на сторінку, на якій вказана основна інформація про автора, саму вікторину, наявні посилання на пошту та телеграм для пропозицій.

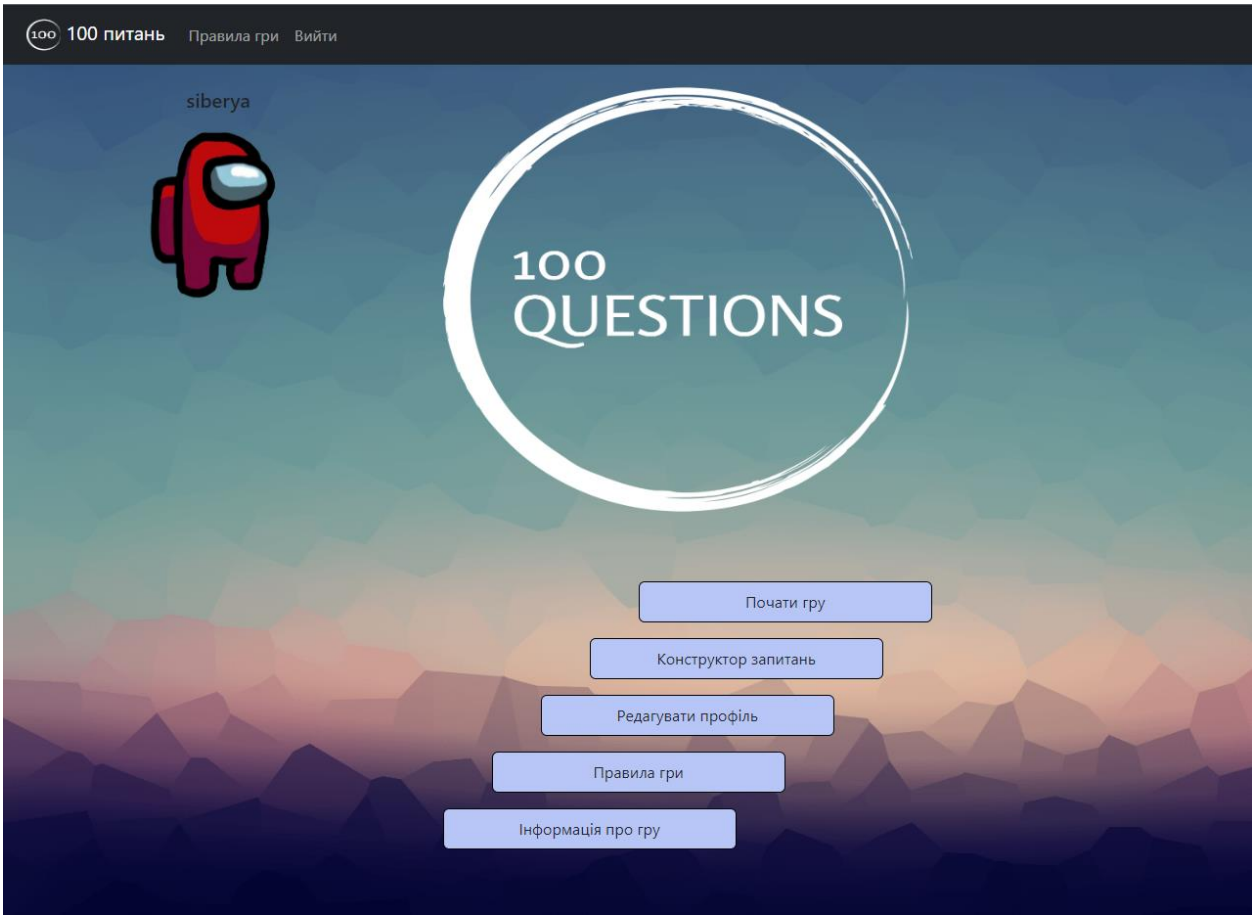

Рис. 3.6 – Головне меню вікторини

Почнемо розробку з трьох останніх пунктів, які можна охарактеризувати як інформаційні та службові розділи вікторини.

3.4. Інформаційні та службові розділи вікторини.

Розпочнемо зі сторінки редагування профілю (рис. 3.7). На цій сторінці можна змінити персональні дані користувача та завантажити новий аватар. Якщо перехід на цю сторінку виконаний після реєстрації користувача за допомогою гостьового профілю, то додаток запропонує користувачу створити власний профіль, після чого буде виконана авторизація у системі на основі нововведених даних.

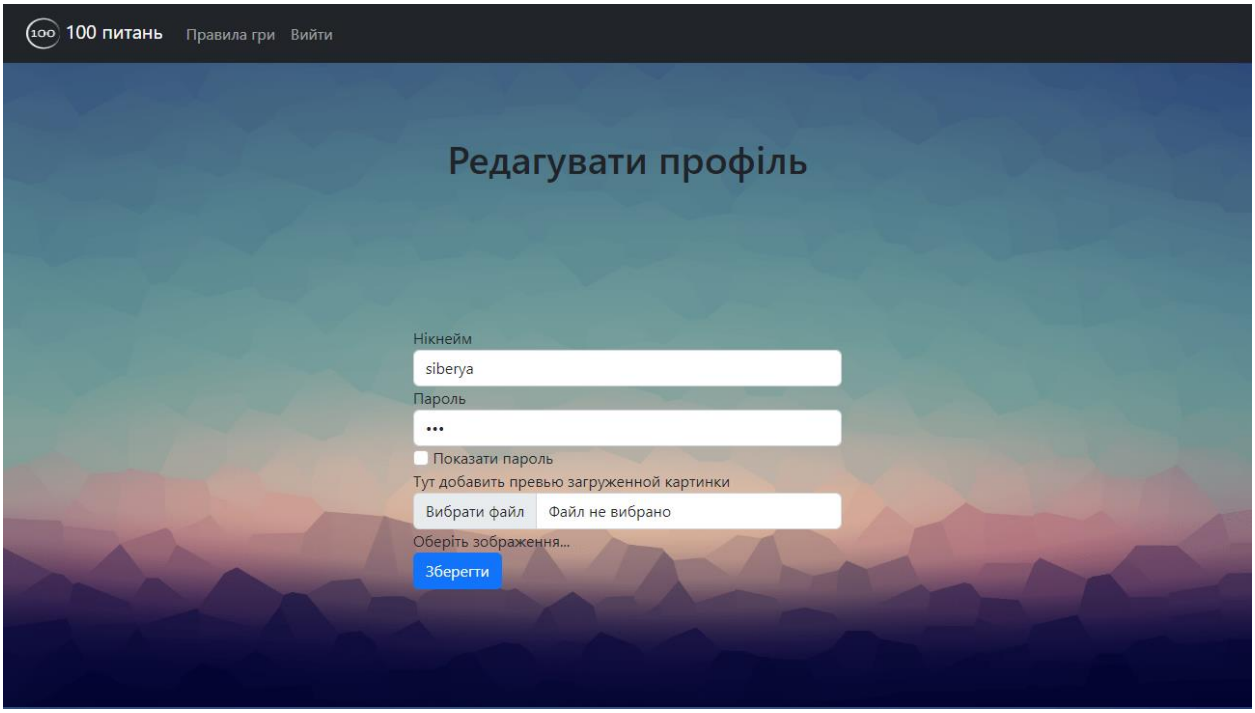

Рис. 3.7 – Сторінка редагування профілю

Наступним розділом є інформаційна сторінка, яка містить правила гри (рис. 3.8). Її можна також відкрити знаходячись на буд-якій сторінці за допомогою навігаційного меню у верхній частині екрану. У такому разі сторінка відкриється у новій вкладці аби не руйнувати процес гри.

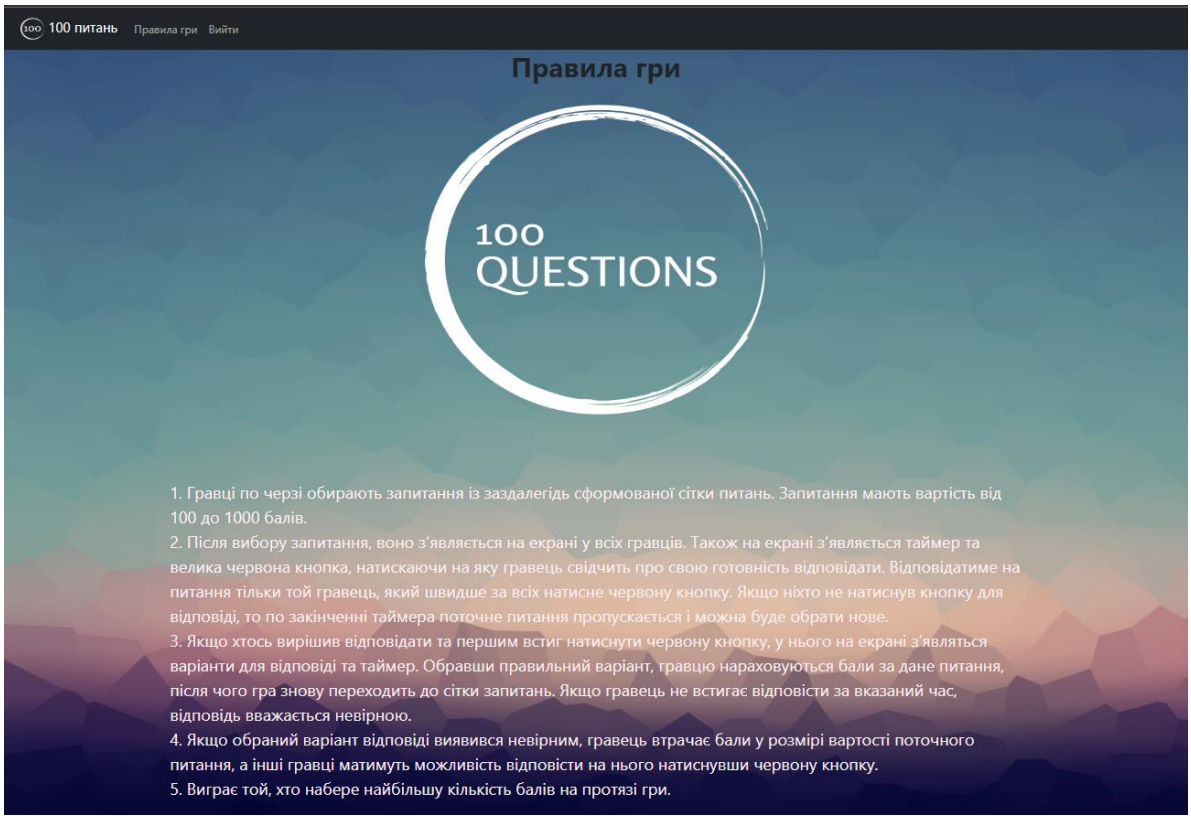

Рис. 3.8 – Правила гри

Останнім розділом є сторінка с інформацією про гру (рис. 3.9), яка містить:

- дані про поточну версію гри;

- інформацію про останні оновлення;

- посилання на пошту та інші ресурси, де можна залишити свої коментарі та побажання для автора;

- інформацію про автора та посилання за яким можна підтримати проект.

