**Quizify: AI-Powered Quiz Application**

**Group Members Details:**

16010122034 Peeth Chowdhary

16010122041 Rohit Deshpande

**Objective:**

The aim of this project is to build an interactive quiz app that personalizes quizzes to match individual user skills and preferences. The app generates quiz content dynamically through the **Gemini API**. Currently, the app’s AI functionality is focused on generating contextually relevant questions based on topics of interest. Designed as a versatile tool, this quiz platform primarily supports students with targeted question sets while enabling educators to manage quizzes and track student performance through analytics.

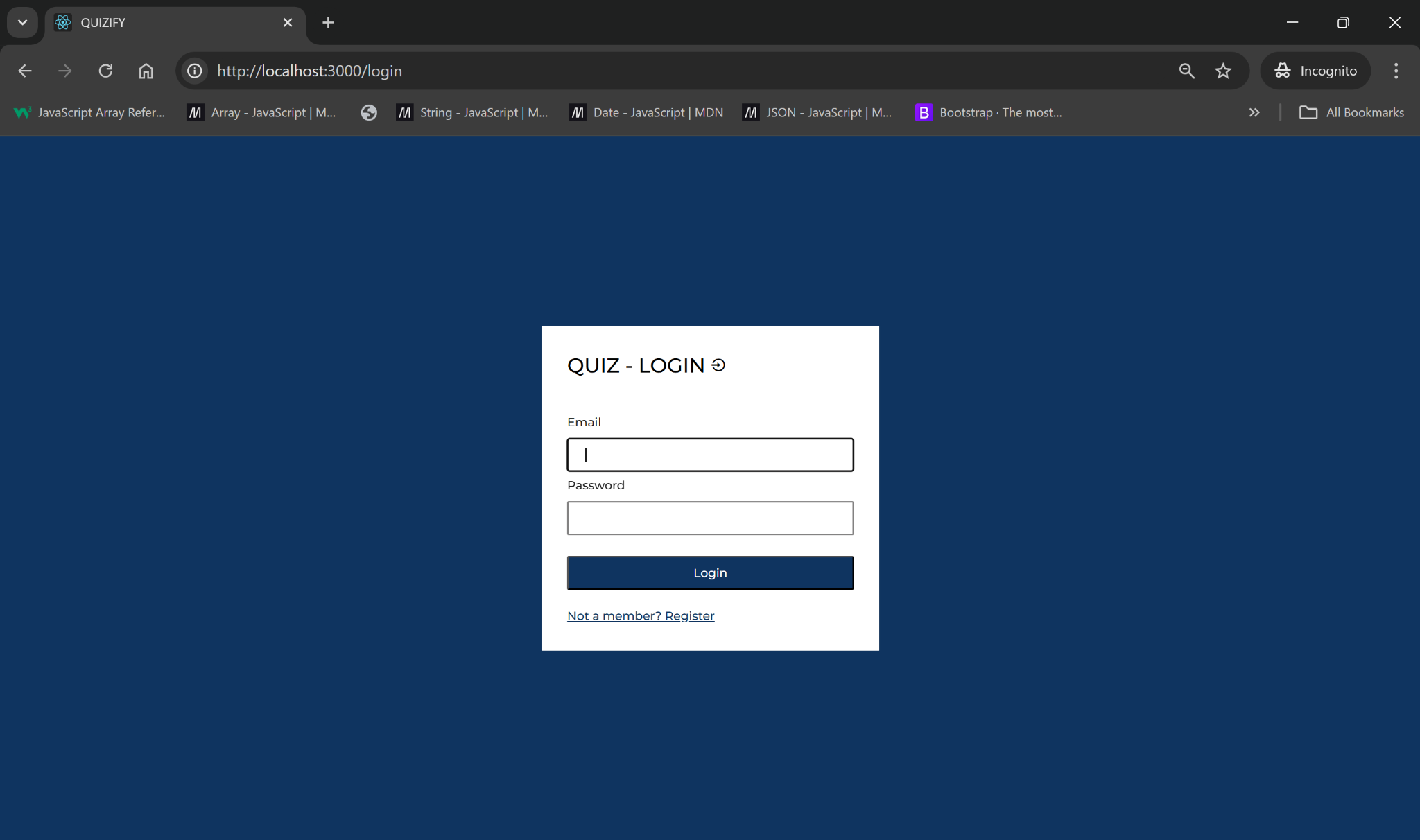
**Work distribution (functionality wise):**

