PROJECT REPORT

MindWars AI

"An AI Generated Quiz Contest Application"

Contributors
-
Arun Chandra
Ashish Sah

TABLE OF CONTENTS

1. ABSTRACT

MindWars AI is a dynamic and intelligent quiz battle platform that harnesses the power of **Generative AI** to deliver personalized and engaging quiz contests. Unlike traditional quiz systems that rely on static question banks, MindWars AI generates quizzes in real-time based on user-defined topics and difficulty levels, offering a highly customized learning experience.

The platform allows users to create or join quiz battles, attempt time-bound questions, and receive immediate performance scores. Beyond basic assessment, MindWars AI integrates advanced AI models (Google Gemini and Educhain) to provide **personalized post-quiz feedback** and a **learning path with curated educational resources**, making it not only a test but also a tool for guided improvement.

Built using a modern tech stack—**React.js** for the frontend, **Flask** for the backend, and **MongoDB** for storage—MindWars AI also features JWT-based authentication, leaderboard ranking, and AI-powered quiz generation and feedback. This project demonstrates the potential of merging **AI with EdTech**, providing a scalable and interactive solution for students, learners, and quiz enthusiasts.

The application is fully web-based, responsive across devices, and serves as a robust example of how AI can enhance self-paced learning through interactive design and intelligent feedback mechanisms.

2. Introduction

- **Problem Statement**
  Traditional quiz apps lack personalization, reuse static question banks, and do not offer adaptive learning feedback.
- **Objective**
  To create an AI-integrated quiz platform that generates contests dynamically and provides users with instant results and personalized learning suggestions.
- **Scope**
  Single-player quiz contests, AI-generated questions, automatic score evaluation, real-time leaderboard, personalized feedback. Hosted as a full-stack web application.

3. Requirements

Software:

- Frontend: React.js, Tailwind CSS
- Backend: Flask (Python), JWT
- AI: Educhain (Python Library) + Google Gemini
- Database: MongoDB
- Deployment: Vercel (frontend), Render(backend)

Hardware:

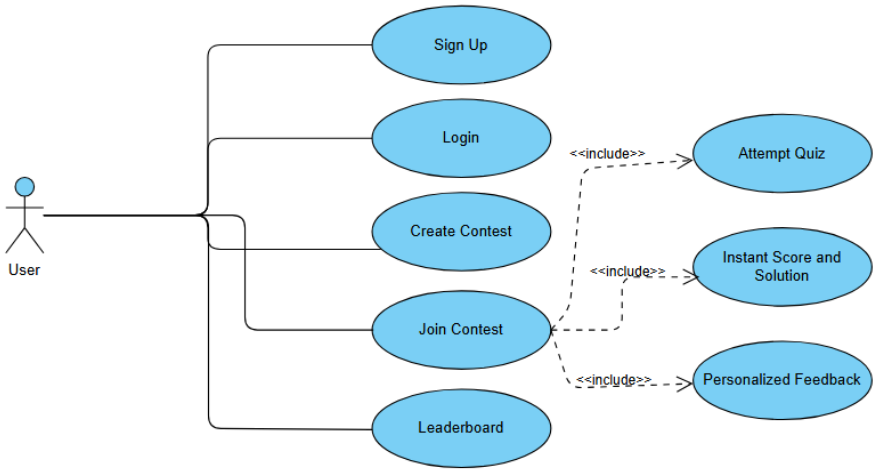
- Any modern browser and internet-enabled device

4. Modules Description

Sno.	Module	Description
1.	User Authentication	JWT-based signup/login using bcrypt and token headers
2.	Create Battle	Users create a quiz battle (name, desc, difficulty, time)
3.	Join Battle	Users can join a battle by attempting quizzes created by themselves or others
4.	AI Question Generator	Uses Gemini AI via Educhain to generate topic-based quiz questions
5.	Scoreboard Generation	Submitting the quiz, User gets the instant score evaluation
6.	AI Personalized Feedback	Generates a post-quiz learning path using LLM
7.	Leaderboard	Shows ranked participants for each quiz with score and time taken

5. Use Cases

Use Case Diagram:



Use Case	Description
Sign Up	A new user registers by providing a username, email, and password. The password is securely hashed and stored in the database.
Login	An existing user provides login credentials (email or username + password). If valid, the system generates a JWT token to authenticate future requests.
Create Quiz	The user provides a topic and description; the system uses Generative AI to generate quiz questions. The quiz is then saved with metadata like time limit, number of questions, and creator name.
Attempt Quiz	A user selects or joins a quiz using a unique ID. The system fetches questions, starts the timer, and records user responses. Navigation and timer enforcement ensure fairness.
Submit Quiz	Upon submission, the system calculates the score, measures time taken, and sends the data to the backend. AI generates personalized feedback and stores it alongside the results.
View Leaderboard	Displays a sorted list of all users who attempted the quiz, ranked by score and time taken. This fosters a competitive environment and encourages reattempts for better ranking.