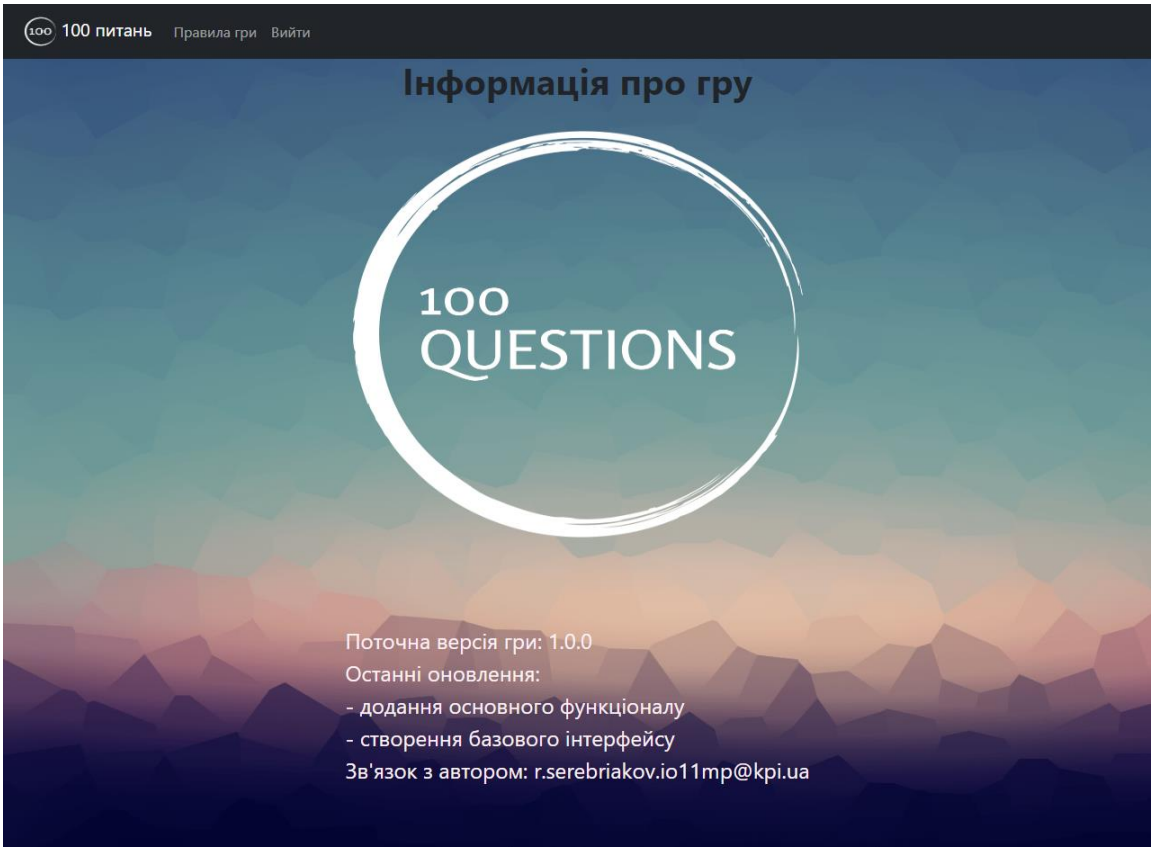

Рис. 3.9 – Інформація про гру

Закінчивши з службовими та інформаційними сторінками перейдемо до розробки конструктора запитань.

3.5. Конструктор запитань.

В даному проекті буде єдина база запитань, до якої привілейовані користувачі зможуть додавати окремі запитання, а також створювати нові категорії. Кожне запитання матиме обов’язкові параметри для вводу, серед яких є категорія, текст запитання, варіанти відповідей та вартість. На початку

гри по кожній з десяти обраних гравцем темах будуть сформовані десять запитань вартістю від 100 до 1000, які будуть обрані випадковим чином.

В якості конструктора запитань для вікторини у даному проекті буде використовуватись open-source проект sails-adminpanel [16]. Даний проект інсталюється у вигляді бібліотеки за допомогою менеджера npm. Використання sails-adminpanel у якості конструктора запитань обумовлене двома причинами:

1. Даний проект ідеально підходить для вікторини у якості конструктора запитань, оскільки він надає весь функціонал, необхідний для побудови конструктора запитань.
2. Я був залучений до розробки значної частини даного проекту, а знання можливостей та структури бібліотеки допоможуть якнайкраще налаштувати її для потреб вікторини.

Серед необхідного функціоналу, який знадобиться при побудові конструктора запитань є:

- робота з базою даних проекту, тобто додавання та редагування запитань та їх категорій;

- налаштування прав доступу для надання можливостей описаних у попередньому підрозділі для окремих користувачів;

- доступ до усіх існуючих даних, маніпулювання якими є дуже корисним на етапі розробки.

Для входу до адмінпанелі необхідно налаштувати модель “Users AP” та “Groups AP”, які використовуються бібліотекою для надання прав доступу до окремих розділів проекту. Для прикладу створимо користувача, який зможе додавати та редагувати запитання, але категорії запитань він зможе тільки передивлятися. Тож спершу додамо користувача до системи у розділі “Users AP”. Для цього задаємо ім’я, логін та пароль, також можемо вказати дату, коли даний профіль стане недійсним (рис. 3.10).

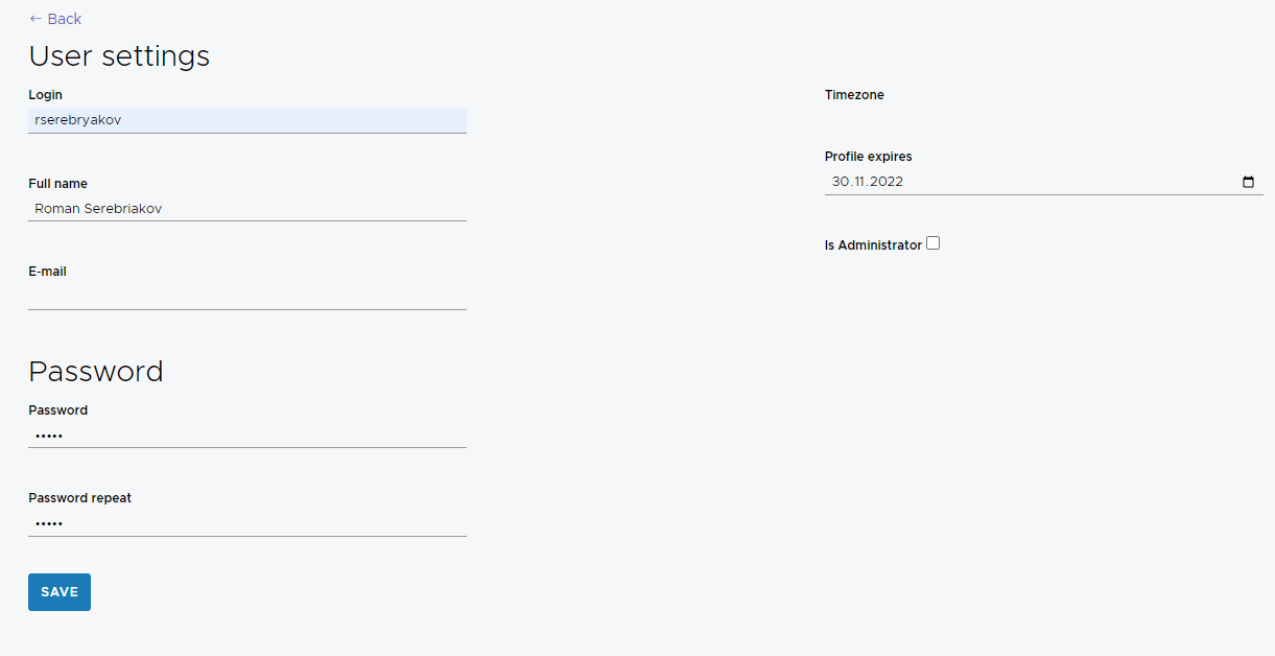

Рис. 3.10 – Налаштування профілю користувача для доступу до конструктора запитань

Після створення профілю користувача потрібно створити групу, яка матиме права доступу до потрібних розділів та додати до неї цього користувача. Для цього перейдемо до розділу “Groups AP” та заповнимо необхідні поля (рис.3.11).

