**ДОДАТОК В**

Специфікація програмного забезпечення

**CS330 Software Engineering**

**Software Requirements Specification (SRS) Template**

Items that are intended to stay in as part of your document are in **bold**; explanatory comments are in *italic* text. Plain text is used where you might insert wording about your project.

The document in this file is an annotated outline for specifying software requirements, adapted from the IEEE Guide to Software Requirements Specifications (Std 830-1993).

Tailor this to your needs, removing explanatory comments as you go along. Where you decide to omit a section, keep the header, but insert a comment saying why you omit the data.

Software System for Styling and Smart Use of Clothes

SmartStyle Software Engineer

Vlad Puatnizki – Frontend Developer

Kovalevych Daniil – Backend Developer

SmartStyle Project Team

**Software Requirements Specification**

**Document**

**Version: 1.0 Date: 20.06.2025**

**ЗМІСТ**

**1. ВСТУП**

1.1 Огляд продукту

1.2 Мета

1.3 Межі

1.4 Посилання

1.5 Означення та абревіатури

**2. ЗАГАЛЬНИЙ ОПИС**

2.1 Перспективи продукту

2.2 Функції продукту

2.3 Характеристики користувачів

2.4 Загальні обмеження

2.5 Припущення й залежності

**3. КОНКРЕТНІ ВИМОГИ**

3.1 Вимоги до зовнішніх інтерфейсів

3.1.1 Інтерфейс користувача

3.1.2 Апаратний інтерфейс

3.1.3 Програмний інтерфейс

3.1.4 Комунікаційний протокол

3.1.5 Обмеження пам’яті

3.1.6 Операції

3.1.7 Функції продукту

3.1.8 Припущення й залежності

3.2 Властивості програмного продукту

3.3 Атрибути програмного продукту

3.3.1 Надійність

3.3.2 Доступність

3.3.3 Безпека

3.3.4 Супроводжуваність

3.3.5 Переносимість

3.3.6 Продуктивність

3.4 Вимоги бази даних

3.5 Інші вимоги

**1. ВСТУП**

1.1 Огляд продукту

"Quiz Party Hub" — це сучасний веб-застосунок, створений для інтерактивного проведення вікторин, який дозволяє користувачам генерувати квізи за довільною тематикою, проходити їх у зручному форматі та змагатися за очки в таблиці лідерів. Застосунок орієнтований як на індивідуальних гравців, так і на групи користувачів, які бажають урізноманітнити навчальний або розважальний процес.

Продукт поєднує функціональність генерації тестових завдань на основі обраної теми, проходження квізу з інтерактивним зворотним зв’язком, авторизації через обліковий запис Google та систему зберігання балів і рейтингів. Завдяки інтуїтивно зрозумілому інтерфейсу, "Quiz Party Hub" забезпечує зручний досвід користування як на ПК, так і на мобільних пристроях.

1.2 Мета

Основною метою проєкту "Quiz Party Hub" є створення інтуїтивно зрозумілого та зручного у використанні веб-застосунку, що забезпечує користувачам можливість:

* Генерувати тести за довільною темою без необхідності ручного введення запитань.
* Проходити квізи у режимі реального часу з інтерактивним відображенням результатів.
* Отримувати миттєвий зворотний зв’язок щодо правильності відповідей.
* Накопичувати бали та переглядати своє місце в загальній таблиці лідерів.
* Ознайомлюватися з відгуками інших користувачів і ділитися власними враженнями.

1.3 Межі

Проєкт "Quiz Party Hub" зосереджений на розробці веб-застосунку з такими ключовими функціональними можливостями:

* Авторизація та автентифікація користувачів за допомогою Google-акаунтів.
* Генерація квізів за довільно введеною тематикою через зовнішній API.
* Послідовне відображення запитань з варіантами відповідей.
* Перевірка правильності обраної відповіді та надання візуального зворотного зв’язку.
* Нарахування балів за правильні відповіді та збереження рейтингу в базі даних.
* Відображення загального результату після завершення квізу.
* Публічний рейтинг користувачів (таблиця лідерів).
* Відображення додаткових інформаційних блоків: «Про гру», «Відгуки», «Контакти».
* Адаптивний інтерфейс, оптимізований для різних розмірів екранів.