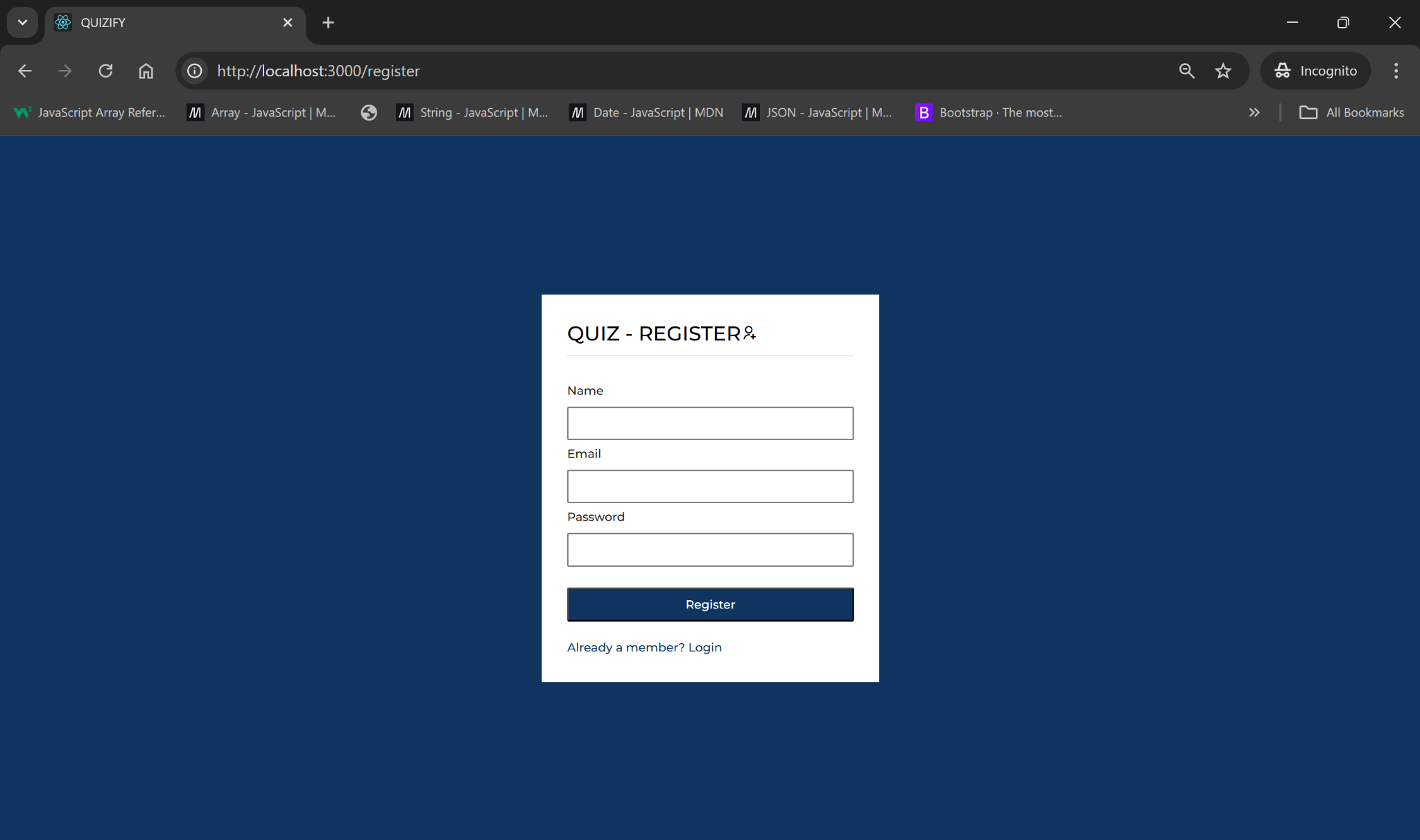
* **Peeth Chowdhary:**
* **Question Generation:** Integrated the **Gemini API** for dynamic question generation tailored to user profiles, enabling AI-powered personalization of quiz questions.
* **Backend Development:** Set up RESTful API endpoints using Express.js to handle user data and quiz-related requests.
* **Educator Features:** Developed functionality for educators to manage quizzes, monitor student results, and download performance analytics for detailed reporting.
* **Rohit Deshpande:**
* **Frontend Development:** Designed and implemented user interface components with React, focusing on an intuitive and responsive layout for the quiz-taking experience.
* **UI/UX Design:** Developed the app’s visual and interactive elements to ensure a smooth user experience.
* **Backend Integration:** Connected frontend components to backend APIs and ensured smooth data flow between the client side and MongoDB, handling data retrieval and submission effectively.