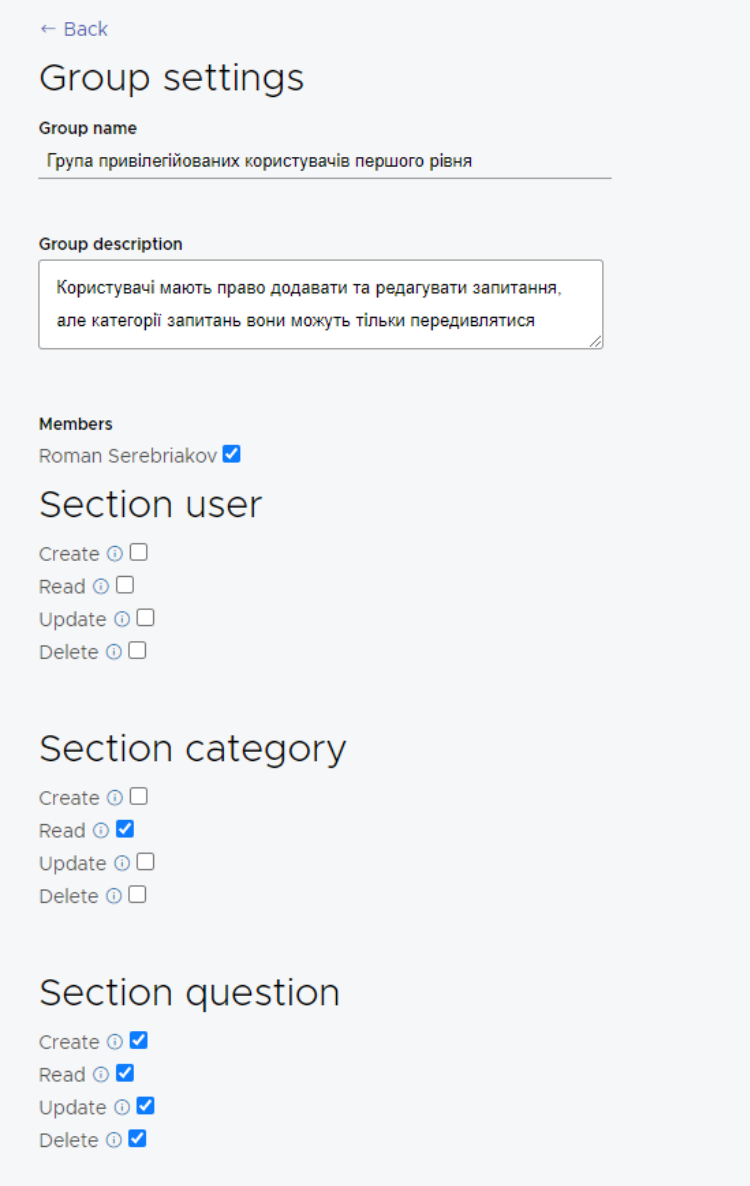

Рис. 3.11 – Налаштування групи користувачів для доступу до окремих розділів проекту

Далі вмикаємо авторизацію у проекті та виконуємо вхід за допомогою створеного профілю. Увійшовши до системи можна побачити, що користувач має повний доступ до моделі запитань (рис. 3.12) та лише доступ на перегляд для моделі категорії запитань (рис. 3.13).

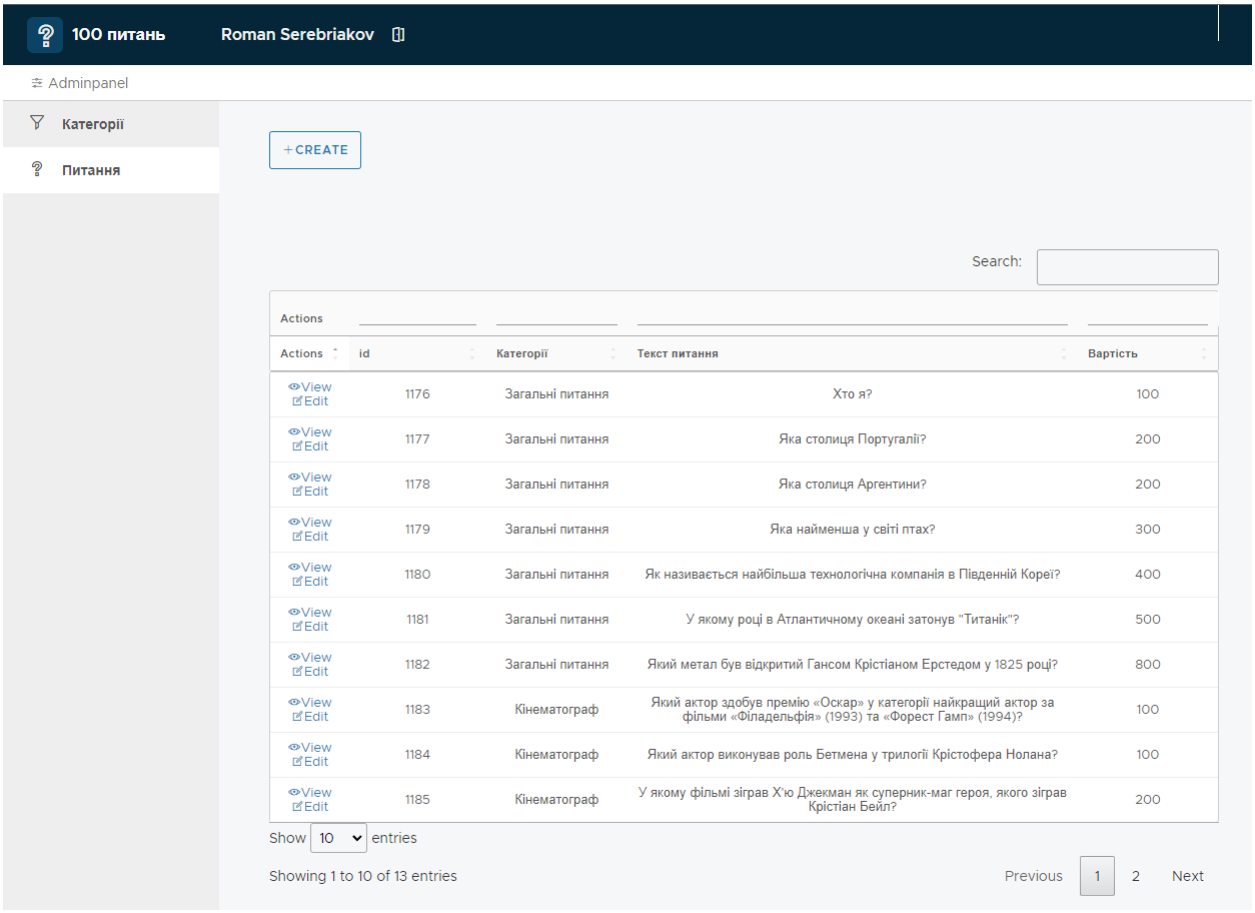

Рис. 3.12 – Повний доступ до моделі запитань

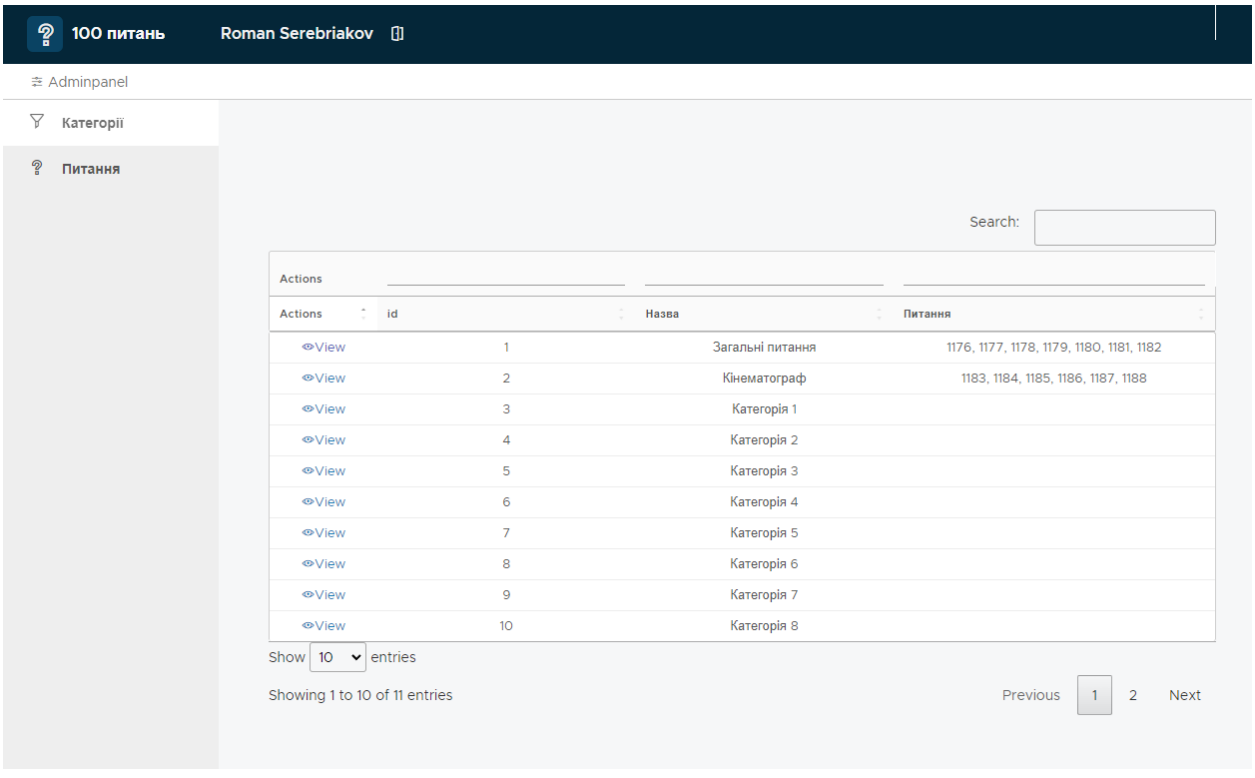

Рис. 3.13 – Доступ на перегляд моделі категорії запитань

Після створення профілю користувача та налаштування прав доступу перейдемо до конфігурації адмінпанелі, яка дозволить легко і швидко створювати запитання. Налаштування адмінпанелі відбувається у файлі adminpanel.js, який знаходиться у папці з іншими конфігураційними файлами проекту. Там можна задати схему роботи додатку, вказати назви розділів та полів, а що найголовніше – обрати віджети для роботи з цими полями. Для створення запитань нам знадобляться наступні віджети:

- select-pure – віджет для створення асоціацій між двома моделями у вигляді випадаючого списку з одинарним або множинним вибором;

- fileuploader – віджет для додавання та попередньої обробки зображень в залежності від обраних параметрів;

- handsontable – віджет для створення таблиць для роботи з різними типами даних.

Як приклад, створимо запитання для категорії «Кінематограф» (рис. 3.14). Для початку з допомогою віджету “select-pure” оберемо категорію «Кінематограф» з моделі категорій запитань. Питання буде текстове, тож поля для додавання зображень та відео пропускаємо. Далі додаємо текст запитання, створюємо варіанти відповідей за допомогою віджета “handsontable” та помічаємо галочкою правильну відповідь. Наприкінці, обираємо вартість запитання, яка варіюється у межах від 100 до 1000 балів. Після збереження дане питання буде використовуватись у грі. Створивши таким чином значну кількість запитань до кожної категорії гравці зможуть вільно використовувати її при формуванні сітки запитань.