1.4 Посилання

У процесі розробки програмного забезпечення було використано низку зовнішніх джерел, які забезпечили технічну підтримку, довідкову інформацію та стандарти, що лежать в основі реалізації системи. Основні з них наведені нижче:

* Кодова база проєкту "Quiz Party Hub" (локальний та віддалений репозиторій GitHub).
* Документація до зовнішнього API для генерації квізів (використовується REST API із підтримкою тематики).
* Документація Firebase (Authentication, Firestore): https://firebase.google.com/docs
* Документація React: <https://react.dev/>
* Стандарти розробки веб-додатків W3C:<https://www.w3.org/standards/webdesign/>

1.5 Означення та абревіатури

У цьому підрозділі наведено основні терміни, абревіатури та технічні поняття, що використовуються в межах документації та під час реалізації програмного забезпечення. Їх тлумачення дозволяє краще зрозуміти логіку проєкту та застосовані технології:

* UI (User Interface): Користувацький інтерфейс.
* UX (User Experience): Досвід користувача у взаємодії з продуктом.
* API (Application Programming Interface): Програмний інтерфейс для взаємодії з зовнішніми сервісами.
* JWT (JSON Web Token): Формат токенів для авторизації користувача у веб-застосунках.
* SPA (Single Page Application): Односторінковий застосунок, який не потребує перезавантаження сторінки.
* Firebase: Платформа від Google для бекенд-розробки (автентифікація, база даних, хостинг тощо).
* React: JavaScript-бібліотека для побудови користувацьких інтерфейсів.
* Frontend: Клієнтська частина застосунку, що працює безпосередньо у браузері користувача.
* Backend: Серверна частина, яка обробляє запити, зберігає й надає дані.
* CRUD: Основні операції з даними — створення, читання, оновлення, видалення (Create, Read, Update, Delete).
* Leaderboard: Таблиця з результатами користувачів, сформована за рейтингом.
* Axios: Бібліотека для виконання HTTP-запитів з клієнтської частини.
* JSON: Формат обміну структурованими даними між клієнтом і сервером.

**2. ЗАГАЛЬНИЙ ОПИС**

2.1 Перспективи продукту

"Quiz Party Hub" — це автономний веб-застосунок, який працює за клієнт-серверною моделлю. Клієнтська частина реалізована з використанням бібліотеки React, що забезпечує динамічний рендеринг інтерфейсу, а також інтерактивну взаємодію з користувачем у режимі односторінкового застосунку (SPA).

Застосунок взаємодіє з бекенд-сервісами за допомогою RESTful API. Основні операції з даними (генерація квізів, оновлення балів, формування рейтингу) виконуються через асинхронні HTTP-запити. Як бекенд-інфраструктура використовується Firebase (Authentication, Firestore), а генерація тестових завдань відбувається за допомогою стороннього API на основі заданої теми.

Продукт проєктувався як незалежна система, що не потребує сторонніх систем управління контентом або складної серверної логіки. Основний фокус — швидкий запуск, гнучкість і зручність користувача без необхідності встановлення чи завантаження.

2.2 Функції продукту

Програмне забезпечення реалізує низку основних функцій, які забезпечують повний цикл взаємодії користувача з системою: від авторизації до отримання результатів. Нижче наведено ключові можливості, реалізовані у межах проєкту:

* Автентифікація користувачів: Вхід у систему та вихід за допомогою Google-акаунту через Firebase Authentication.
* Генерація квізів: Надсилання запиту до зовнішнього API з темою, введеною користувачем, та отримання списку запитань.
* Проходження квізу: Покрокове відображення запитань з варіантами відповідей, інтерактивний вибір, миттєвий фідбек.
* Оцінювання відповідей: Перевірка правильності кожної відповіді, візуальне відображення результату.
* Нарахування балів: Підрахунок очок за правильні відповіді з оновленням рейтингу у Firestore.
* Вивід результатів: Показ підсумкового результату після завершення квізу.
* Таблиця лідерів: Формування та відображення глобального рейтингу користувачів за кількістю очок.
* Інформаційні блоки: Відображення сторінок «Про гру», «Відгуки», «Контакти» з коротким описом та стилізованими елементами.
* Адаптивність: Інтерфейс, що коректно відображається на різних пристроях (ПК, планшети, мобільні телефони).
* Локалізація (перспектива): Підготовка до впровадження багатомовної підтримки (українська, англійська).