---

6. Database Design

a. Users

Field Name	Data Type	Description
username	varchar	Unique identifier for the user
email	varchar	User's email address
password	varchar	User's password

b. Quizzes

Field Name	Data Type	Description
quiz_id	varchar (Primary Key)	Unique identifier for each quiz
quiz_name	text	Title of the quiz
quiz_description	text	Short description of the quiz

num_of_questions	int	Total number of questions
time_limit	int	Time limit in minutes
created_at	timestamp	Date and time of creation
creator_username	varchar (Foreign Key)	Username of the quiz creator
deadline	timestamp	Time after which the quiz expires

c. Questions

Field Name	Data Type	Description
quiz_id	varchar	Identifier of the associated quiz
question	text	Question text
answer	varchar	Correct answer
explanation	text	Explanation for the answer
option_1	varchar	First option
option_2	varchar	Second option
option_3	varchar	Third option
option_4	varchar	Fourth option

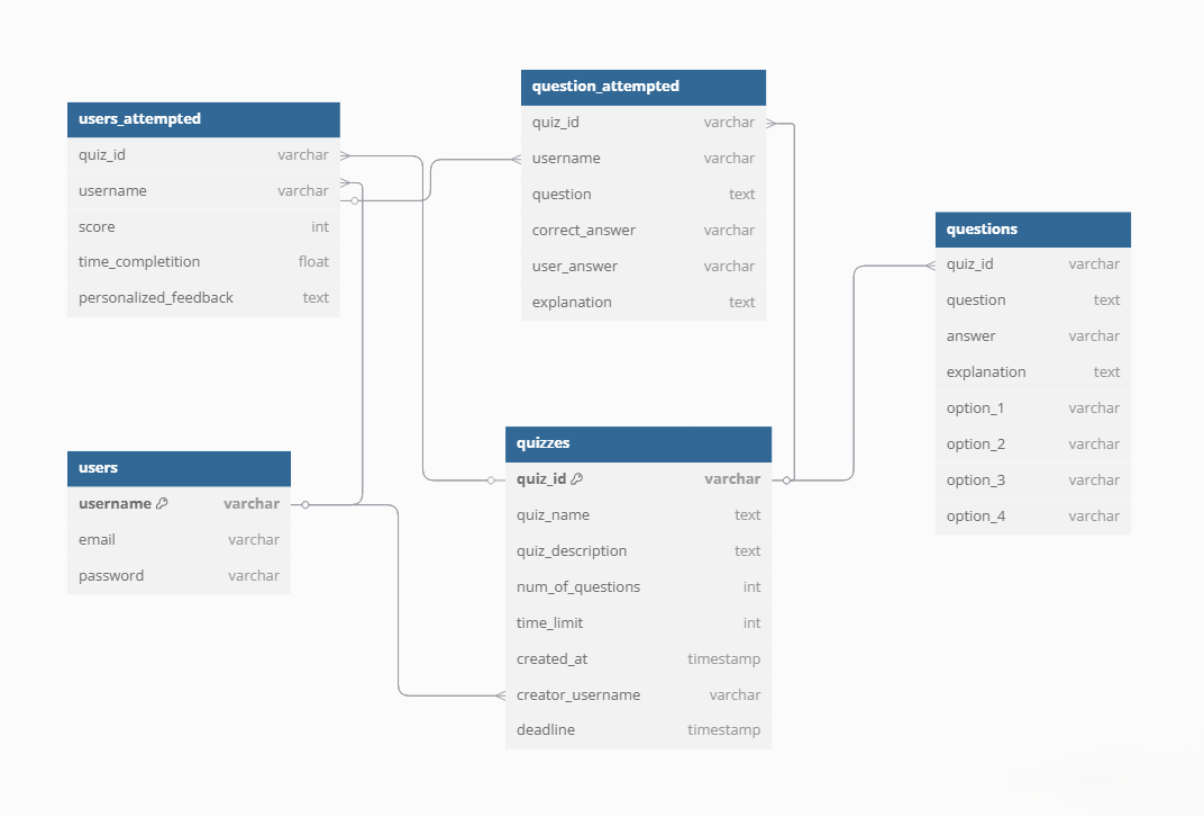
d. Users Attempted

Field Name	Data Type	Description
quiz_id	varchar	ID of the attempted quiz
username	varchar	Username of the participant
score	Int	Score achieved by the user
time_completition	float	Time taken to complete the quiz
personalized_feedback	text	AI-generated feedback after submission

e. Question Attempted

Field Name	Data Type	Description
quiz_id	varchar	Quiz ID
username	varchar	Username of the user
question	text	The question attempted
correct_answer	varchar	Correct answer to the question
user_answer	varchar	Answer selected by the user
explanation	text	Explanation of the correct answer