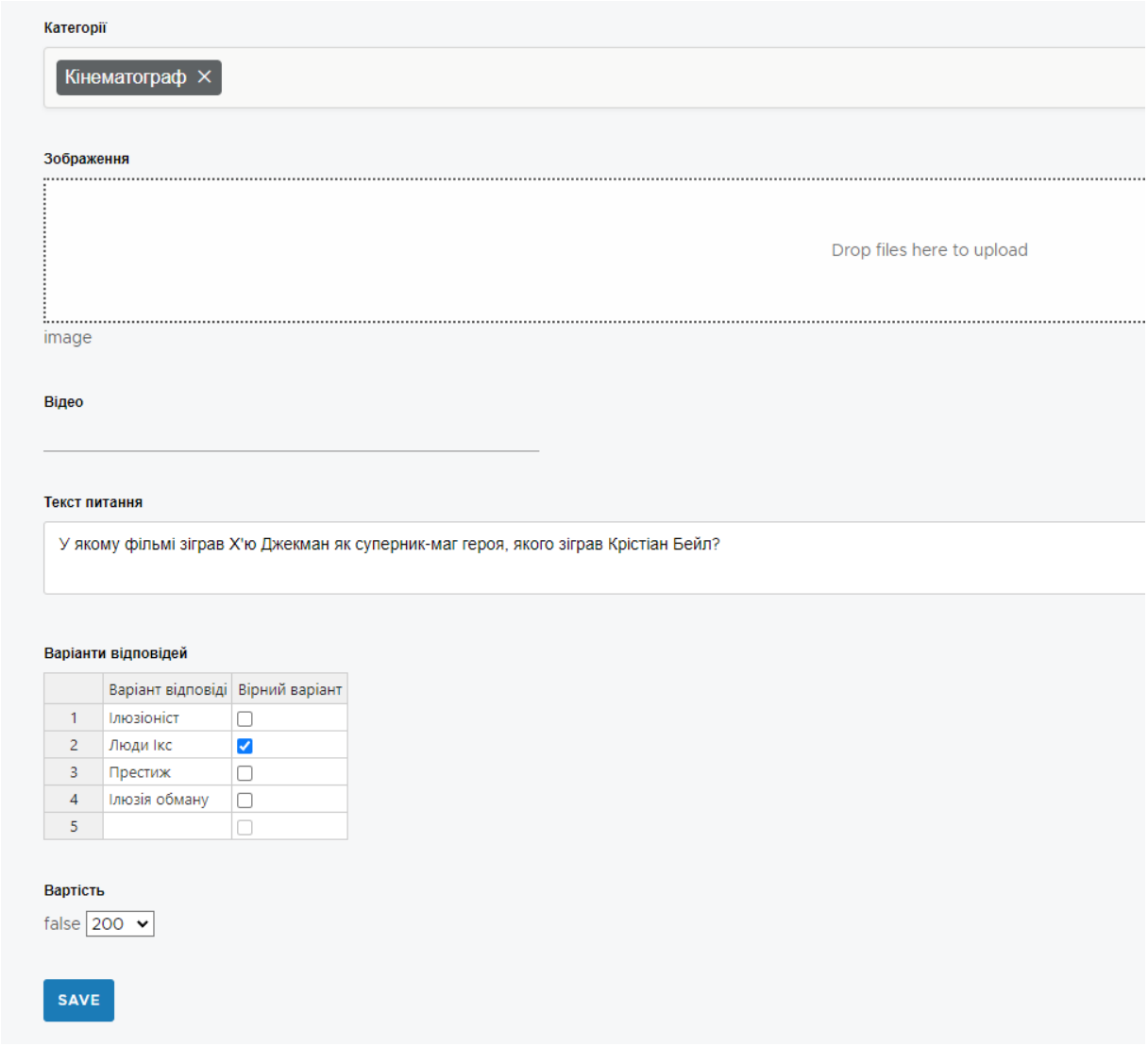

Рис. 3.14 – Створення запитання у категорії «Кінематограф»

3.6. Розробка процесу гри та засобів взаємодії гравців.

Розробка буде ґрунтуватися на використанні сокетів як головного способу взаємодії між клієнтами [17], тобто гравцями, а сам процес гри починається зі створення так званої «кімнати» для гравців. Кімната у даному випадку є об’єктом, до якого підв’язані ідентифікатори сокетів гравців. За допомогою назви кімнати та функції “broadcast()” сервер може відправляти повідомлення гравцям, які прив’язані до конкретної кімнати.

При виборі розділу «Почати гру» у головному меню вікторини гравець потрапляє до лобі (рис. 3.15). В цей момент сервер за допомогою функції “join()” додає гравця до кімнати під назвою “lobby”. Також сервер створює об’єкт sails.rooms, який буде містити інформацію про існуючі кімнати,

доєднаних до них гравців, а також внутрішню ігрову інформацію. Структура об’єкту sails.rooms наведена на рис. 3.16. Кожна кімната має свої дані про гравців та стан гри, окрім lobby, яка є службовою кімнатою та містить тільки ідентифікатори сокетів користувачів. Гравці, які знаходяться в лобі, можуть доєднуватися до інших кімнат та розпочинати гру. Інформацію, яка міститься в інших кімнатах на рисунку вказано як roomInformation. Структура цього об’єкту докладно описана у табл. 3.5.

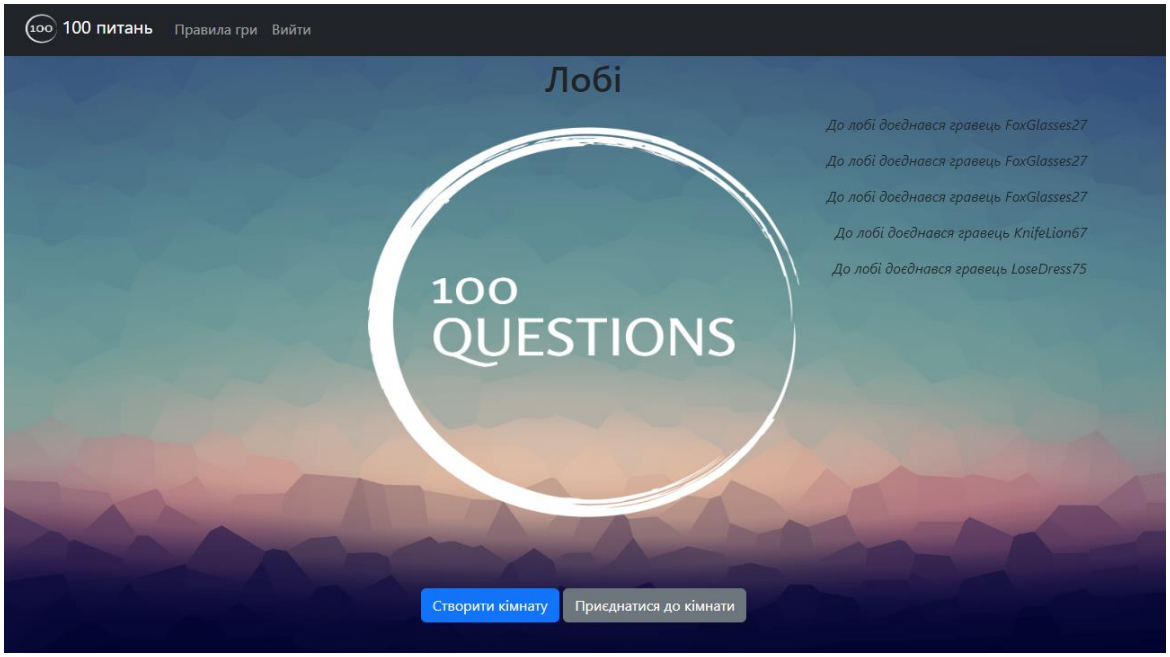

Рис. 3.15 – Лобі

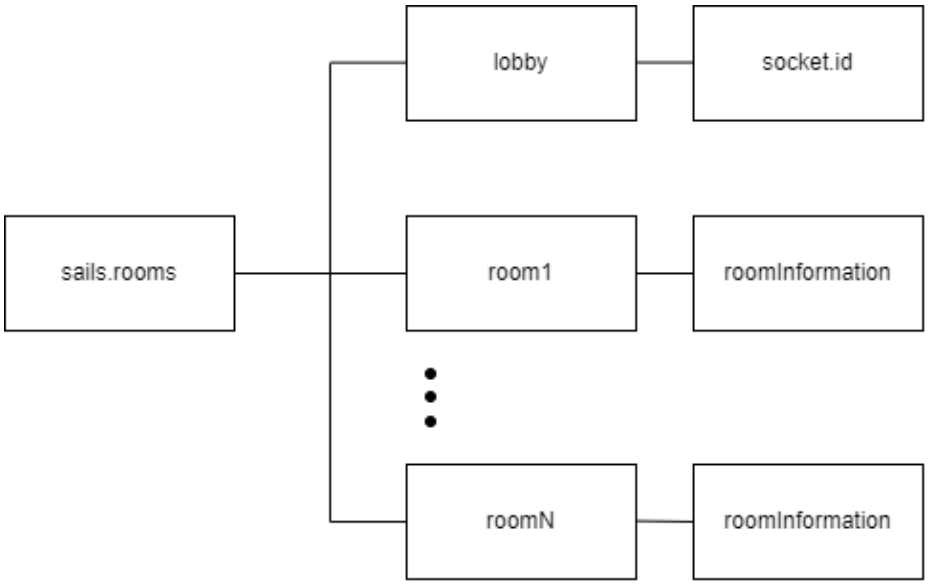

Рис. 3.16 – Структура об’єкту sails.rooms

Таблиця 3.5

Структура об’єкту roomInformation

Властивість	Тип	Опис
Нікнейм користувача (поле зі змінною назвою)	object	Об’єкт, який містить повну інформацію про користувача, яка може знадобитись системі під час гри. Створюється при доєднанні користувача до кімнати.
questionsPack	object	Об’єкт, який містить набір запитань для гри. Створюється після того, як користувач сформує сітку запитань.
currentPlayer	string	Поточний гравець
questionTimer	function	Таймер на вибір запитання
answerTimer	function	Таймер на відповідь
answerOptionsTimer	function	Таймер на вибір правильної відповіді
pointsList	object	Об’єкт, який зберігає набрані гравцями бали
playersWhoAlreadyAnswered	array	Список гравців, які вже відповіли на поточне запитання

Записавши до об’єкта sails.rooms інформацію про те, що користувач наразі перебуває у кімнаті “lobby”, сервер за допомогою функції “broadcast()”

надсилає іншим гравцям, що перебувають у цій кімнаті, інформацію про під’єднання нового гравця.