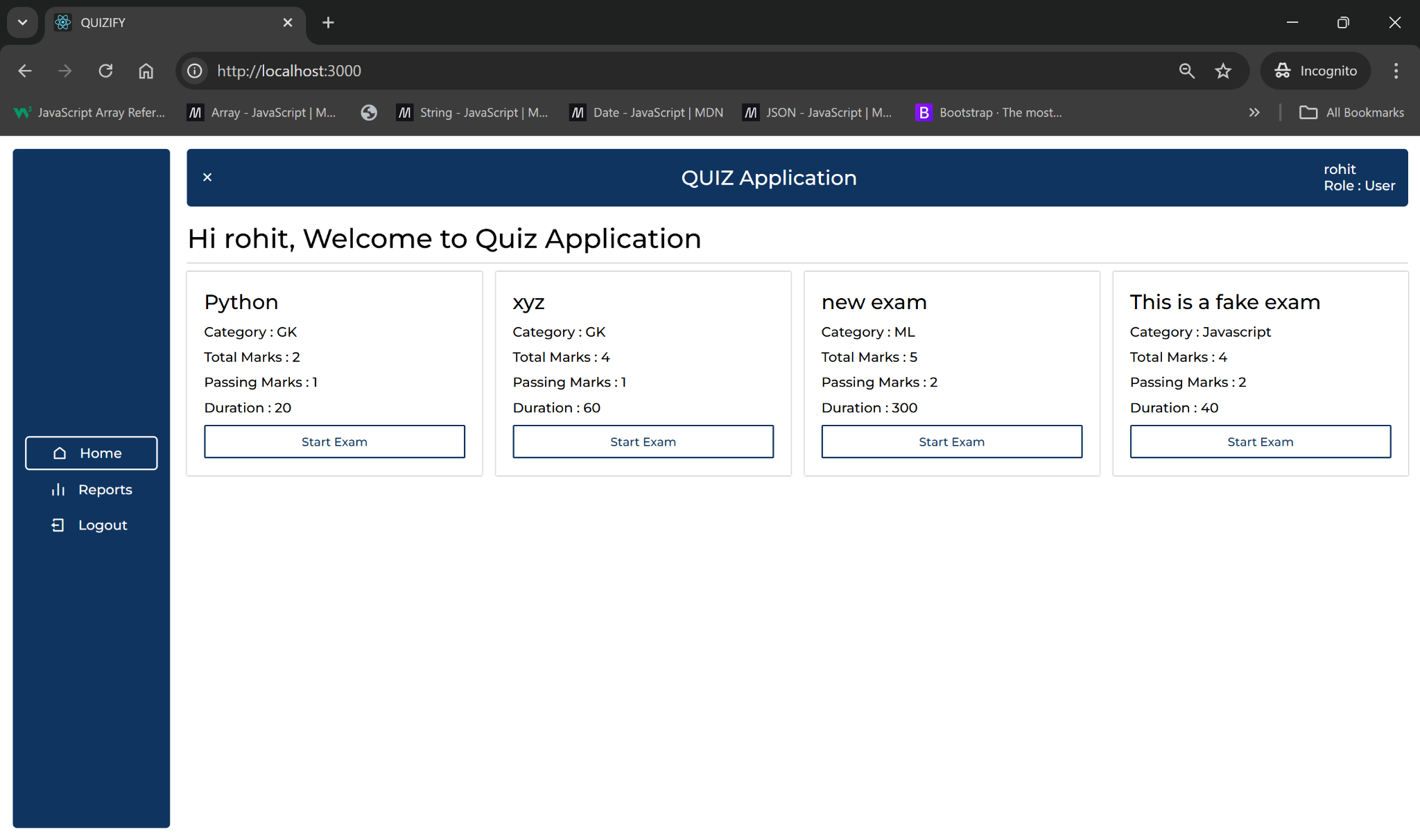
**Database:**

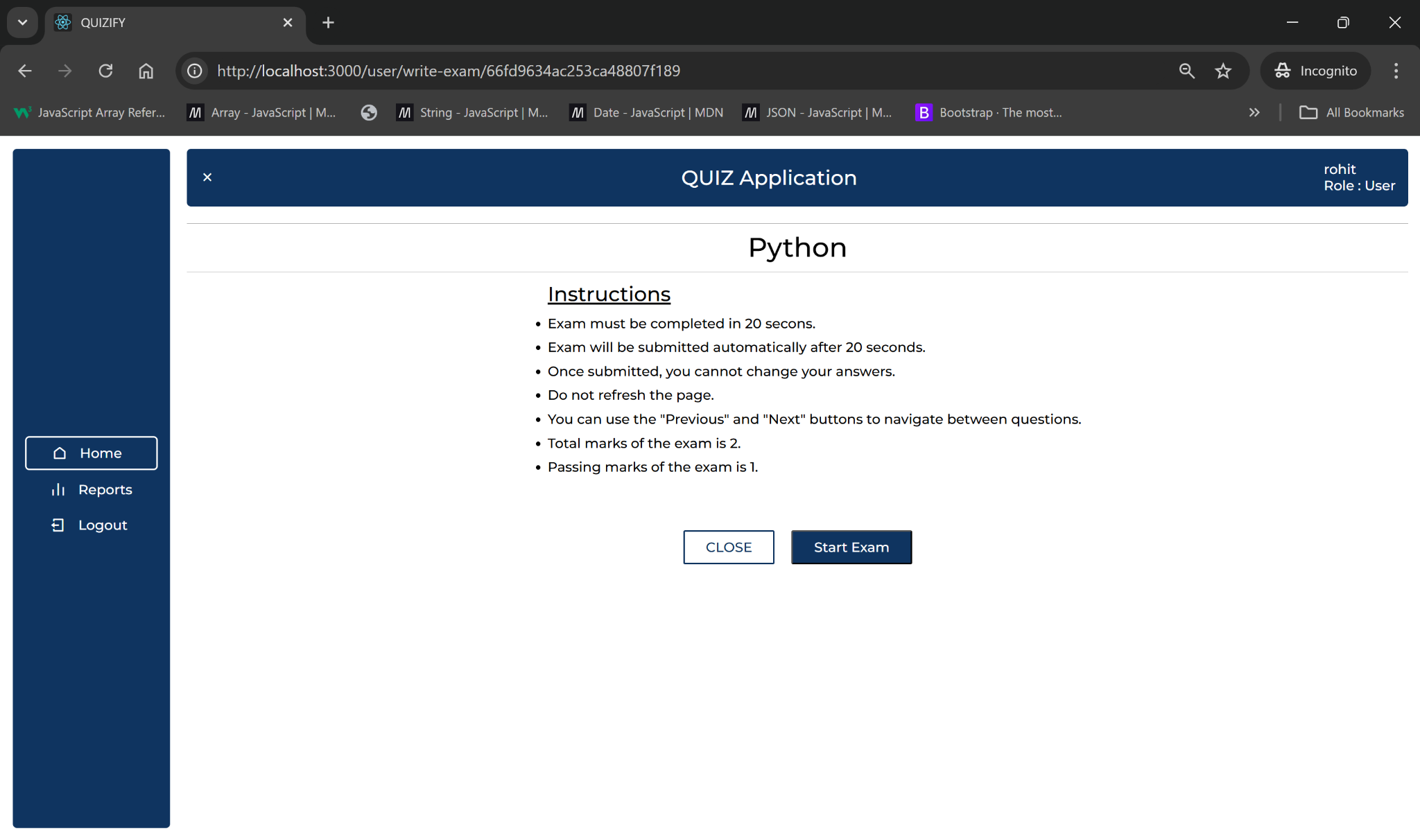
* **Database Type:** MongoDB
* **Structure:** NoSQL database with collections and schemas optimized for managing users, exams, questions, and reports, each linked to support personalized quiz generation and performance tracking.
* **Collections and Schemas:**
  + **Users:**
    - Stores essential user information, including name, email, password, and isAdmin status.
    - Schema fields:
      * name (String): User’s full name.
      * email (String): Unique email for user identification.
      * password (String): Hashed password for secure login.
      * isAdmin (Boolean): Identifies admin users, with a default value of false.
  + **Exams:**
    - Manages each exam instance, including details like duration, category, total and passing marks, and associated questions.
    - Schema fields:
      * name (String): Exam name or title.
      * duration (Number): Time limit for the exam in minutes.
      * category (String): Category or subject of the exam.
      * totalMarks (Number): Maximum achievable marks for the exam.
      * passingMarks (Number): Minimum marks required to pass.
      * questions (Array of ObjectId references): Links to associated questions in the questions collection.
  + **Questions:**
    - Contains individual questions, each with options, a correct answer, and a reference to its associated exam.
    - Schema fields:
      * name (String): The question text.
      * correctOption (String): The correct answer option.
      * options (Object): An object containing all answer options for the question.
      * exam (ObjectId reference): Links to the specific exam in which the question appears.
  + **Reports:**
    - Stores performance data for each user’s exam attempt, including the result and user feedback.
    - Schema fields:
      * user (ObjectId reference): Links to the user who took the exam.
      * exam (ObjectId reference): Links to the specific exam.
      * result (Object): An object containing the user’s score, answers, and other performance metrics.
* **Configuration:** Indexed key fields (e.g., user ID, exam ID, question ID) to speed up query response times, especially for common operations such as retrieving user reports, exams, and related questions.