2.3 Характеристики користувачів

Програмне забезпечення орієнтоване на різні категорії користувачів залежно від рівня доступу до функціональності. Визначено дві основні групи користувачів, кожна з яких має власні можливості та обмеження:

* Зареєстровані користувачі: Основна цільова аудиторія застосунку. Це користувачі, які здійснили автентифікацію через Google-акаунт і мають доступ до повного функціоналу, зокрема генерації квізів, проходження тестів, нарахування балів та перегляду таблиці лідерів. Передбачається, що вони володіють базовими навичками взаємодії з веб-застосунками та знайомі з поняттями акаунту, авторизації та інтерактивного інтерфейсу.
* Незареєстровані користувачі: Гості вебзастосунку, які можуть ознайомитися з загальнодоступними сторінками — «Про гру», «Відгуки», «Контакти» — але не мають доступу до ігрового функціоналу. Для взаємодії із квізами їм буде запропоновано увійти в систему.

2.4 Загальні обмеження

Під час реалізації програмного забезпечення були виявлені низка технічних та організаційних обмежень, які можуть впливати на функціональність системи, масштабованість або стабільність роботи. Основні з них перелічено нижче:

* Залежність від зовнішніх API: Генерація квізів здійснюється через зовнішній API, тому працездатність функціоналу напряму залежить від стабільності цього сервісу та якості відповіді. У разі недоступності API створення тестів неможливе.
* Локальне розгортання бекенду: На етапі розробки бекенд-сервер працює локально, що обмежує доступ до застосунку лише авторизованими тестувальниками з відповідним середовищем. Публічне використання можливе лише після розгортання на зовнішньому хостингу.
* Веб-орієнтований інтерфейс: Застосунок розроблено як веб-додаток, оптимізований під браузери. Наразі відсутня підтримка або адаптація під нативні мобільні середовища (Android, iOS).
* Одноосібна сесія: Кожен користувач проходить квіз індивідуально; функціонал багатокористувацької взаємодії (наприклад, змагання в реальному часі) не реалізовано.
* Безпечне зберігання результатів: Всі користувацькі дані зберігаються у Firestore, однак захист від зловживань покладається на авторизацію через Firebase, без додаткової серверної валідації з боку бекенду.

2.5 Припущення й залежності

Для коректної роботи програмного забезпечення передбачено низку початкових умов, технічних вимог та зовнішніх залежностей. Їх дотримання є критично важливим для стабільного функціонування системи:

* Користувачі повинні мати стабільне інтернет-з'єднання для завантаження інтерфейсу, авторизації та взаємодії з API.
* API-ключі для зовнішнього генератора запитань мають бути заздалегідь налаштовані у середовищі бекенд-серверу та залишатися дійсними протягом усього часу використання.
* Обліковий запис Firebase повинен бути активним, а сервіси Firestore та Authentication — коректно налаштованими.
* Бекенд-сервер і фронтенд-додаток повинні бути розгорнуті в сумісних середовищах, із підтримкою CORS і безпечного обміну токенами (JWT).
* Браузери користувачів повинні підтримувати сучасні вебтехнології, включно з JavaScript ES6+, fetch API, localStorage та адаптивну верстку.

**3. КОНКРЕТНІ ВИМОГИ**

3.1 Вимоги до зовнішніх інтерфейсів

Програмне забезпечення «Quiz Party Hub» взаємодіє з низкою зовнішніх компонентів, які забезпечують його повноцінне функціонування. Зовнішні інтерфейси охоплюють користувацький інтерфейс (UI), програмні інтерфейси (API), а також інтерфейси бази даних. Нижче наведено основні вимоги до кожного типу.