Зайшовши до лобі, гравець може створити власну кімнату або ж приєднатися до вже створеної кімнати. Обравши варіант «Приєднатися до кімнати» гравцю потрібно буде ввести назву кімнати у модальному вікні, після чого той хто створив дану кімнату повинен буде прийняти заявку на приєднання. А обравши перший варіант гравець потрапляє до новоствореної кімнати, де він матиме можливість запросити до кімнати інших гравців, а також сформувати сітку запитань з існуючих в базі даних категорій. Внаслідок цієї дії сервер видаляє даного гравця з кімнати “lobby” та додає його до кімнати, названої його ім’ям, а також надсилає йому список усіх категорій запитань для формування ігрової сітки. Після цього клієнтська частина змінює інтерфейс лобі на інтерфейс кімнати (рис. 3.17).

Рис. 3.17 – Інтерфейс кімнати

Для того щоб запросити інших гравців до створеної кімнати, потрібно ввести їх унікальні нікнейми у полі для запрошень у лівій частині екрану. Додати можна лише того користувача, який знаходиться у лобі. Тому сервер спочатку перевірить чи виконується ця умова. Якщо умова виконана, сервер надсилає знайденому гравцю запит на приєднання до кімнати використовуючи функцію “broadcast()” та ідентифікатор сокету, який можна дістати з об’єкту sails.rooms.lobby за ім’ям користувача. Після цього

відправник отримує повідомлення про те, що запит був надісланий, а одержувачу надходить запит на приєднання до кімнати. Якщо він погоджується, сервер переміщує його в потрібну кімнату та сповіщає усіх гравців у цій кімнаті про додавання нового гравця. Відбувається це за допомогою події “updateRoomMembers”, внаслідок чого інтерфейс кімнати поповнюється ім’ям та аватаром нового гравця. Доданому гравцю клієнтська частина змінює інтерфейс та відображає поточний список гравців. Він також має змогу запрошувати інших гравців до кімнати, але у нього відсутня можливість вносити зміни до сітки запитань або ж розпочинати гру.

Гравцю, який створював кімнату окрім того щоб запросити друзів потрібно також сформувати сітку запитань. Вона складається з 10 категорій по 10 запитань у кожній. Разом 100 запитань. Обираючи категорію можна скористатись пошуком (рис. 3.18). Якщо категорія не була обрана гравцем, то вона буде випадковим чином обрана сервером. Зміна категорії буде відображатися усім гравцям у кімнаті за допомогою події “changeCategory”.

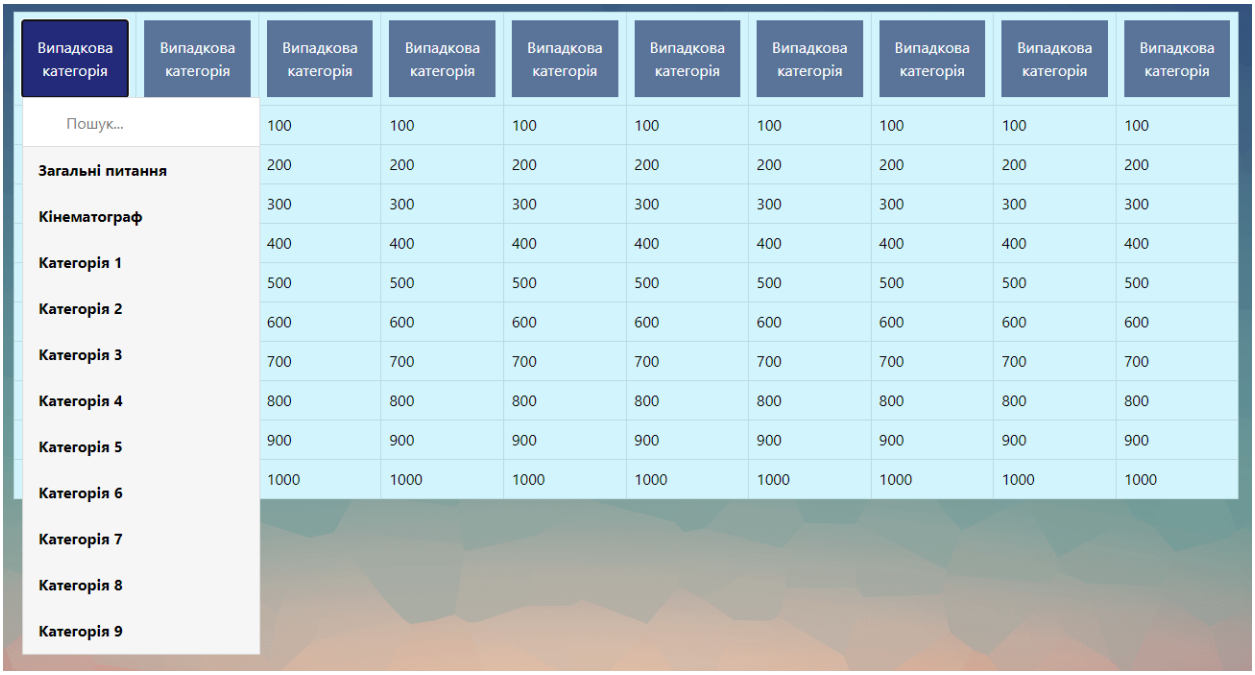

Рис. 3.18 – Використання пошуку при виборі категорії запитань

Обравши категорії запитань, які будуть використовуватися у грі, гравцю потрібно натиснути кнопку «Почати гру», після чого сервер почне формувати ігрову сітку запитань. Робить він це шляхом вибору випадкового запитання

кожної вартості для кожної категорії. Діставши з бази даних усі обрані категорії, він проходить по запитанням кожного номіналу та випадковим чином обирає з них одне, після чого переходить до наступної категорії. Сформувавши сітку запитань, сервер записує її до сховища sails.rooms, а її копію, яка складається лише з назв категорій та ідентифікаторів запитань відправляє гравцям за допомогою події “start”. Клієнтська частина в цей час змінює інтерфейс кімнати на інтерфейс гри (рис. 3.19) усім гравцям та заповнює сітку запитань даними з серверної частини. Починається гра.

Як і було зазначено у правилах, гравці ходять по черзі обираючи запитання з сітки. Для коректної обробки дій гравців до об’єкту кімнати додаються наступні поля:

- currentPlayer – гравець, який зараз виконує хід, тобто обирає запитання для гри;

- questionTimer – таймер на вибір запитання для гравця currentPlayer;

- answerTimer – таймер для гравців після вибору запитання, по закінченню якого питання буде вважатись зіграним, якщо ніхто не натисне кнопку «Відповісти»;

- answerOptionsTimer – таймер для вибору правильної відповіді на запитання, по закінченню якого буде вважатись, що гравець надав неправильну відповідь;

- pointsList – список набраних балів гравців;

- playersWhoAlreadyAnswered – список гравців, які вже відповідали на поточне запитання.

Для вибору запитання гравцю надається певний час, по закінченню якого питання обирається випадковим чином автоматично. Після вибору запитання, замість інтерфейсу сітки запитань усім гравцям буде відображатись текст запитання. При наявності, до нього також додаватиметься зображення або відео. Також буде запущено таймер на відповідь, по закінченню якого, якщо ніхто з гравців не захоче на нього відповідати, питання буде вважатись зіграним. Після натискання кнопки

«Відповісти» сервер відправить гравцю варіанти відповіді, серед яких гравцю потрібно обрати вірний. Іншим гравцям у цей час буде відображатись, що певний гравець вирішив відповідати, а також час, за який він повинен надати відповідь. Після вибору варіанту відповіді, сервер перевірить його на правильність, після чого або нарахує гравцю відповідну до запитання кількість балів та збереже їх у об’єкт “pointsList”, або ж вирахує таку ж кількість балів з рахунку гравця, додасть його до списку “playersWhoAlreadyAnswered” та відновить таймер на відповідь, після чого інші гравці зможуть спробувати дати відповідь на це запитання. В цей же час відбудеться оновлення набраних балів гравців, які будуть відображатись під ім’ям кожного гравця. Інтерфейс ігрового процесу вказаний на рис. 3.19.

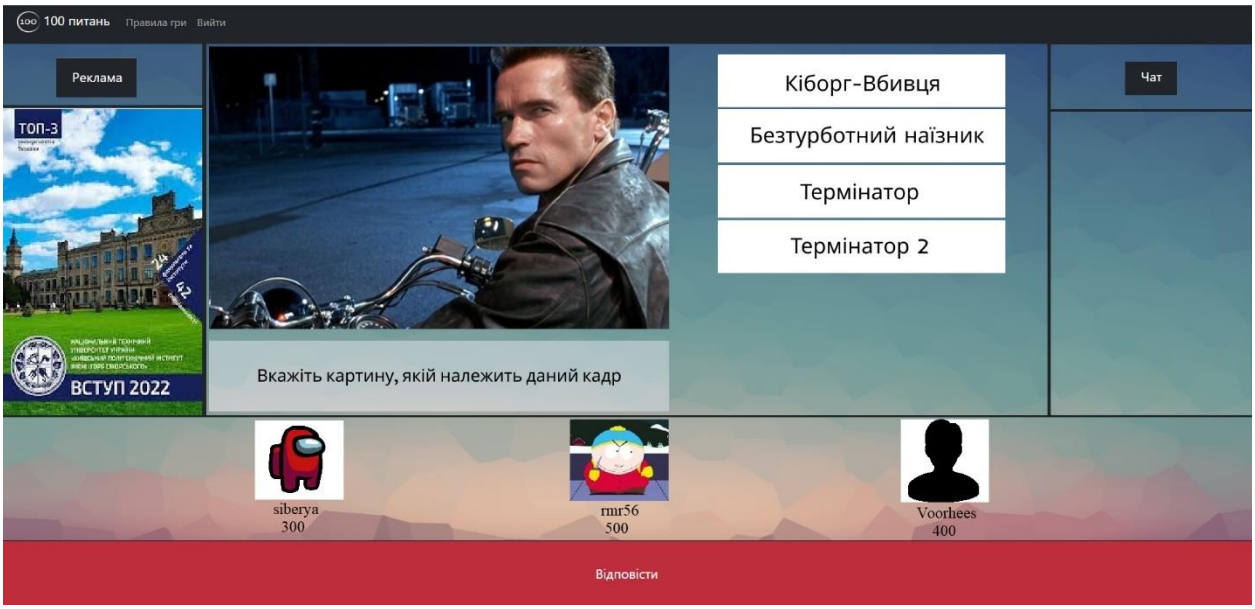

Рис. 3.19 – Інтерфейс ігрового процесу

Може трапитись ситуація, коли майже одночасно двоє або більше гравців натиснуть кнопку «Відповісти» і контролер не встигне опрацювати коректну відповідь. Для вирішення такої ситуації використаємо прапорець, який буде вказувати на те, що гравець вже відповідає на питання, а всі інші повинні чекати на його відповідь, такий собі м’ютекс. М’ютекси використовуються у паралельному програмуванні та є двійковими семафорами, тобто універсальними механізмами для організації взаємодії процесів. Коли перший гравець потраплятиме у контролер, м’ютекс буде

набувати стану “true” і в момент часу, коли сервер ще не передав інформацію іншим гравцям про те, що певний гравець вже відповідає на питання, цей механізм блокуватиме можливі наступні запити, що можуть надійти на контролер. Таким чином взаємодія гравців буде відбуватися належним чином і конфлікт одночасного натискання кнопки відповіді буде усунено.

