**Quiz Application Documentation**

**1. Project Overview**

The Quiz Application is a simple web-based system that allows users to:

* View their quiz performance on a dashboard.
* Take a quiz with randomly selected questions.
* Submit answers and track their performance.
* View the overall score after completing the quiz.

The application is built using:

* **Java 21**
* **Spring Boot 3.4.0**
* **H2 In-Memory Database**
* **RESTful APIs**
* **Postman** for testing the APIs

**2. Project Setup Instructions**

**2.1 Prerequisites**

Before you set up and run the project, ensure you have the following installed on your system:

* **Java Development Kit (JDK) 21** or higher.
* **Maven** for managing dependencies.
* **Postman** (for API testing).

**2.2 Cloning the Repository**

Clone the project from your version control system (e.g., GitHub):

git clone <repository-url>

cd QuizApp

**2.3 Building the Project**

To build the project, use the following Maven command:

mvn clean install or **Just Load the dependencies manually by clicking load button**

**2.4 Running the Application**

Start the application using:

mvn spring-boot:run or **YOU CAN JUST CLICK ON RUN BUTTON**

The application will start on **http://localhost:8080**.

**3. API Documentation**

**3.1 API Overview**

| **Method** | **Endpoint** | **Description** |
| --- | --- | --- |
| GET | /quiz/dashboard/{userId} | View user performance dashboard. |
| POST | /quiz/take/{userId} | Get a random quiz question. |
| POST | /quiz/submit/{userId} | Submit an answer to a question. |
| POST | /quiz/end/{userId} | End the quiz and view final results. |

**3.2 API Endpoints**

**3.2.1 View User Performance (Dashboard)**

* **Endpoint**: GET /quiz/dashboard/{userId}
* **Description**: Retrieves the user's quiz performance, including total questions attempted, correct answers, and overall score percentage.
* **Request Example**:
* GET http://localhost:8080/quiz/dashboard/1
* **Response Example**:
* {
* "id": 1,
* "username": "user1",
* "totalQuestionsAttempted": 5,
* "correctAnswers": 3,
* "scorePercentage": 60.0
* }

**3.2.2 Take a Quiz**

* **Endpoint**: POST /quiz/take/{userId}
* **Description**: Returns a random quiz question for the user.
* **Request Example**:
* POST http://localhost:8080/quiz/take/1
* **Response Example**:
* {
* "id": 1,
* "questionText": "What is 2 + 2?",
* "options": ["3", "4", "5", "6"]
* }

**3.2.3 Submit an Answer**

* **Endpoint**: POST /quiz/submit/{userId}
* **Description**: Submits the user's answer to a quiz question and updates their performance.
* **Request Example**:
* POST http://localhost:8080/quiz/submit/1
* **Request Body**:
* {
* "questionId": 1,
* "selectedAnswer": "4"
* }
* **Response Example** (Correct Answer):
* "Correct Answer"
* **Response Example** (Incorrect Answer):
* "Incorrect Answer"

**3.2.4 End the Quiz**

* **Endpoint**: POST /quiz/end/{userId}
* **Description**: Ends the quiz for the user and returns their final performance.
* **Request Example**:
* POST http://localhost:8080/quiz/end/1
* **Response Example**:
* {
* "id": 1,
* "username": "user1",
* "totalQuestionsAttempted": 5,
* "correctAnswers": 3,
* "scorePercentage": 60.0
* }

**4. Database Configuration**

**4.1 Default Database**

The project uses an **H2 in-memory database**, which is automatically configured and starts with default data (users and questions) on application startup.

* **H2 Console URL**: <http://localhost:8080/h2-console>
* **JDBC URL**: jdbc:h2:mem:testdb
* **Username**: sa
* **Password**:password

**4.2 Preloaded Data**

* **Users**:

| **User ID** | **Username** |
| --- | --- |
| 1 | user1 |
| 2 | user2 |
| 3 | user3 |
| 4 | user4 |
| 5 | user5 |

* **Questions**: 10 sample questions are preloaded into the database.

**5. Error Handling**

* **User Not Found**:
* {
* "error": "User with ID 15 not found"
* }
* **Question Not Found**:
* {
* "error": "Question not found"
* }
* **No Questions Available**:
* {
* "error": "No questions available"
* }

**6. Testing the APIs**

You can use **Postman** or any REST client to test the APIs.

1. **Open Postman**.
2. **Create a new request**.
3. Enter the appropriate URL and method.
4. Provide the necessary request body (for POST requests).
5. Send the request and view the response.