3.1.1 Інтерфейс користувача

Інтерфейс користувача вебзастосунку розроблено відповідно до сучасних стандартів зручності, адаптивності та інтерактивності. Основна мета — забезпечити просту, зрозумілу та приємну взаємодію з системою. Нижче наведено ключові характеристики інтерфейсу:

* Дизайн: Сучасний, мінімалістичний і адаптивний дизайн, орієнтований на простоту та зручність користувача. Інтерфейс підтримує коректне відображення як на настільних, так і на мобільних пристроях. Кольорова палітра включає світлий фон з акцентами синього та бірюзового кольору.
* Навігація: Інтуїтивно зрозуміла навігаційна панель розміщена у верхній частині сторінки (компонент Header), забезпечує доступ до ключових розділів: «Квіз», «Лідери», «Про гру», «Відгуки», «Контакти». Додатково у футері (Footer) дублюються контактні дані та навігаційні посилання.
* Елементи управління: Для кнопок, полів вводу та таблиць використано стилізовані React-компоненти з класами Tailwind CSS. Активні елементи мають стан наведення (hover) та візуальний відгук при натисканні.
* Відображення результатів: Кнопки відповіді автоматично підсвічуються зеленим або червоним кольором після вибору. Реалізовано чіткий зворотний зв’язок для користувача без перезавантаження сторінки.
* Доступність: Тексти, заголовки та кнопки мають достатній контраст і розмір для комфортного перегляду. Розроблено базову мобільну адаптацію без скролів по горизонталі.

3.1.2 Апаратний інтерфейс

Вимоги до апаратного забезпечення для роботи вебзастосунку "Quiz Party Hub" є типовими для сучасних односторінкових застосунків і не передбачають використання спеціалізованого обладнання.

З боку клієнта, для повноцінної взаємодії із застосунком користувачу достатньо будь-якого пристрою з підтримкою сучасного веб-браузера (Google Chrome, Mozilla Firefox, Microsoft Edge тощо). Це може бути персональний комп’ютер, ноутбук, планшет або смартфон. Обов’язковою умовою є наявність стабільного доступу до мережі Інтернет. Такий підхід забезпечує максимальну доступність сервісу для широкої аудиторії.

З боку сервера, для розгортання бекенд-серверу (за умови виходу за межі локального середовища) потрібна наявність хостинг-середовища з можливістю підтримки Node.js та інтеграції з Firebase. Конкретні апаратні вимоги (CPU, RAM, диск) залежать від кількості одночасних запитів, частоти генерації квізів та обсягу збережених даних, але для етапу розробки та тестування достатньо локального середовища розробника.

3.1.3 Програмний інтерфейс

Програмна архітектура "Quiz Party Hub" побудована на взаємодії кількох ключових програмних компонентів та сучасних вебтехнологій.

Клієнтська частина (Frontend) реалізована мовою JavaScript (стандарт ES6+) з використанням бібліотеки React.js, яка є основою для побудови адаптивного інтерфейсу та динамічного управління станами. Навігація між екранами реалізована через умовне рендерування компонентів, без використання сторонніх роутерів, що відповідає концепції SPA (Single Page Application). Взаємодія з серверною частиною виконується за допомогою бібліотеки Axios, яка забезпечує надсилання асинхронних HTTP-запитів у форматі JSON.

Візуальне оформлення реалізовано із застосуванням Tailwind CSS, що дозволяє швидко стилізувати компоненти за допомогою утилітарних класів. Вибрані елементи інтерфейсу (кнопки, блоки результатів, аватари) відображаються інтерактивно, із динамічною зміною стилів у залежності від дій користувача.

Серверна частина (Backend) на етапі розробки функціонує як окремий RESTful API, розгорнутий локально. Генерація запитань до квізів здійснюється через зовнішній API, а обробка балів користувачів виконується з використанням Firebase Cloud Firestore. Авторизація користувачів реалізована через Firebase Authentication, що дозволяє виконувати вхід за допомогою Google-акаунтів.

У якості хмарної бази даних використовується Firebase Firestore, що дозволяє зберігати дані користувачів, їхні результати, а також формувати рейтинг на основі накопичених балів.

Система інтегрується з такими зовнішніми програмними інтерфейсами:

* API генерації квізів — для створення питань на основі довільно введеної теми;
* Firebase SDK — для авторизації, роботи з базою даних, зчитування інформації про користувача;
* Google Identity Platform — як провайдер для Google Sign-In.

3.1.4 Комунікаційний протокол

Взаємодія між клієнтською та серверною частинами системи "Quiz Party Hub" реалізована з використанням протоколів HTTP/HTTPS, що є стандартом для вебзастосунків. Обмін даними здійснюється відповідно до архітектурного стилю REST (Representational State Transfer), який передбачає безстатусну комунікацію (stateless), де кожен запит містить усю необхідну інформацію для його обробки.