Після того як питання буде пройдене, воно видалиться із сітки запитань. Гра закінчиться, коли усі запитання будуть пройдені, після чого на екран буде виведено ім’я переможця. Переможець визначається по кількості набраних балів на протязі гри. По закінченню гри гравці будуть мати можливість розпочати нову гру у тому ж складі та опиняться на стадії вибору категорій для сітки запитань.

ВИСНОВОК ДО ТРЕТЬОГО РОЗДІЛУ

У даному розділі були описані всі етапи розробки системи, від формування структури проекту та правил гри до розробки конкретних частин програми, а саме:

- сторінок реєстрації та авторизації;

- інформаційних та службових сторінок додатку, таких як правила гри, редагування профілю та інформації про гру;

- конструктора запитань;

- безпосередньо гри з усіма елементами взаємодії гравців необхідних для дотримання правил гри.

Також було створено три моделі даних:

- модель користувача;

- модель запитання;

- модель категорії запитань.

Розробка була виконана мовою Javascript з використанням фреймворку Sails.js.

РОЗДІЛ 4

РОЗРОБКА СТАРТАП ПРОЕКТУ

4.1. Опис ідеї проекту

Таблиця 4.1

Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Створити безкоштовну онлайн-вікторину «Сто запитань», частково засновану на телевізійній грі “Jeopardy”	Може використовуватись будь-якими користувачами, які мають доступ до інтернету та браузер	Можливість проводити ігрові сесії відповідаючи на різноманітні питання
		Можливість додавати та редагувати запитання, які будуть використані у грі

Для реалізації ідеї проекту було обрано наступні технології: мова програмування JavaScript, фреймворк Sails.js, середовище розробки WebStorm.

4.2. Технологічний аудит проекту

Таблиця 4.2

Технологічна здійсненність ідеї проекту

№	Ідея проекту	Технології її реалізації	Наявність технології	Доступність технологій
1	Онлайн-вікторина у вигляді веб додатку	Мова програмування JavaScript	Є в наявності	Доступна безкоштовно

Продовження таблиці 4.2

№	Ідея проекту	Технології її реалізації	Наявність технології	Доступність технологій
2		Фреймворк Sails.js	Є в наявності	Доступний безкоштовно
3		Середовище розробки WebStorm	Є в наявності	Доступне безкоштовно для навчання, коштує грошей для використання у комерційних цілях

4.3. Аналіз ринкових можливостей запуску стартап-проекту

Визначення ринкових можливостей, які можна використати під час ринкового впровадження проекту, та ринкових загроз, які можуть перешкодити реалізації проекту, дозволяє спланувати напрями розвитку проекту із урахуванням стану ринкового середовища, потреб потенційних клієнтів та пропозицій проектів-конкурентів. В таблиці 4.3 наведена попередня характеристика потенційного ринку для розроблюваного стартап-проекту.

Таблиця 4.3

Попередня характеристика потенційного ринку стартап-проекту

№	Показники стану ринку	Характеристика
1	Кількість головних гравців, од	2
2	Динаміка ринку (якісна оцінка)	Ринок завжди потребує цікавих проектів, які зможуть зацікавити користувача

Продовження таблиці 4.3

№	Показники стану ринку	Характеристика
3	Наявність обмежень для входу (вказати характер обмежень)	Конкуренція серед вже існуючих продуктів
4	Специфічні вимоги до стандартизації та сертифікації	Специфічних вимог до стандартизації та сертифікації немає

Таблиця 4.4

Характеристика потенційних клієнтів стартап-проекту

№	Потреба, що формує ринок	Цільова аудиторія	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Проведення ігрової сесії у онлайн-вікторині	Власники ПК або інших пристроїв, здатних виходити до мережі Інтернет за допомогою браузеру	Швидкість доступу до мережі Інтернет	Надання безкоштовного додатку для проведення ігрових сесій у режимі онлайн з іншими гравцями
2	Створення та редагування бази запитань для гри	Користувачі, які зможуть урізноманітнити ігровий процес за допомогою збільшення бази запитань	Різні кола інтересів користувачів	Зручний та функціональний конструктор запитань

Таблиця 4.5

Фактори загроз

№	Фактор	Зміст загрози	Можлива реакція компанії
1	Блекаут	Аварійні або планові відключення світла по всій території країни	Додання можливості збереження стану гри з можливістю завантажити гру з цього моменту.
2	Конкуренція	Інформація про додаток може не дійти до користувачів через домінування вже існуючих додатків	Впровадження рекламних кампаній на різноманітних порталах з метою зацікавлення нових користувачів
3	Нестача ресурсів на підтримку продукту	Задля успішного функціонування продукт потрібно підтримувати та розвивати, для чого необхідні як матеріальні, так і людські ресурси	Пошук можливих спонсорів та спеціалістів, які допоможуть проекту триматися на плаву

Таблиця 4.6

Фактори можливостей

№	Фактор	Зміст можливості	Можлива реакція компанії
1	Залучення інвестицій	Отримання коштів від спонсорів, а також розміщення рекламних оголошень всередині додатку	Отримання додаткових коштів на розвиток проекту та залучення декількох працівників для його підтримки

Продовження таблиці 4.6

№	Фактор	Зміст можливості	Можлива реакція компанії
2	Розвиток функціоналу продукту	Додання різних режимів гри, налаштувань ігрової сесії тощо	Дозволить урізноманітнити ігровий процес та підтримувати інтерес до продукту у гравців
3	Розповсюдження продукту на інші платформи	Створення десктопних версій продукту для операційних систем Linux та Windows, а також мобільних версій для Android та IOS	Розширення списку підтримуваних платформ дозволить залучити більшу кількість користувачів та збільшити популярність продукту

Таблиця 4.7

Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкуренто-спроможною)
Складність запуску додатку	Привернути увагу користувачів попри існуючих конкурентів	Впровадження рекламних кампаній на різноманітних порталах, що допоможуть зацікавити користувачів
Популярність існуючих аналогів	Перехопити аудиторію користувачів, які користуються вже існуючими додатками	Створення цікавих бонусів для нових користувачів та персоналізований підхід до кожної аудиторії користувачів

Таблиця 4.8

Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти в галузі	Постачальники	Клієнти	Товари-замінники
	Jeopardy та SIGame	Невідомо	Jeopardy Labs та Vladimir Khil	Користувачі, що мають браузер та доступ до мережі	Інші онлайн-вікторини, засновані на різних ідеях та правилах
Висновки:	Наявна велика кількість різноманітних рішень у конкурентів	Невідомо, які ще компанії збираються створити конкурентний продукт	Постачальники підтримують вже створені проекти та додають новий функціонал, чим залучають все більше користувачів	Клієнтом може стати будь-який користувач у мережі, якого цікавлять онлайн-вікторини	Існує велика кількість товарів-замінників, які мають схожий функціонал та дозволяють користувачам його використовувати

Таблиця 4.9

Обґрунтування факторів конкурентоспроможності

№	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Варіативний конструктор запитань	При створенні запитання до нього окрім тексту можна додати зображення та відео, що безперечно покращує змістовність самого запитання. Конструктор також є вбудованим у додаток, тож його не потрібно встановлювати окремо, як це може бути в додатках конкурентів
2	Автоматичне адміністрування гри	Завдяки заздалегідь доданим варіантам відповідей до кожного запитання, гра не потребуватиме ведучого, тому всі користувачі зможуть бути залучені безпосередньо до гри. Перевірку запитань та нарахування балів виконуватиме серверна частина додатку
3	Єдина база запитань	Набори запитань не потрібно шукати на сторонніх ресурсах, вони зберігаються в єдиній базі додатку і доступні усім користувачам. При заповненні сітки гри, користувачам допоможе зручний пошук по назвам категорій запитань, завдяки якому можна дуже швидко створити набір запитань для ігрової сесії

Таблиця 4.10

Порівняльний аналіз сильних та слабких сторін

№	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у по- рівнянні з розроблюваним продуктом						
			-3	-2	-1	0	1	2	3
1	Варіативний конструктор запитань	16					+		
2	Автоматичне адміністрування гри	18	+						
3	Єдина база запитань	15						+	
4	Взаємодія гравців у режимі онлайн	10				+			

Таблиця 4.11

SWOT-аналіз стартап-проекту

Сильні сторони: - варіативний конструктор запитань - автоматичне адміністрування гри - єдина база запитань - взаємодія гравців у реальному часі	Слабкі сторони: - єдина база буде мати дефіцит запитань на початку роботи проекту - немає налаштування ігрових режимів
Можливості: - додавання нового функціоналу до продукту - портування додатку на нові платформи	Загрози: - відключення світла у країні - конкуренція - нестача ресурсів для підтримки проекту

Таблиця 4.12

Альтернативи ринкового впровадження стартап-проекту

№	Альтернатива ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Рекламні інтеграції на різних порталах	Реклама на популярних ресурсах високоймовірно допоможе зацікавити інвесторів та завоювати популярність серед користувачів	В найкоротший час
2	Розповсюдження проекту серед студентського ком’юніті	Поширення інформації про додаток серед студентів з метою зацікавлення нової аудиторії дуже позитивно вплине на збільшення користувачів додатку	В найкоротший час
3	Укладання договору з різними компаніями	Розміщення рекламних інтеграцій всередині проекту позитивно позначиться на матеріальному забезпеченні проекту	Орієнтовно від декількох місяців, сильно залежить від популярності проекту

