# 

# **Backend Task - AI Quizzer**

## Task

Build and host a microservice for an **AI-powered Quiz Application** with authentication, quiz management, AI-based evaluation, and score tracking.

## Core functionalities (Mandatory)

**Authentication**

* Mock authentication service accepting any username/password.
* Returns a signed JSON Web Token ([**JWT token**](https://jwt.io/)**)** for subsequent requests.
* Token validation for all quiz endpoints.

**Quiz Management REST API endpoints**

* Generates new quiz with data provided in the request body (including grade level) **(AI)**
* [Sample payload](https://jsonkeeper.com/b/CQ9E)
* Submits quiz answers and returns evaluated score using **(AI)**
* [Sample payload](https://jsonkeeper.com/b/84KD)
* Retrieves quiz history and scores based on filters provided in the request
* Filter based on grade, subject, marks, completedDate, etc.
* Filter Specific date range i.e. *from=01/09/2024, to=09/09/2024*
* Allows retrying a quiz and re-evaluates scores
* Old submission should also be accessible

**AI Features (Mandatory)**

* **Hint Generation** – AI provides a hint for a question when requested.
* **Result Suggestions** – After quiz submission, AI suggests 2 improvement tips based on mistakes.
* **Adaptive Question Difficulty** – Adjusts question difficulty in real-time based on user performance in the current quiz.

#### **Database**

* Store quizzes, submissions, scores and other required fields.
* Provide SQL scripts or migration files to create the required schema**.**

**Application hosting**

* Create and deploy the docker image on a third-party hosting service (Heroku, DigitalOcean, AWS, etc).
* Submit hosted API URL and a **Postman collection** with sample calls.

## Important:

* Attach API documentation with a request which can be executed directly. You can use Postman Collection, Swagger, etc for documentation.
* Attach **README** with:
  + Setup instructions
  + AI integration details (API used, endpoints/models)
  + Any known issues
* Please refrain from adding **node\_modules** or other auto generated packages in your submission.

## Bonus functionalities: Extra points for attempting these :)

* Implement a **notification feature** to send results over email
* Integrate a caching layer mechanism (e.g., Redis) for reduced API latency while fetching quiz data
* Create a **Leaderboard API** – Display top scores for a grade/subject.

## Technical Requirements and Guidelines

* You are free to use any language (Java, Node.js, Python, etc.)
* You are free to use any RDBMS / NoSQL.
* There are no restrictions on the frameworks to use / database design schema. Choose whichever seems best suited for the task.
* You can use any AI tool to integrate and generate quiz data. (Suggested AI Tool: Groq)

## 

## 

## Things we are interested in

* Completeness of the APIs - authentication & AI Quizzer with handling of all possible edge cases
* Knowledge of REST APIs, SQL/NoSQL, JWT, and integration with AI services
* Database design quality
* Modularity & readability of code
* Hosting correctness & accessibility.

## How to submit

Please email your submission to the following address.

To: assessments@playpowerlabs.com

Subject: <Your Name> - <College shortname> Backend Task

Along with:

1. Zip of your code
2. Hosted URL (Optional) [Netlify, Heroku, etc..]
3. Screen recording of your working application (Optional) [You can use Loom to

create one and send us the link]

For any clarifications contact: hiring.support@playpowerlabs.com

## Important

We condemn plagiarism. Please maintain the dignity and originality of your work. If we suspect any attempt towards copying, we will disqualify your submission.   
**Also, Don't put your code on GitHub.  
Please DO NOT submit *node\_modules* or other auto generated packages.**

**Remember:** While bonus features are impressive, prioritize completing the core features with high-quality, well-structured code. Demonstrate your understanding of modern web development practices and your ability to create efficient, user-friendly applications. We thoroughly review each submission, examining both the overall functionality and the nuances of your code. Show us your best work!