Формат обміну даними між клієнтом (React-застосунок) і сервером або зовнішніми API — JSON (JavaScript Object Notation). Цей формат обрано як оптимальний для JavaScript-орієнтованих середовищ завдяки його легкості, читабельності та сумісності з більшістю клієнтських і серверних платформ.

Автентифікація користувача реалізована на основі Firebase Authentication, яка після успішного входу надає унікальний ID токен у форматі JWT (JSON Web Token). Цей токен включається в заголовок Authorization кожного наступного запиту до захищених ресурсів API у вигляді:

**Authorization: Bearer <token>**

Сторонні API (наприклад, генератор квізів) також використовуються в режимі REST, з аутентифікацією через API-ключ або аналогічний токен доступу.

Такий підхід дозволяє забезпечити:

* масштабованість застосунку;
* безпеку доступу до персональних даних;
* гнучкість у розширенні або заміні окремих сервісів у майбутньому.

3.1.5 Обмеження пам’яті

Хоча система не має жорстко встановлених обмежень на використання пам’яті, її архітектура спроєктована з урахуванням принципів ефективності та економного використання ресурсів на стороні як клієнта, так і сервера.

На клієнтській стороні, вебзастосунок оптимізований для мінімального споживання оперативної пам’яті браузера. Це досягається завдяки обмеженому одночасному рендерингу компонентів, ефективному використанню хуків React (useState, useEffect) та видаленню непотрібних елементів з DOM після завершення взаємодії. Підхід дозволяє зберігати продуктивність навіть на пристроях із базовими технічними характеристиками, таких як планшети або бюджетні ноутбуки.

На серверній стороні, зберігання та обробка даних реалізовані з урахуванням оптимального використання пам’яті. Замість обробки великих об’ємів даних у вигляді вкладених об’єктів або медіа, система зберігає лише мінімально необхідні текстові поля (наприклад, ім’я, тема квізу, очки). Усі зображення аватарів користувачів завантажуються через URL-адреси, які надає Google-профіль, і не зберігаються локально. Це дозволяє уникнути перевантаження бази даних Firebase і покладатись на хмарну інфраструктуру для медіа-ресурсів.

Архітектура дозволяє масштабувати систему за кількістю користувачів без істотного впливу на обсяг оперативної пам’яті серверного оточення.

3.1.6 Операції

Програмний інтерфейс застосунку (API) надає набір кінцевих точок (endpoints) для виконання основних функціональних операцій. Усі запити надсилаються у форматі JSON за допомогою HTTP-протоколу. Операції логічно згруповані відповідно до функціональних блоків:

1) Авторизація користувача (/api/auth): Набір операцій, відповідальний за аутентифікацію через Google:

* + POST /api/auth/google – ініціація входу користувача через Google-акаунт;
  + GET /api/auth/session – перевірка наявності активної сесії користувача;
  + POST /api/auth/logout – вихід із системи.

2) Генерація квізу (/api/quiz): Обробка теми, введеної користувачем, та генерація запитань:

* + POST /api/quiz/generate – надсилання теми для генерації квізу;
  + GET /api/quiz/:id – отримання масиву питань за ідентифікатором квізу (локально або з кешу).

3) Збереження результату (/api/score): Операції, пов’язані з підрахунком балів та оновленням таблиці лідерів:

* + POST /api/score/update – збереження результатів користувача;
  + GET /api/leaderboard – отримання актуального рейтингу користувачів;
  + GET /api/score/me – перегляд особистої статистики користувача.

4) Статичні сторінки (/api/info): Отримання додаткової інформації:

* + GET /api/info/about – сторінка «Про гру»;
  + GET /api/info/feedback – отримання відгуків;
  + GET /api/info/contacts – контактна інформація.

3.1.7 Функції продукту

Ключовий функціонал продукту визначається набором можливостей, доступних кінцевому користувачеві через вебінтерфейс. Система забезпечує повний цикл автентифікації та персоналізованого досвіду, ґрунтуючись на авторизації через Google-акаунт.

У межах особистого кабінету користувач має змогу взаємодіяти з системою через інтуїтивно зрозумілий інтерфейс. Після авторизації йому доступна центральна функція — створення квізу за власною темою. Для цього необхідно ввести тему, після чого система надсилає запит до зовнішнього API для генерації тестових запитань.