4.4. Розроблення ринкової стратегії проекту

Таблиця 4.13

Вибір цільових груп потенційних споживачів

№	Опис профілю цільової групи потенційних клієнтів	Готовність споживачі в сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Користувачі, що мають браузер та вихід до інтернету	Не всі користувачі готові сприйняти	Не всі будуть зацікавлені в продукті	Висока конкуренція	Вхід не є складний
2	Любителі інтелектуальних або розважальних вікторин	Готові сприйняти	Продукт повинен зацікавити всіх користувачів	Середня конкуренція	Вхід помірно-складний
3	Геймери	Високоймовірно не готові сприйняти	Продукт може не зацікавити досвідчених геймерів	Висока конкуренція	Складний вхід
Які цільові групи обрано: Потрібно працювати зі всіма цільовими групами, перевагу надати любителям жанру вікторин в цілому					

Таблиця 4.14

Визначення базової стратегії розвитку

№	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкуренто спроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1	Розвиток шляхом впровадження маркетингової кампанії	Розповсюдження інформації про продукт у різних колах споживачів	Акцентувати увагу споживачів на функціональних можливостях продукту та перспективності його розвитку	Встановлення зв’язку з користувачем та додання функціоналу, якого він потребуватиме

Таблиця 4.15

Визначення базової стратегії конкурентної поведінки

№	Чи є проект «першопро хідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
1	Проект не є першопрох ідцем на ринку	Необхідно як приваблювати нових користувачів, так і забирати існуючих у конкурентів	Так, компанія буде копіювати та покращувати основні характеристики товарів конкурентів	Пропонувати схожий функціонал на кращих умовах та оперативно його оновлювати

Таблиця 4.16

Визначення стратегії позиціонування

№	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкуренто-спроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1	Зручність у використанні, надійність, постійне оновлення та підтримка роботи додатку	Залучення штату програмістів для підтримки та розвитку проекту	Оперативне виправлення недоліків та впровадження побажань	Варіативний конструктор запитань, взаємодія гравців у реальному часі, автоматичне адміністрування гри

4.5. Розроблення маркетингової програми стартап-проекту

Таблиця 4.17

Визначення ключових переваг концепції потенційного товару

№	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити
1	Проведення ігрової сесії у онлайн-вікторині	Автоматичне адміністрування гри та взаємодія гравців у реальному часі	Не потрібно обирати ведучого для адміністрування процесу гри, перевірки наданих відповідей, нарахування балів тощо. Гравці взаємодіють між собою в реальному часі

Продовження таблиці 4.17

№	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
2	Часте оновлення програми	Врахування побажань та виправлення помилок	Створення відділу підтримки для аналізування реагування на відгуки клієнтів
3	Наявність конструктора запитань	Наявність варіативного конструктора запитань	Конструктор запитань не потрібно завантажувати окремо. Він також надає розширений функціонал для створення та редагування запитань

Таблиця 4.18

Опис трьох рівнів моделі товару

<table>
<tr><td>Рівні товару</td><td colspan="2">Сутність та складові</td></tr>
<tr><td>І. Товар за задумом</td><td colspan="2">Онлайн-вікторина «Сто запитань» у вигляді веб-додатку з вбудованим конструктором запитань</td></tr>
<tr><td rowspan="6">ІІ. Товар у реальному виконанні</td><td>Властивості/характеристики</td><td>Значення</td></tr>
<tr><td>Вікторина «Сто запитань»</td><td>Основна частина</td></tr>
<tr><td>Конструктор запитань</td><td>Для створення ігрової бази продукту</td></tr>
<tr><td>Сторінки авторизації та налаштування профілю</td><td>Для розпізнавання та відображення користувача у системі</td></tr>
<tr><td colspan="2">Якість: налаштований модуль аутентифікації, внутрішнє тестування програмного продукту</td></tr>
<tr><td colspan="2">Пакування: веб-додаток розміщений в мережі</td></tr>
<tr><td></td><td colspan="2">Марка: «Сто запитань»</td></tr>
</table>

Продовження таблиці 4.18

Рівні товару	Сутність та складові
III. Товар із підкріпленням	До продажу: стартап-ідея та програмний код
	Після продажу: повноцінна робоча система
За рахунок чого потенційний товар буде захищено від копіювання: основна логіка винесена у серверну частину, база даних також знаходиться на серверній частині, додані всі необхідні перевірки користувача при спробі доступу до сервера	

Таблиця 4.19

Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі становлення ціни на товар/послугу
1	0 грн – 200 грн	0 грн	> 150 грн	0 грн – 100 грн

Таблиця 4.20

Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Безкоштовний доступ до додатку, придбання додаткового контенту	Хостинг додатку на сервері з можливістю безкоштовного доступу до нього	Канал збуту одноріневий (через хостинг на віддаленому сервері)	Вертикальна (право власності залишається у розробника ПЗ)

Таблиця 4.21

Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Доступ до дотатку за допомогою браузера	Служба підтримки, електронна пошта	Проведення ігрових сесій, наявність конструктора запитань	Демонстрація основних функцій розробленого продукту	Охоплення аудиторії, пояснення функцій та можливостей розробленого продукту та його переваг

ВИСНОВОК ДО ЧЕТВЕРТОГО РОЗДІЛУ

Четвертий розділ присвячений розробці стартап-проекту для розроблюваного програмного продукту. Було зроблено опис ідеї проекту, проведено детальний аналіз конкурентів та можливостей, що доступні на даному ринку. Було наведено загальний опис ідеї проекту для виходу на ринок, описані можливості проекту, визначено основних конкурентів та співставлено можливості конкурентів для визначення переваг розроблюваного продукту над обраними конкурентами. Проведено технічний аудит проекту і визначено можливості реалізації програмного продукту. Було проаналізовано ринкові можливості для запуску проекту, порівняно перспективи запуску в порівнянні з конкурентами, встановлено план та ринкову стратегію розвитку продукту, визначено цільові групи та способи просування проекту. Розроблено маркетингову програму для просування продукту на ринку, описано способи та основні канали збуту, визначено пріоритетні цільові групи та маркетингові повідомлення для розширення клієнтської бази.

Після виконання вищеописаних дій було отримано повноцінний стартап-проект для запуску розробленого програмного продукту на ринок, отримано навички створення стартап-проектів та побудови маркетингової стратегії, аналізу обраного ринку.

ВИСНОВКИ

У даній магістерській роботі була розроблена онлайн-вікторина «Сто запитань», яка була виконана у вигляді веб-додатку. Розробка проводилась у середовищі WebStorm з використанням мови програмування JavaScript та фреймворку Sails.js.

Розроблений додаток дозволяє проводити ігрові онлайн-сесії відповідаючи на запитання, а гравці взаємодіють між собою у режимі онлайн. Також у додаток був доданий конструктор запитань, який дозволяє редагувати базу запитань, яка використовується у грі.

У першому розділі було виконано огляд існуючих рішень та описано актуальність розробки інтелектуально-розважальних онлайн-вікторин. Було виділено основні недоліки існуючих рішень, які були враховані при розробці даного проекту.

У другому розділі було зроблено огляд технологій, які можуть бути використані для реалізації даного проекту. Було обрано технології, які надають максимально зручний та ефективний функціонал для розробки цього програмного продукту.

У третьому розділі були описані всі етапи розробки веб-додатку. Був виконаний огляд функціоналу та описані всі можливості даного програмного продукту.

У четвертому розділі було виконано реалізацію стартап-проекту для розроблюваного продукту. Було виконано опис ідеї, аналіз сильних та слабких сторін проекту, а також можливостей та загроз для подальшого розвитку. Було проведено аналіз ринкової можливості для запуску проекту та визначено цільові групи серед яких даний проект буде просуватися.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Історія відеоігор – Вікіпедія [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Історія_відеоігор — Дата доступу: Листопад 2022.
2. Вікторина – Вікіпедія [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Вікторина — Дата доступу: Листопад 2022.
3. Web-додатки – переваги та недоліки [Електронний ресурс] – Режим доступу: https://mydiv.net/arts/view-webprilozhenija_preimuxhestva_i_nedostatki.html — Дата доступу: Листопад 2022.
4. Jeopardy! – Вікіпедія [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Jeopardy! — Дата доступу: Листопад 2022.
5. Найкращі фреймворки Nodejs для розробки додатків у 2022 році [Електронний ресурс] – Режим доступу: https://www.simform.com/blog/best-nodejs-frameworks/ — Дата доступу: Листопад 2022.
6. AdonisJS - A fully featured web framework for Node.js [Електронний ресурс] – Режим доступу: https://adonisjs.com/ — Дата доступу: Листопад 2022.
7. NestJS - A progressive Node.js framework [Електронний ресурс] – Режим доступу: https://nestjs.com/ — Дата доступу: Листопад 2022.
8. Express - фреймворк веб-застосунків Node.js [Електронний ресурс] – Режим доступу: http://expressjs.com/uk/ — Дата доступу: Листопад 2022.
9. Koa - next generation web framework for node.js [Електронний ресурс] – Режим доступу: https://koajs.com/ — Дата доступу: Листопад 2022.
10. Sails.js | Realtime MVC Framework for Node.js [Електронний ресурс] – Режим доступу: https://sailsjs.com/ — Дата доступу: Листопад 2022.

11. Різниця між веб-сокетами та Socket.IO [Електронний ресурс] – Режим доступу: https://habr.com/ru/post/498996/ — Дата доступу: Листопад 2022.
12. MVC – Вікіпедія [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Модель-вид-контролер — Дата доступу: Листопад 2022.
13. Клієнт-серверна архітектура – Вікіпедія [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Клієнт-серверна_архітектура — Дата доступу: Листопад 2022.
14. HTML: HyperText Markup Language [Електронний ресурс] – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/HTML — Дата доступу: Листопад 2022.
15. Bootstrap [Електронний ресурс] – Режим доступу: https://getbootstrap.com/docs/5.0/getting-started/introduction/ — Дата доступу: Листопад 2022.
16. Документація бібліотеки sails-adminpanel [Електронний ресурс] – Режим доступу: https://github.com/sails-adminpanel/sails-adminpanel/tree/master/docs — Дата доступу: Листопад 2022.
17. Документація бібліотеки socket.io [Електронний ресурс] – Режим доступу: https://socket.io/docs/v4/ — Дата доступу: Листопад 2022.