Logical Database Diagram:



7. Edge Cases Handling

Edge Case	Description	Handling Strategy
Unauthorized Access to Start Contest Page	User tries to access the Start Contest page without signing in.	Restrict navigation and display a toast notification prompting the user to sign in.
Invalid Login Credentials	User enters wrong email/username or incorrect password.	Return 401 Unauthorized with a clear error message like “Invalid credentials.”
JWT Token Missing or Expired	Protected route accessed without a token or with an expired token.	Return 401 Unauthorized; redirect to login and show appropriate message.
Quiz Not Found	User tries to access a deleted or non-existent quiz.	Return 404 Quiz not found; display a friendly error message on the frontend.
Duplicate Quiz Submission	User attempts to submit the same quiz multiple times.	Check if already submitted; block re-submission and notify the user.
AI API Timeout or Failure	Feedback generation fails due to Gemini/Educhain timeout or error.	Show fallback message like “Feedback unavailable”; log the error for debugging.

Empty Answers or Skipped Questions	User submits the quiz without attempting all questions.	Score only attempted questions; skipped questions marked in the review section.
Accessing Expired Quiz	User tries to start a quiz after its deadline has passed.	Block access and show “This quiz has expired” message.
Same Username Attempting Same Quiz Twice	A user re-attempts a quiz that was already submitted.	Prevent duplicate attempts by checking users_attempted in database.
Unexpected Server Errors	Any unhandled backend issue (e.g., database crash, internal error).	Return 500 Internal Server Error; log issue and show a generic error message.
Invalid Quiz Data from Frontend	Malformed or incomplete data submitted during quiz creation.	Validate inputs server-side; return 400 Bad Request with detailed feedback.

8. Future Enhancements

i. User History and Scoreboard

Add a user dashboard to store and display past quiz attempts, scores, and feedback.
This will help users track progress and review previous performances over time.

ii. Multi-User Real-Time Battles

Enable real-time quiz contests between multiple users with live countdown and dynamic leaderboard.
This adds competitiveness and makes the platform more interactive.

iii. Quiz Categories and Filters

Introduce topic-based categories like Technology, Science, and Current Affairs, along with difficulty filters. This will allow users to easily discover and attempt relevant quizzes.

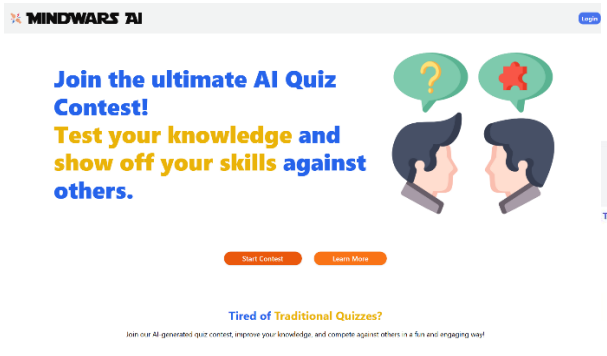
iv. Admin Panel for Content Management

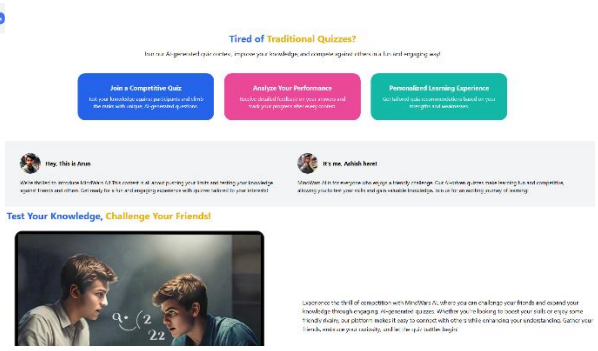
Develop an admin interface to manage users, monitor activity, and moderate AI-generated content.
It will ensure quality control and platform governance.