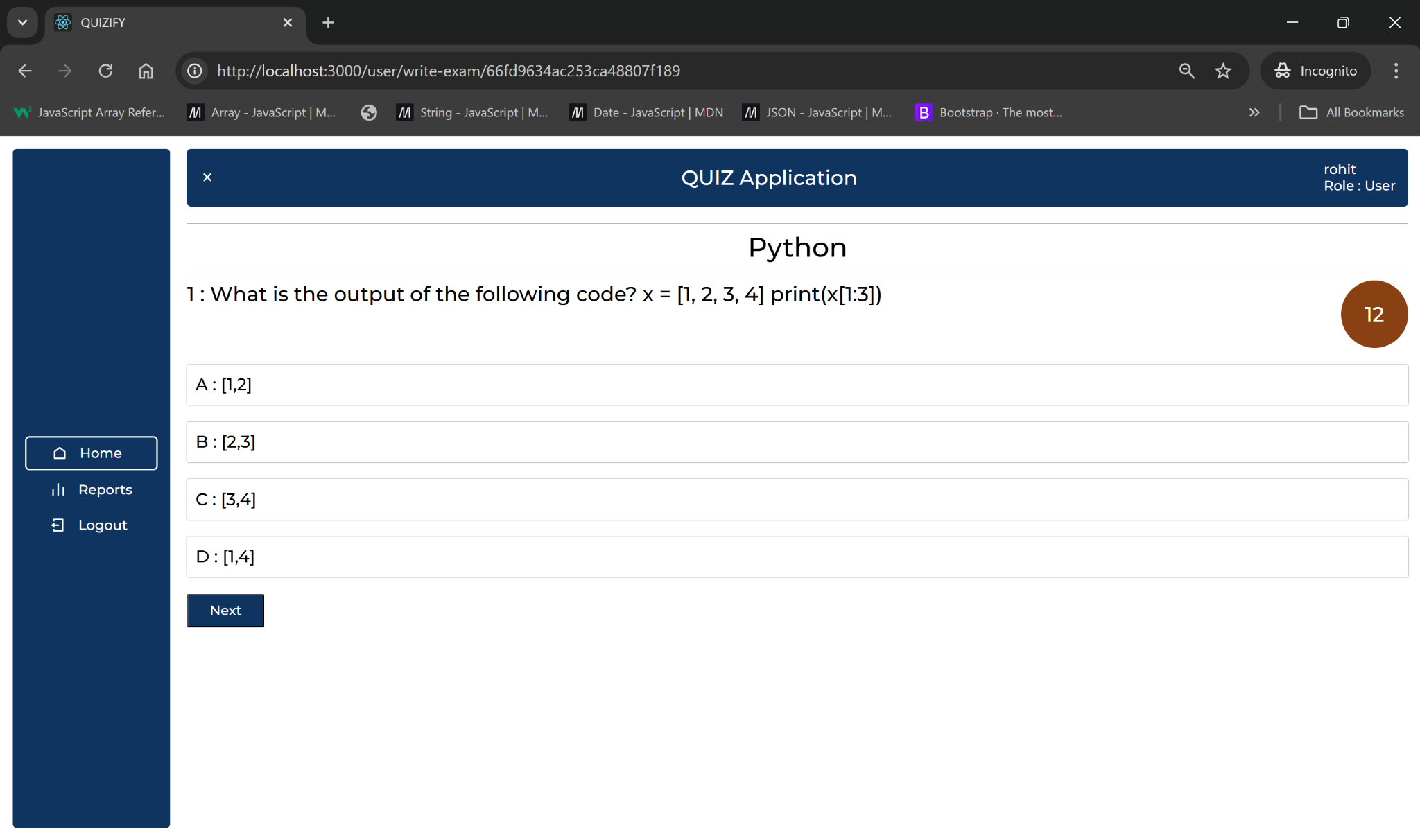
**Screenshot:**

