iNeuron

QUIZ GAME APP

High Level Design (HLD) iNeuron

Date Issued	Version	Description	Author
15/06/24	1	Abtract and intro	sneha
16/0624	2	design	sneha
19/06/24	3	medthodolgy	sneha
20/06/24	4	conclusion	sneha
20/06/24	5	Final	sneha

Contents

1. Introduction

1.1 Purpose

The purpose of this High-Level Design (HLD) document is to outline the overall architecture and design of the Quizzy Bee application. It provides a comprehensive view of the system components, their interactions, and the underlying data structures to ensure a cohesive and efficient implementation.

1.2 Scope

This document covers the high-level design aspects of the Quizzy Bee application, including system architecture, module descriptions, database design, interface design, and security considerations.

1.3 Definitions, Acronyms, and Abbreviations

- **HLD**: High-Level Design
- **API**: Application Programming Interface
- **UI**: User Interface
- **DB**: Database

2. Overall Description

2.1 Product Perspective

Quizzy Bee is an interactive quiz application designed to provide users with engaging and educational quizzes on various topics. It aims to enhance learning through a gamified experience.

2.2 Product Functions

- User registration and login
- Creating and managing quizzes
- Taking quizzes
- Viewing quiz results and analytics
- Receiving notifications

2.3 User Classes and Characteristics

- **Admin**: Manages the application, users, and quizzes.
- **Registered User**: Creates and takes quizzes.
- **Guest User**: Takes quizzes without registration (limited functionality).

2.4 Operating Environment

- **Frontend**: ReactJS, HTML, CSS
- **Backend**: Node.js, Express.js
- **Database**: MongoDB
- **Hosting**: Cloud-based hosting (e.g., AWS, Heroku)

2.5 Design and Implementation Constraints

- Must support multiple devices (responsive design)
- Secure authentication and data protection
- Scalable to handle high traffic

2.6 Assumptions and Dependencies

- Users have access to the internet.
- Users have basic knowledge of operating web applications.

3. System Architecture

3.1 Component Diagram

3.2 Deployment Diagram

4. Modules Description

4.1 User Management

- **Registration**: Users can register by providing necessary details.
- **Login/Logout**: Secure authentication for users to log in and out.
- **Profile Management**: Users can update their profile information.

4.2 Quiz Management

- **Create Quiz**: Users can create quizzes by adding questions and answers.
- **Edit Quiz**: Users can edit existing quizzes.
- **Delete Quiz**: Users can delete their quizzes.

4.3 Quiz Taking

- **Start Quiz**: Users can start a quiz.
- **Answer Questions**: Users can answer quiz questions.
- **Submit Quiz**: Users can submit the quiz to get results.

4.4 Results and Analytics

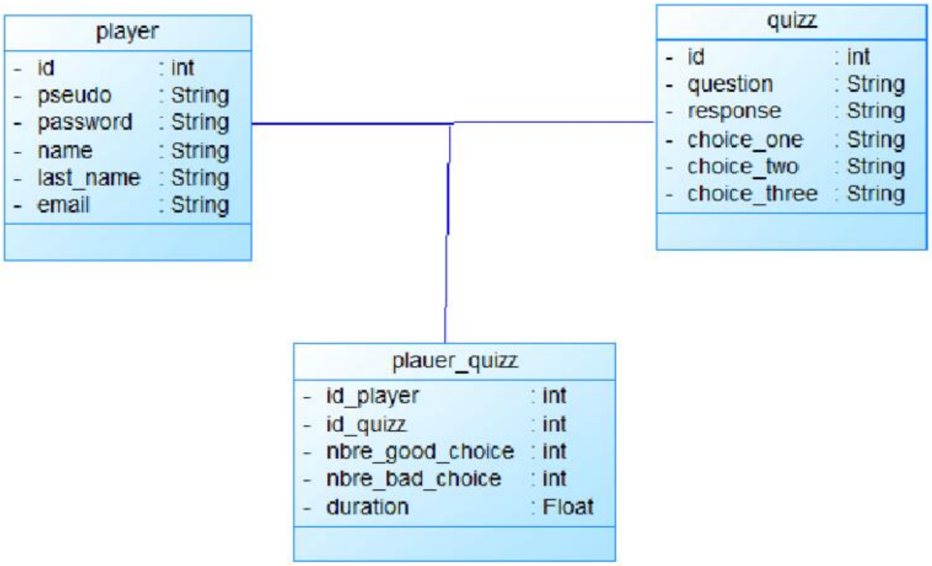
- **View Results**: Users can view their quiz results.
- **Analytics Dashboard**: Users can see their performance analytics.

4.5 Notifications

- **Email Notifications**: Users receive email notifications for important updates.
- **In-App Notifications**: Users receive notifications within the app.

5. Database Design

5.1 Schema Diagram



player	
- id	: int
- pseudo	: String
- password	: String
- name	: String
- last_name	: String
- email	: String

quizz	
- id	: int
- question	: String
- response	: String
- choice_one	: String
- choice_two	: String
- choice_three	: String

plauer_quizz	
- id_player	: int
- id_quizz	: int
- nbre_good_choice	: int
- nbre_bad_choice	: int
- duration	: Float

5.2 Table Definitions

- **Users**: Stores user information.
- **Quizzes**: Stores quiz information.
- **Questions**: Stores quiz questions.
- **Results**: Stores quiz results.

6. Interface Design

6.1 User Interface

- **Login Page**: Allows users to log in.
- **Dashboard**: Main user interface after login.
- **Quiz Creation Page**: Interface for creating quizzes.
- **Quiz Taking Page**: Interface for taking quizzes.
- **Results Page**: Interface for viewing quiz results.

6.2 API Interface

- **User API**: Manages user-related operations.
- **Quiz API**: Manages quiz-related operations.
- **Results API**: Manages results-related operations.

7. Security

7.1 Authentication and Authorization

- **JWT Tokens**: Used for secure authentication.
- **Role-Based Access Control**: Different access levels for admin and users.

7.2 Data Protection

- **Encryption**: Encrypt sensitive data in the database.
- **Data Backup**: Regular data backups to prevent data loss.