Процес проходження квізу реалізований у покроковому форматі: питання відображаються по одному, відповіді вибираються з варіантів, а система надає миттєвий візуальний зворотний зв’язок про правильність. За кожну правильну відповідь користувачу нараховуються бали.

Після завершення квізу система автоматично відображає підсумковий результат, а також оновлює таблицю лідерів. Рейтинг формується динамічно, з урахуванням накопичених очок, та відображається у загальнодоступному форматі.

Додаткові функції включають ознайомлення з розділом «Про гру», перегляд користувацьких відгуків та контактної інформації. Всі ці блоки інтегровані у єдиний SPA-інтерфейс та доступні без перезавантаження сторінки.

3.1.8 Припущення й залежності

Коректна робота системи "Quiz Party Hub" ґрунтується на низці технічних припущень та зовнішніх залежностей, від яких безпосередньо залежить функціональність застосунку.

Передбачається, що:

* користувач має активне підключення до Інтернету та використовує сучасний браузер, сумісний з JavaScript ES6+, React та функціональністю Firebase;
* авторизація користувача виконується через Google Sign-In, тому залежність від платформи Firebase Authentication є критичною;
* дані про користувача (зокрема, user.uid та user.displayName) коректно зчитуються з контексту автентифікації та не змінюються вручну;
* API-ключі до сервісу генерації квізів збережені у змінних середовища або у файлі .env з обмеженим доступом, і не виводяться у відкритому коді;
* зовнішнє API, що використовується для генерації запитань, доступне та працює стабільно (від цього залежить ключовий функціонал продукту);
* Firestore функціонує в режимі реального часу та гарантує збереження даних (бали, результати, рейтинг);
* користувачі взаємодіють із системою добросовісно, не намагаючись модифікувати дані вручну з браузера (вразливості типу localStorage чи підміна запитів не фільтруються сервером на рівні продакшн-логіки).

Ці припущення дозволяють зосередитись на реалізації клієнтської логіки без впровадження додаткових серверних валідацій на етапі прототипу.

3.2 Властивості програмного продукту

Система "Quiz Party Hub" повинна забезпечувати повний набір функціональностей, необхідних для проходження користувачем повного циклу взаємодії з додатком — від входу до перегляду результатів та участі в загальному рейтингу.

Після входу через Google-акаунт користувач повинен мати доступ до основного функціоналу без повторної авторизації при перезавантаженні сторінки. Сесія зберігається за допомогою Firebase Authentication та локального токена доступу.

Ключовою функціональною вимогою є надання можливості генерації квізів за заданою темою, проходження запитань у покроковому режимі, перевірки правильності відповідей та відображення результату. Для цього система повинна забезпечувати повноцінну підтримку операцій типу CRUD для обробки:

* тем користувача (створення);
* згенерованих запитань (читання);
* результатів квізу (оновлення);
* рейтингу та історії (видалення – за потреби).

Додатково система повинна надавати доступ до загальних сторінок («Про гру», «Контакти», «Відгуки»), вести облік результатів у реальному часі та оновлювати таблицю лідерів у Firestore.

Інтерфейс має бути адаптивним для різних типів пристроїв і підготовленим до реалізації багатомовності (українська, англійська), навіть якщо на поточному етапі реалізована лише одна мова.

3.3 Атрибути програмного продукту

Програмний продукт повинен відповідати певним якісним характеристикам, які забезпечують його зручність, надійність та ефективність у використанні. Основні атрибути якості охоплюють зручність обслуговування, портативність, безпеку та інші властивості, що мають критичне значення для користувача та розробника.

3.3.1 Надійність

Система "Quiz Party Hub" спроєктована з урахуванням високих вимог до надійності та стабільності роботи у вебсередовищі. Надійність забезпечується комплексною обробкою потенційних помилок як на клієнтській стороні (у React-компонентах), так і на стороні серверних API-запитів.

Усі критичні запити (автентифікація, генерація квізу, збереження результатів) супроводжуються обробкою винятків та відповідним виведенням повідомлень для користувача. У випадках мережевої помилки або недоступності зовнішнього API застосунок коректно обробляє ситуацію без перезавантаження сторінки чи втрати даних.