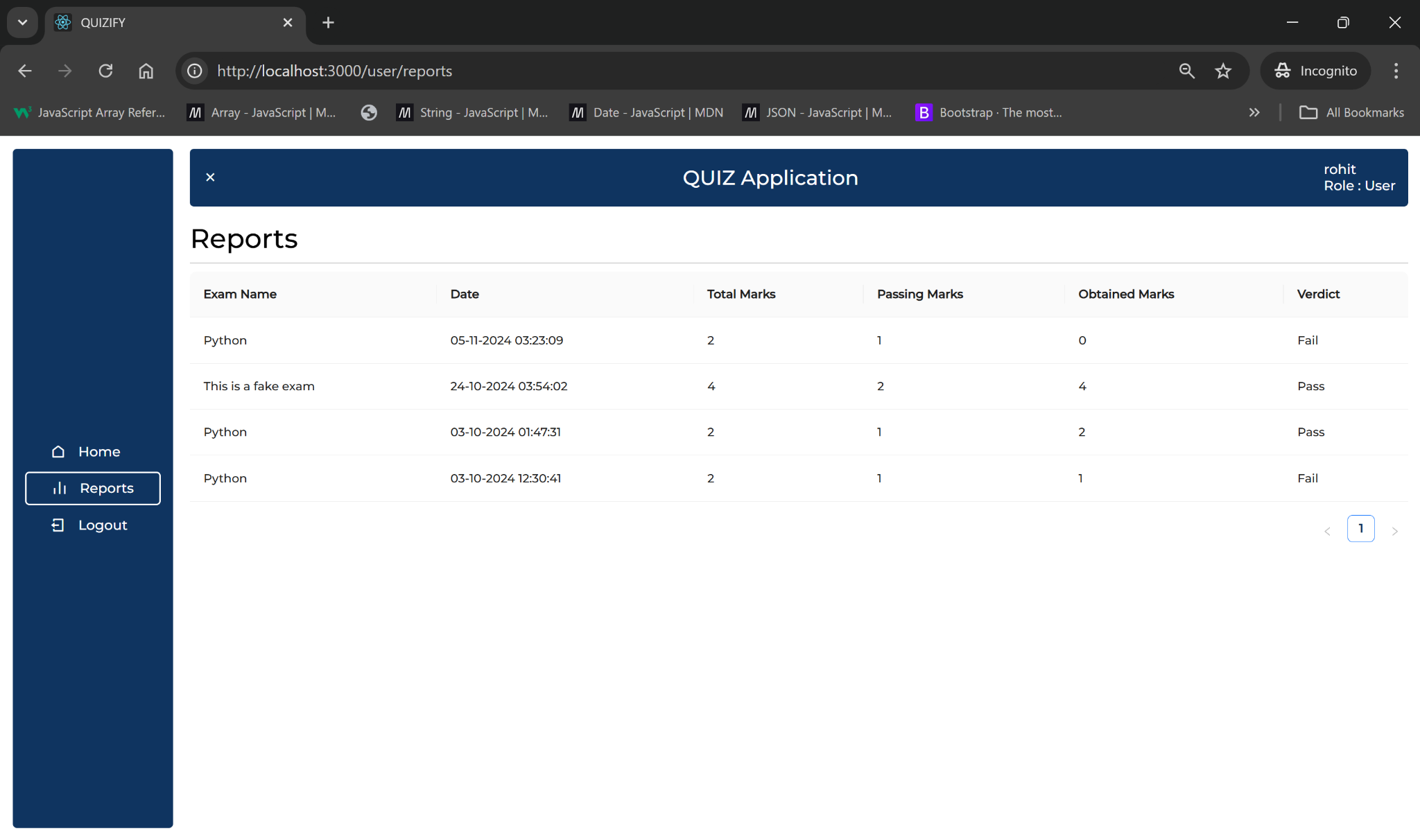
****

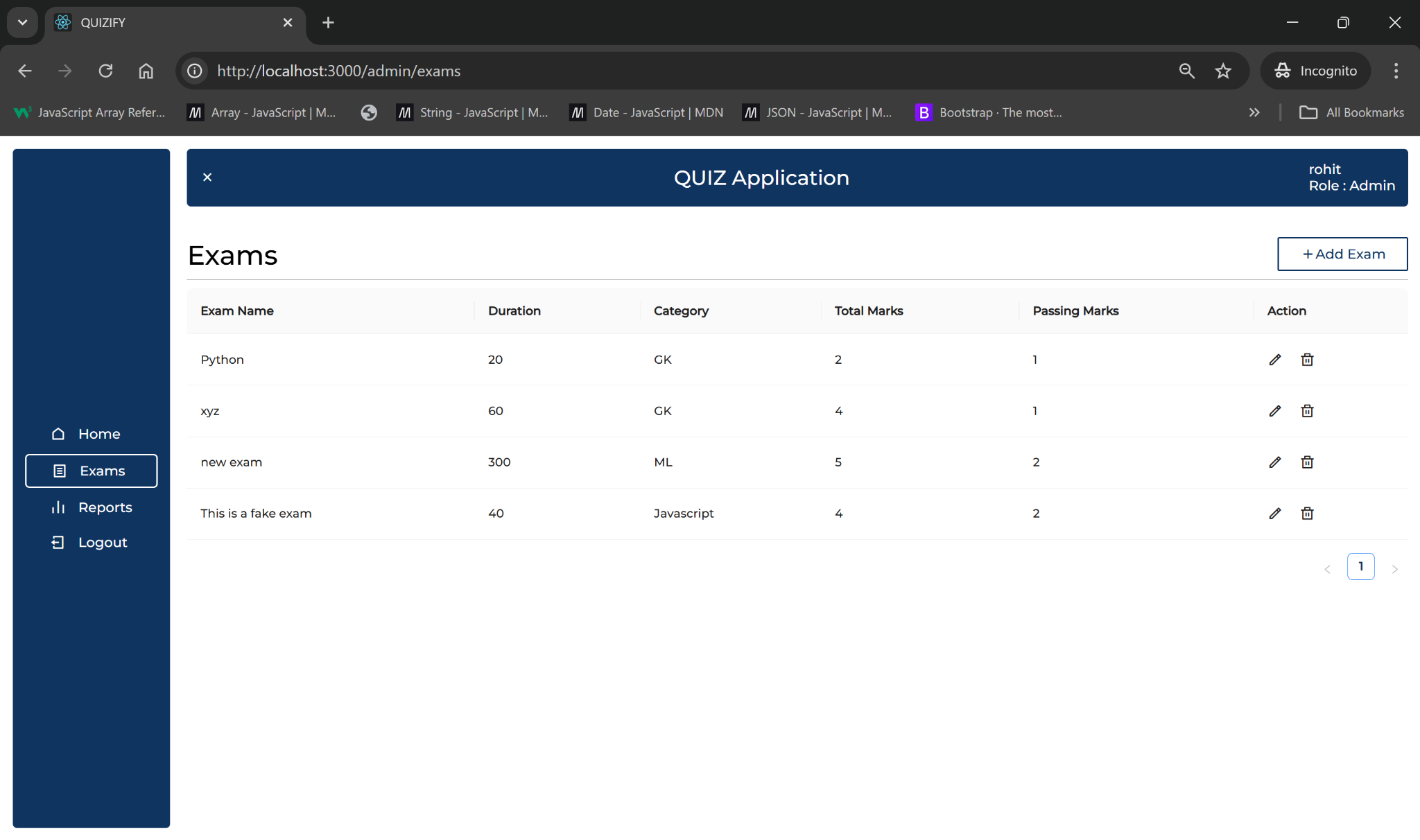