Додаток А

до магістерської дисертації

на тему «Онлайн-вікторина «Сто запитань»»

Лістинг фрагментів коду програми

GameController.js

```
module.exports = {
  index: function (req, res) {
    if (!req.session.user) {
      return res.redirect("/login")
    }

    return res.view('pages/game');
  },

  lobby: function (req, res) {
    if (!req.isSocket) {
      return res.badRequest();
    }

    sails.sockets.join(req, 'lobby', function(err) {
      if (err) {return res.serverError(err);}
    });
    if (!sails.rooms['lobby']) {
      sails.rooms['lobby'] = {};
    }
    sails.rooms['lobby'][req.session.user.nickname] = req.socket.id;
    sails.sockets.broadcast('lobby', 'join', {name:
req.session.user.nickname});

    return res.sendStatus(200);
  },

  createRoom: async function (req, res) {
    if (!req.isSocket) {
      return res.badRequest();
    }

    if (sails.rooms[req.session.user.nickname]) {
      return res.status(403).send("You can host one room at a time")
    }

    sails.sockets.leave(req, 'lobby');
    delete sails.rooms['lobby'][req.session.user.nickname];

    sails.sockets.join(req, req.session.user.nickname, function(err) {
      if (err) {return res.serverError(err);}
    });
    sails.rooms[req.session.user.nickname] = {};
    sails.rooms[req.session.user.nickname][req.session.user.nickname] = {};

sails.rooms[req.session.user.nickname][req.session.user.nickname]["socketId"]
= req.socket.id;

sails.rooms[req.session.user.nickname][req.session.user.nickname]["avatar"] =
req.session.user.avatar;

    let categories = await Category.find();
    let categoryNames = categories.map(category => {return category.name});
    return res.send({categories: categoryNames, roomName:
req.session.user.nickname,
      player: {nickname: req.session.user.nickname, avatar:
req.session.user.avatar}})
  },

  invite: function (req, res) {
    if (!req.isSocket) {
```

```
      return res.badRequest();
    }

    if (req.session.user.nickname !== req.body.room) {
      return res.status(403).send("Only host can invite users");
    }

    if (!req.body.invitedUser) {
      return res.status(400).send("To invite user enter nickname");
    }

    if (!sails.rooms['lobby'][req.body.invitedUser]) {
      return res.status(404).send("Invited user not present in lobby");
    }

    sails.sockets.broadcast(sails.rooms['lobby'][req.body.invitedUser],
'invite', {room: req.body.room});

    res.status(200).send("Invitation has sent")
  },

  join: function (req, res) {
    if (!req.isSocket) {
      return res.badRequest();
    }

    sails.sockets.leave(req, 'lobby');
    delete sails.rooms['lobby'][req.session.user.nickname];
    sails.sockets.join(req, req.body.room, function(err) {
      if (err) {return res.serverError(err);}
    });
    sails.rooms[req.body.room][req.session.user.nickname] = {};
    sails.rooms[req.body.room][req.session.user.nickname]["socketId"] =
req.socket.id;
    sails.rooms[req.body.room][req.session.user.nickname]["avatar"] =
req.session.user.avatar;

    sails.sockets.broadcast(req.body.room, 'updateRoomMembers', {room:
sails.rooms[req.body.room]});

    res.send({room: sails.rooms[req.body.room]});
  },

  changeCategory: function (req, res) {
    if (!req.body.columnNumber || !req.body.category || !req.body.room) {
      sails.log.error("Cannot change category without required parameters");
      return
    }

    sails.sockets.broadcast(req.body.room, 'changeCategory', {columnNumber:
req.body.columnNumber, category: req.body.category});
    res.sendStatus(200)
  },

  start: async function (req, res) {
    if (!req.body.chosenCategories || !req.body.room ||
req.body.chosenCategories.length !== 10) {
      sails.log.error("Cannot start game without required parameters");
      return
    }

    let questionsPack = [];
    let categoriesNum = await Category.count();
    let randomCategoriesNeeded = 0;
```

```
    for (let categoryName of req.body.chosenCategories) {
      if (categoryName === 'Випадкова категорія') {
        randomCategoriesNeeded++;
      } else {
        let neededCategory = await Category.findOne({name:
categoryName}).populate("questions");
        if (!neededCategory) {
          randomCategoriesNeeded++;
        } else {
          questionsPack.push(neededCategory);
        }
      }
    }

    let randomNum = Math.floor(Math.random() * (categoriesNum -
randomCategoriesNeeded));
    let randomCategories = await Category.find({skip: randomNum, limit:
randomCategoriesNeeded}).populate("questions");
    questionsPack = questionsPack.concat(randomCategories);

    for (let category of questionsPack) {
      if (category.questions && category.questions.length) {
        let filteredQuestions = [];
        for (let currentCost = 100; currentCost <= 1000; currentCost+=100) {
          let currentCostQuestions = category.questions.filter(item =>
{return item.cost == currentCost});

filteredQuestions.push(currentCostQuestions[Math.floor(Math.random() *
currentCostQuestions.length)]);
        }
        category.questions = filteredQuestions;
      } else {
        category.questions = [];
      }
    }

    sails.rooms[req.body.room].questionsPack = questionsPack;
    let gameTable = questionsPack.map(category => {
      return {name: category.name, questions: category.questions.map(question
=> {return question !== undefined ? question.id : question})}
    });

    sails.sockets.broadcast(req.body.room, 'start', {gameTable: gameTable});

    res.sendStatus(200);
  },

  question: function (req, res) {
    let category = sails.rooms[req.body.room].questionsPack.find((item) =>
{return item.name === req.body.categoryName});
    let question = category.questions.find((item) => {return item && item.id
=== req.body.questionId});
    sails.sockets.broadcast(req.body.room, 'question', {question: {id:
question.id, image: question.image, video: question.video,
      text: question.text}, category: req.body.categoryName, room:
req.body.room});
    res.sendStatus(200);
  },

  answer: function (req, res) {
    let category = sails.rooms[req.body.room].questionsPack.find((item) =>
{return item.name === req.body.categoryName});
    let question = category.questions.find((item) => {return item && item.id
=== req.body.questionId});
```

```
    let answerOptions = question.answerOptions.map(item => {return
item.answerOption});
    res.send(answerOptions);
  }
};
```

ProfileController.js

```
let path = require("path")

module.exports = {
  index: async function (req, res) {
    if (!req.session.user) {
      return res.redirect('/login');
    }

    let user = await User.findOne({nickname: req.session.user.nickname});
    if (!user) {
      return res.view('pages/profile', {title: "Створити власний профіль",
user: {}});
    }

    return res.view('pages/profile', {title: "Редагувати профіль", user:
user});
  },

  create: function (req, res) {
    return res.view('pages/profile', {title: "Створити профіль", user: {}});
  },

  save: async function(req, res) {
    if (!req.body.nickname || !req.body.password) {
      return res.send({status: 400, message: "Нікнейм та пароль є
обов'язковими полями для заовнення"})
    }

    if (req.session.user && req.session.user.nickname === req.body.nickname)
{
      let user = await User.findOne({nickname: req.body.nickname});
      if (!user) {
        return res.send({status: 404, message: "Неможливо оновити профіль,
користувач з таким ім'ям не знайдений"})
      }

      let oldAvatar = user.avatar;
      let avatar = await uploadAvatar(req);
      if (!avatar) {
        avatar = oldAvatar
      } else {
      }

      await User.update({nickname: req.body.nickname}, {password:
req.body.password, avatar: avatar})

      req.session.user = {
        type: "authorized",
        nickname: req.body.nickname,
        avatar: avatar
      }
      return res.send({status: 200, message: "Профіль було успішно
оновлено"})
    } else {
      let duplicateNickname = await User.findOne({nickname:
```

```
req.body.nickname});
      if (duplicateNickname) {
        return res.send({status: 403, message: "Користувач з таким ім'ям вже
зареєстрований, будь-ласка оберіть інше ім'я"})
      }

      let avatar = await uploadAvatar(req);
      await User.create({nickname: req.body.nickname, password:
req.body.password, avatar: avatar})
      req.session.user = {
        type: "authorized",
        nickname: req.body.nickname,
        avatar: avatar
      }
      return res.send({status: 200, message: "Профіль було створено
успішно"})
    }
  }
};

async function uploadAvatar(req) {
  return new Promise(function (resolve, reject) {
    req.file("avatar").upload({
      dirname: `../public/avatars`,
    }, function (err, uploadedFiles) {
      if (err) {
        sails.log.error(err);
        return reject(err)
      }
      if (uploadedFiles && uploadedFiles.length) {
        let filename = path.basename(uploadedFiles[0].fd)
        return resolve([{
          "id": 1, name: uploadedFiles[0].filename, "url":
`/avatars/${filename}`,
          "urlS": `/avatars/${filename}`, "urlL": `/avatars/${filename}`
        }]);
      }
      return resolve();
    })
  })
}
```

AuthController.js

```
const generator = require('nickname-generator');

module.exports = {
  index: function (req, res) {
    return res.view('pages/login')
  },

  login: async function (req, res) {
    let user = await User.findOne({nickname: req.body.nickname, password:
req.body.password});
    if (!user) {
      return res.sendStatus(401);
    } else {
      req.session.user = {
        type: "authorized",
        nickname: user.nickname,
        avatar: user.avatar
      }
      return res.sendStatus(200)
```

```
    }
  },

  guest: function (req, res) {
    const nickname = generator.randomNickname();
    req.session.user = {
      type: 'guest',
      nickname: nickname,
      avatar: [{
        "id": 1, name: "Guest avatar", "url": "/avatars/userGuestIcon.png",
        "urlS": "/avatars/userGuestIcon.png", "urlL":
"/avatars/userGuestIcon.png"
      }]
    }
    res.redirect('/');
  },

  logout: function (req, res) {
    delete req.session.user;
    return res.redirect("/");
  }
}
```

InfoController.js

```
module.exports = {
  index: function (req, res) {
    if (!req.session.user) {
      return res.redirect('/login')
    }

    return res.view('pages/info');
  }
};
```

MainMenuController.js

```
module.exports = {
  index: function (req, res) {
    if (!req.session.user) {
      return res.redirect("/login")
    }

    return res.view('pages/mainMenu', {user: req.session.user})
  }
}
```

RulesController.js

```
module.exports = {
  index: function (req, res) {
    if (!req.session.user) {
      return res.redirect('/login')
    }

    return res.view('pages/info');
  }
};
```