Цілісність даних забезпечується засобами платформи Firebase Firestore, яка гарантує атомарність операцій запису та узгодженість колекцій (зокрема, при оновленні рейтингу). Додатково використовується контроль доступу до документів бази через Firestore Security Rules, що унеможливлює некоректні або неавторизовані зміни з боку клієнта.

Завдяки розподіленій природі Firebase інфраструктури, система зберігає працездатність навіть при часткових збоїх у мережі або зниженні швидкості з'єднання, що є важливою перевагою при роботі з мобільних пристроїв.

3.3.2 Доступність

Вебзастосунок "Quiz Party Hub" розробляється з урахуванням вимог до максимальної доступності та надійної роботи у режимі 24/7, за винятком коротких інтервалів можливого планового технічного обслуговування або оновлення клієнтської частини.

Система оптимізована для доступу з різних пристроїв — настільних ПК, ноутбуків, планшетів і смартфонів — через будь-який сучасний веббраузер. Завдяки адаптивній верстці та мінімалістичному дизайну користувацький інтерфейс легко сприймається навіть на невеликих екранах.

Окрему увагу приділено зручності використання: інтерфейс побудований за принципом логічної послідовності дій. Всі ключові функції (вхід, створення квізу, проходження тесту, перегляд результатів) доступні в один-два кліки. Це дозволяє користувачам з різним рівнем цифрової компетентності швидко освоїти систему без необхідності у додатковій інструкції чи навчанні.

3.3.3 Безпека

Безпека є одним із ключових атрибутів системи "Quiz Party Hub", оскільки вона оперує персональними даними користувачів та результатами проходження квізів.

Для захисту облікових записів використовується Firebase Authentication, що забезпечує безпечну автентифікацію через облікові записи Google. Усі сесії обслуговуються через токени доступу (ID token) з обмеженим терміном дії. Ці токени автоматично оновлюються та не зберігаються у відкритому вигляді.

Авторизація здійснюється на основі унікального ідентифікатора користувача (uid), що надається Firebase. Це гарантує, що кожен користувач має доступ виключно до своїх особистих даних (наприклад, результатів квізів або балів у рейтингу).

Усі дані в базі Firestore мають захист на рівні Security Rules, що дозволяє обмежити доступ до документів лише для авторизованих користувачів з відповідним правом доступу. Додатково, всі запити між клієнтом і сервером відбуваються по зашифрованому каналу (HTTPS).

Також система розроблена з урахуванням базових засобів захисту від XSS (вставка скриптів у відповіді або імена) та не використовує SQL, що повністю виключає ризик SQL-інʼєкцій.

3.3.4 Супроводжуваність

Система "Quiz Party Hub" розроблена з урахуванням вимог до легкої підтримки, масштабування та подальшого розвитку. Архітектура застосунку побудована на принципах модульності — кожна частина функціоналу реалізована як окремий компонент або логічний блок.

Код написаний відповідно до загальноприйнятих стандартів розробки JavaScript/React, включаючи форматування, іменування змінних та розділення відповідальностей. Для складних функцій та нетривіальних блоків реалізації надані коментарі, що пояснюють логіку роботи, особливо в частинах, повʼязаних з обробкою відповідей, нарахуванням балів та оновленням рейтингу.

Усі зовнішні залежності чітко задекларовані у файлі package.json, що дозволяє швидко ініціалізувати проєкт на новій машині або сервері. Середовище розробки передбачає використання стандартного менеджера пакетів npm, що забезпечує кросплатформеність та автоматичне керування версіями бібліотек.

Також проєкт містить структуровану файлову організацію, де логіка, стилі, компоненти та утиліти розділені по папках, що значно полегшує навігацію та внесення змін у майбутньому.

3.3.5 Переносимість

Система "Quiz Party Hub" спроєктована з урахуванням високої переносимості та сумісності з різними платформами.

Клієнтська частина (Frontend), розроблена на базі React.js, забезпечує крос-браузерну сумісність і стабільно функціонує у сучасних версіях основних веб-браузерів, включно з Google Chrome, Mozilla Firefox, Safari та Microsoft Edge. Адаптивна верстка дозволяє використовувати застосунок як на десктопних, так і на мобільних пристроях без потреби у встановленні окремих додатків.