****

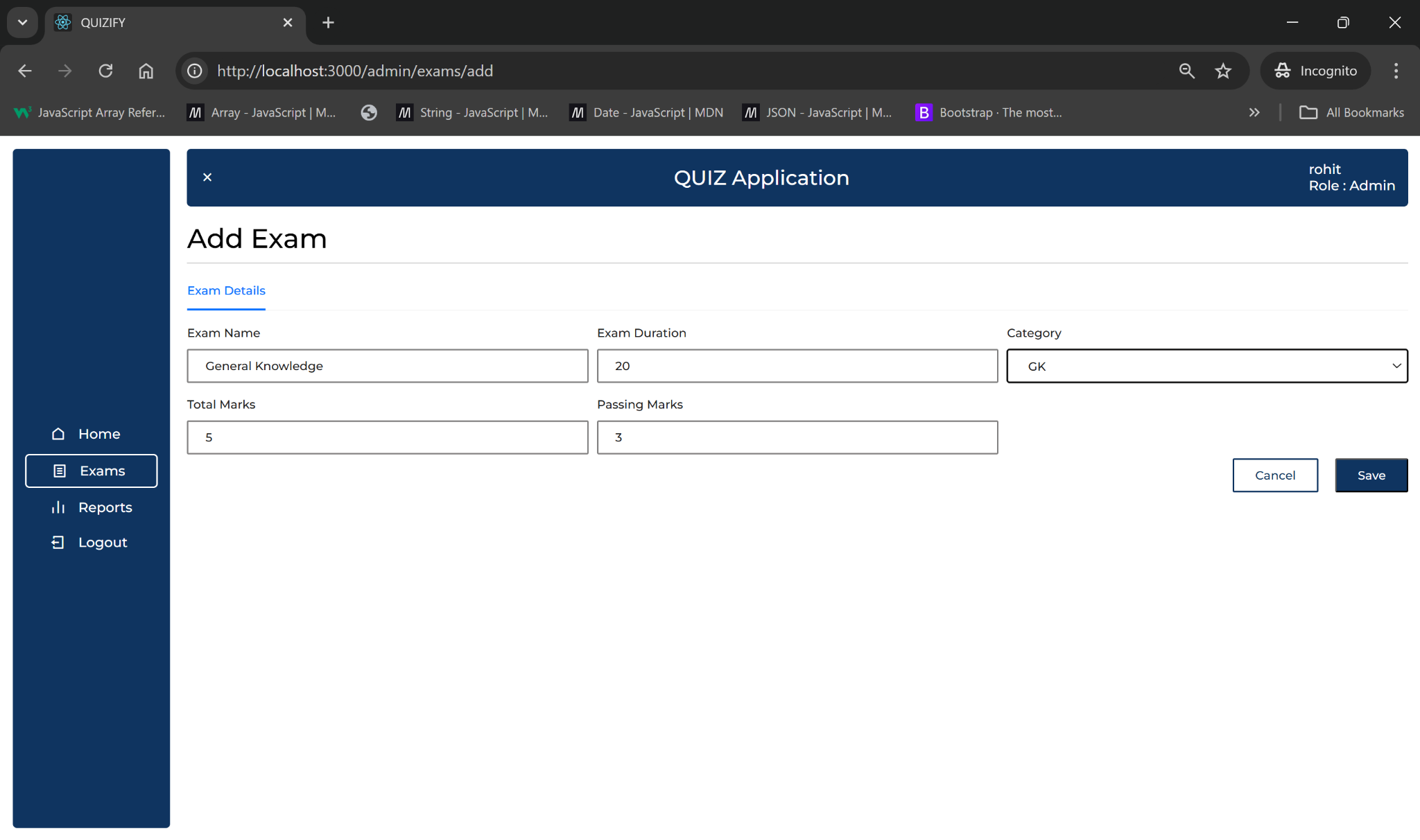
****

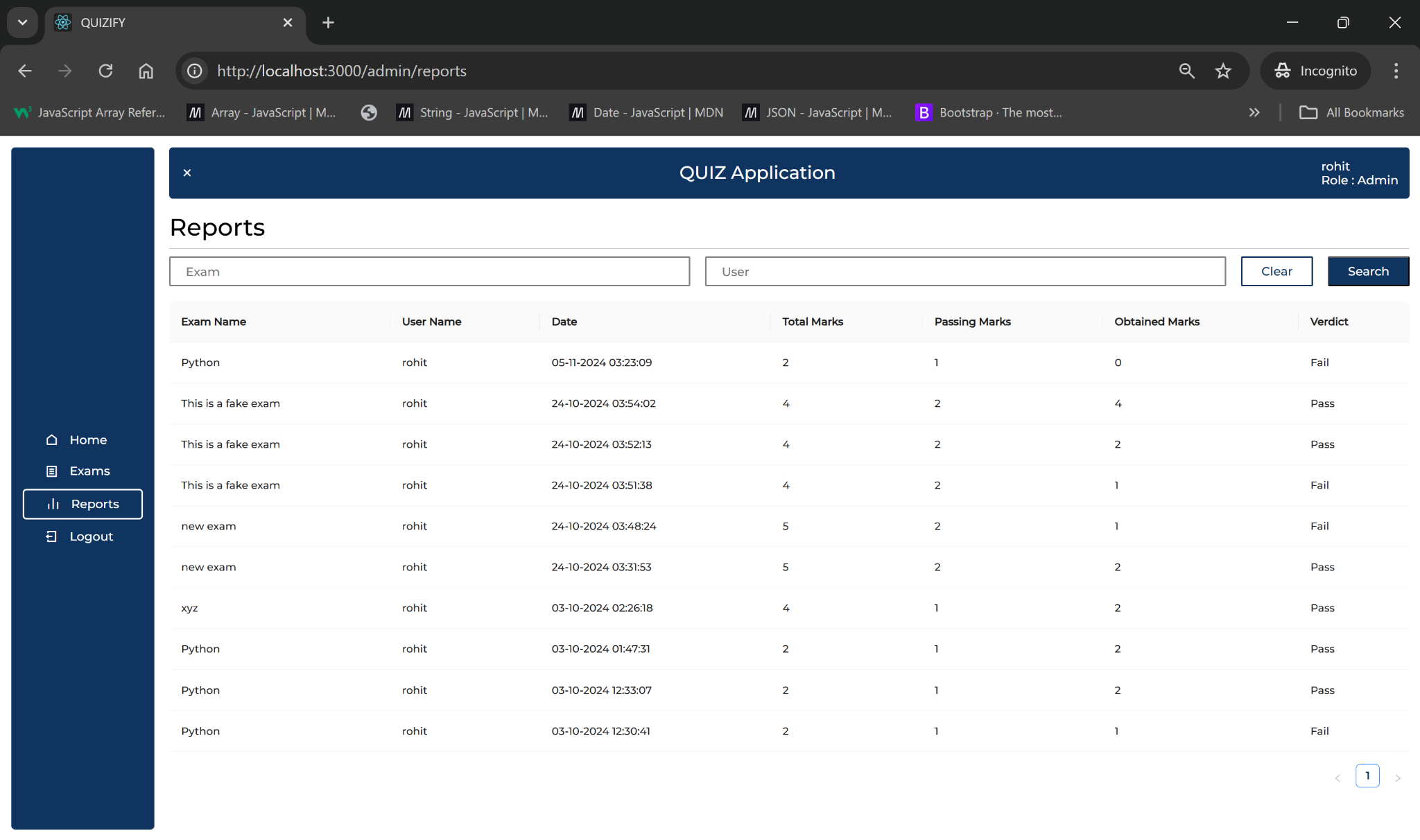
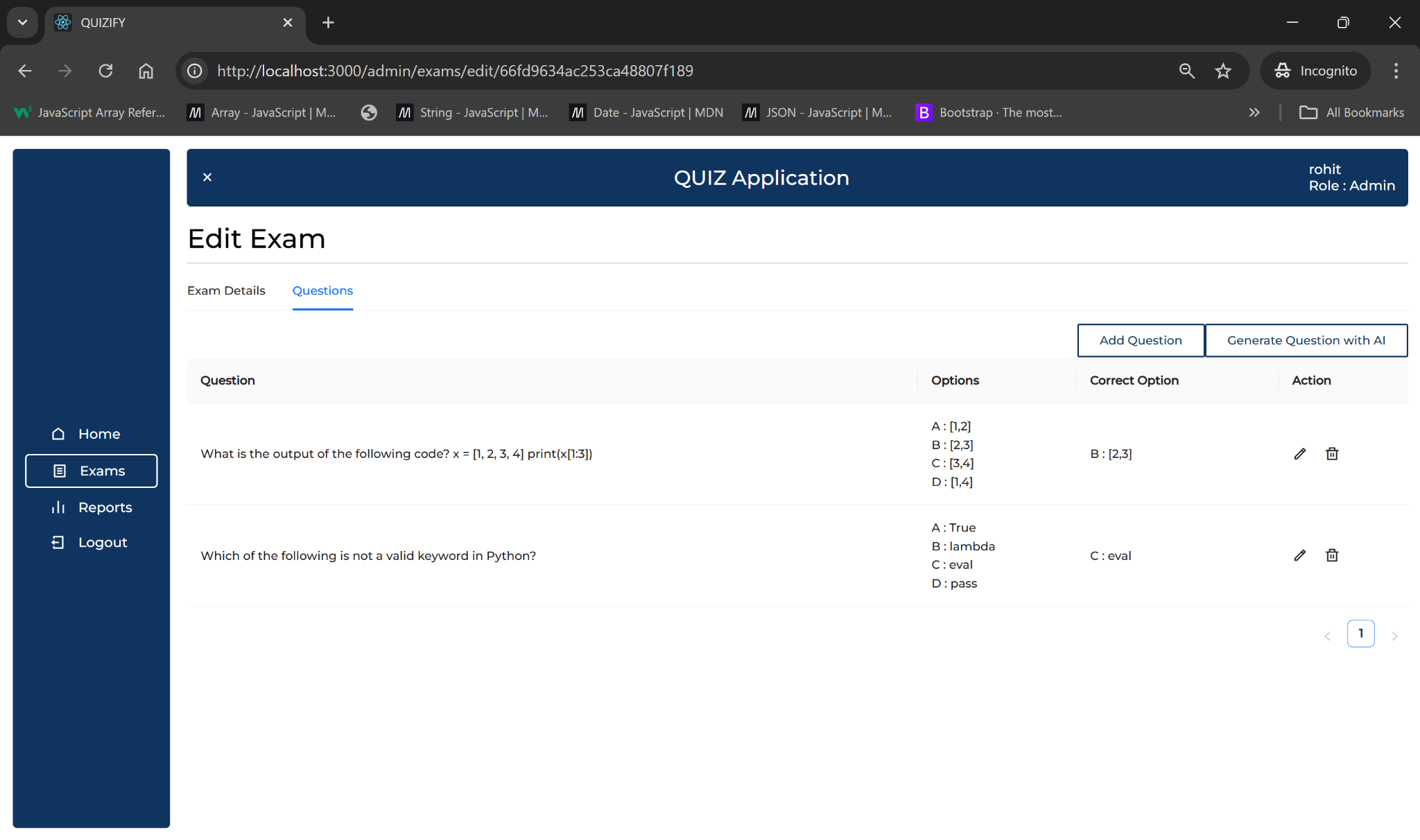
****