Серверна частина, яка взаємодіє з зовнішніми API та Firebase, базується на технологіях Node.js та хмарних сервісах Google, що забезпечує кросплатформеність. Серверна логіка та інфраструктура можуть бути розгорнуті як на локальній машині розробника (Windows, macOS, Linux), так і у хмарних середовищах (Vercel, Firebase Hosting, Heroku тощо).

Завдяки використанню безсерверних технологій у Firebase, розгортання та масштабування можливо здійснювати без зміни коду залежно від середовища виконання.

3.3.6 Продуктивність

Продуктивність є ключовим нефункціональним атрибутом системи "Quiz Party Hub". Застосунок розроблено таким чином, щоб забезпечити швидку реакцію інтерфейсу та мінімізувати затримки при взаємодії з користувачем.

Час відповіді API для стандартних запитів (генерація квізу, збереження результатів, оновлення рейтингу) не повинен перевищувати 500 мс за нормальних умов роботи мережі. Це досягається за рахунок використання асинхронних запитів через Axios та оптимізації структури запитів.

Час завантаження основних сторінок (головна, квіз, лідери, проєкт) оптимізований завдяки попередньому кешуванню стилів, попередньому рендерингу компонентів та мінімізації обсягу передаваних ресурсів. Додаток дотримується принципів лінивої загрузки (lazy loading), що покращує час першого відображення.

Розуміючи, що генерація запитань залежить від стороннього API, в інтерфейсі реалізовано візуальні індикатори завантаження (анімовані спінери), які інформують користувача про стан обробки запиту. Це забезпечує позитивний користувацький досвід навіть у разі незначної затримки відповіді сервера.

3.4 Вимоги бази даних

У проєкті "Quiz Party Hub" для зберігання та обробки даних використовується хмарна нереляційна база даних Firestore, яка входить до складу платформи Firebase. Firestore забезпечує масштабоване зберігання у форматі документів, що особливо зручно для SPA-застосунків із гнучкою структурою даних.

Замість класичних реляційних таблиць система оперує колекціями документів, основними з яких є:

* users — містить інформацію про користувачів, автентифікованих через Google (UID, ім’я, аватар, загальні бали);
* scores — зберігає результати проходження квізів;
* leaderboard — агрегована таблиця з лідерами системи;
* feedbacks — записи користувацьких відгуків.

Оскільки Firestore є документно-орієнтованою СУБД, вона не потребує явного визначення зовнішніх ключів, проте логічні зв’язки реалізуються через UID користувача та референси в полях документів. Це дозволяє уникнути складних операцій JOIN та забезпечує високу продуктивність при читанні даних.

Для підвищення ефективності:

* використовуються індекси на основі частоти запитів, наприклад, для uid, totalPoints, timestamp;
* реалізовано обмеження на структуру документів для збереження узгодженості;
* доступ до бази керується Firebase Security Rules, які гарантують, що користувачі мають доступ лише до власних даних.

Firestore обрано як основну СУБД через її гнучкість, масштабованість, підтримку роботи в реальному часі та інтеграцію з іншими сервісами Firebase.

3.5 Інші вимоги

Окрім основних функціональних та нефункціональних характеристик, до системи "Quiz Party Hub" також висуваються додаткові вимоги, які впливають на зручність використання та якість підтримки.

* Зручність взаємодії: Інтерфейс повинен бути логічно структурованим та інтуїтивно зрозумілим, щоб користувач з мінімальним технічним досвідом міг без труднощів взаємодіяти із системою. Усі дії (вхід, запуск квізу, вибір відповіді, перегляд результатів) мають виконуватись у два-три кліки без перевантаження інтерфейсу.
* Документування коду: Внутрішня структура проєкту повинна містити належну документацію у вигляді коментарів до функцій, змінних та логічних блоків коду, що дозволяє іншим розробникам швидко орієнтуватися в кодовій базі.
* Обробка помилок: Повідомлення про помилки повинні бути не лише технічно коректними, але й зрозумілими для кінцевого користувача. У разі виникнення збоїв (наприклад, неможливість отримати питання через API або втрата з’єднання), інтерфейс має відображати відповідне інформативне повідомлення з підказкою щодо можливих дій користувача.
* Адаптація під майбутнє розширення: Архітектура повинна дозволяти подальше розширення функціоналу без повної перебудови проєкту — наприклад, додавання багатокористувацьких режимів або локалізації.