****

****

****

****

****

**Tech Stack Used:**

* **Frontend:** React.js, CSS, HTML, Axios for API integration
* **Backend:** Node.js, Express.js
* **Database:** MongoDB
* **AI Question Generation:** Gemini API for AI-powered, contextually relevant question generation based on user profiles and preferences.
* **Additional Libraries/Tools:**
  + **Mongoose:** MongoDB object modeling for schema-based solutions.
  + **JWT, bcrypt:** For secure user authentication.
  + **Axios:** Used to handle API requests and responses between frontend and backend.

**Features Implemented:**

1. **User Registration and Profile Creation:**
   * New users register by creating profiles that store their basic information. All sensitive data, including passwords, is securely hashed using the bcrypt library to ensure data protection and privacy.
   * User data, preferences, and quiz history are securely stored in MongoDB, allowing for a personalized experience each time they log in.
   * Profiles are accessible to users for review, and data is updated, ensuring the quiz generation remains aligned with users’ evolving interests.
2. **AI-Driven Question Generation with Gemini API:**
   * The app uses the **Gemini API** to dynamically create questions based on user profiles, skills, and selected topics.
   * Topics currently supported include subjects like JavaScript, Python, Business Studies etc, enabling users to test and improve specific skill sets.
   * Each generated question is unique to the user’s selected topic and preferences, promoting personalized learning experiences.
3. **Quiz Storage and Retrieval:**
   * Every quiz a user takes, along with their responses, is stored in MongoDB, building a record of their performance and progress.
   * Users have access to past quizzes, which they can review for continuous learning, allowing them to monitor improvement over time.
   * Educators can create and assign quizzes to students, view completed quizzes, and track cumulative results to assess performance at both individual and group levels.
   * Data on quizzes and user responses are efficiently indexed to allow fast access and retrieval, even as the database grows.
4. **Analytics and Performance Tracking for Educators:**
   * Educators have access to an analytics dashboard that provides detailed insights into each student’s performance, including metrics such as accuracy rates, average scores, and progress over time.
   * Performance data is visualized in easily interpretable formats, helping educators quickly identify areas where students excel or may need additional support.

**Technical Issues Encountered and Solved:**

* **Issue 1:** **Handling Gemini API Request Latency**  
  *Problem:* Initial API calls to the Gemini AI for question generation caused latency, impacting user experience during quiz generation.

*Solution:* Implemented caching to store frequently requested questions temporarily and pre-loaded common question topics to reduce waiting time for users. Added error handling for API timeouts to retry calls efficiently.

* **Issue 2:** **Optimizing Quiz Storage in MongoDB**  
  *Problem:* Storing large sets of quiz data led to slow retrieval times, especially as the user base grew.

*Solution:* Streamlined MongoDB schema design by indexing crucial fields and using embedded documents for quick access to user performance data. Additionally, applied data pruning techniques to archive older quizzes.

* **Issue 3:** **Synchronizing Frontend and Backend Data Flows**  
  *Problem:* Inconsistencies between the frontend UI and backend data formats led to errors in user data display and quiz loading.

*Solution:* Standardized API response formats across all endpoints and implemented Axios interceptors for consistent error handling. This ensured that frontend components received expected data structures.

**The Future Scope of the Project:**

1. **Expand Question Generation Capabilities:**
   * Broaden the subject range and question types generated by the **Gemini API** to cover additional topics, enhancing quiz diversity and catering to various learning requirements.
2. **User Feedback and Question Quality Improvement:**
   * Implement a more advanced feedback system, where user ratings on question relevance, difficulty, and accuracy are continuously analyzed to enhance the quality of future quizzes.
3. **Group Collaboration Features:**
   * Develop a real-time collaboration mode that allows multiple users to participate in quizzes together, promoting interactive and collaborative learning environments.
4. **Personalized Learning Paths and Topic Recommendations:**
   * Create tailored learning paths based on individual quiz performance, identifying user strengths and weaknesses to recommend specific topics or quizzes for improvement.
   * Include AI-driven topic recommendations that guide users toward areas requiring additional practice or advanced study, based on scores and feedback.
5. **Mobile App Development:**
   * Design a mobile application for the quiz platform to improve accessibility and reach a wider user base, particularly those who prefer mobile devices for learning.
6. **Automated Recommendations and Feedback Analysis:**
   * Integrate AI to analyze quiz results and user feedback in-depth, generating personalized recommendations for improvement and suggesting supplementary learning materials.
7. **Real-Time Adaptive Quizzing:**
   * Introduce adaptive quizzing, where question difficulty adjusts dynamically based on user responses, creating a more challenging and engaging experience that matches the user’s skill level.

**Github Link:** <https://github.com/irohitdeshpande/mern-quiz>

**README.md file:** [**Link**](https://github.com/irohitdeshpande/mern-quiz/blob/main/README.md)