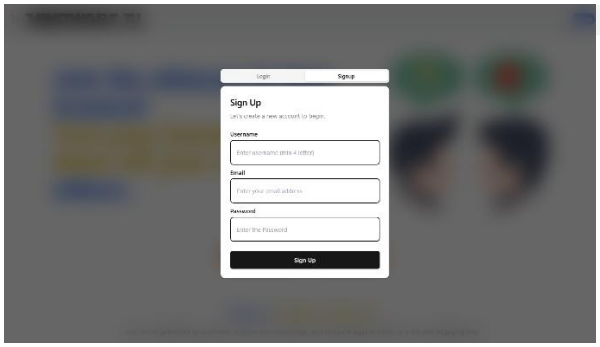
9. References

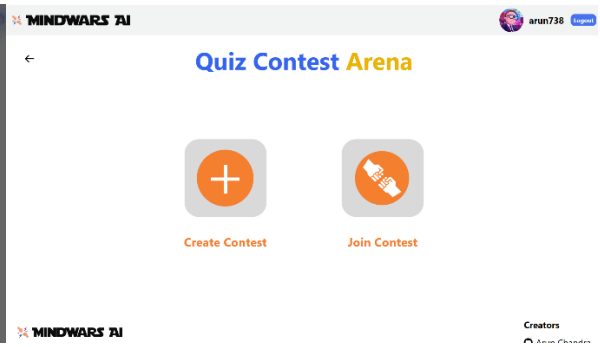
- Flask Docs: [Flask Docs]
- MongoDB Docs: [MongoDb Docs]
- React Docs: [React.js]
- Educhain & Gemini AI Docs: [Educhain-GitHub] [[AI-Google-Dev]
- Vercel & Render Deployment Guides: [Vercel] [Render]

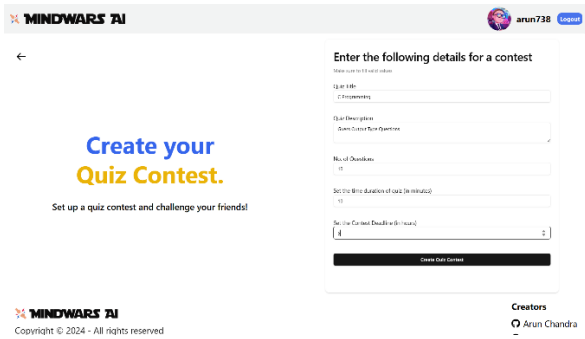
10. Implementation

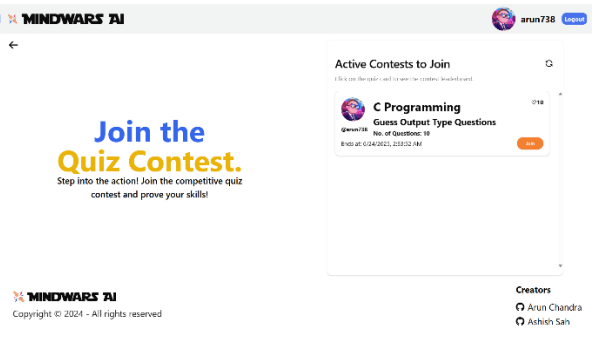


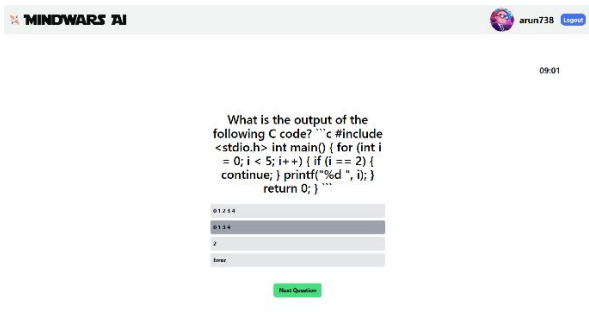


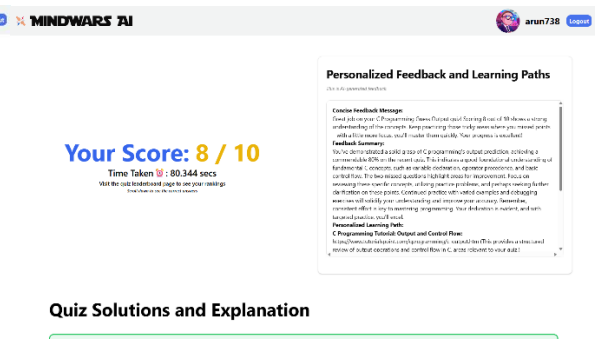


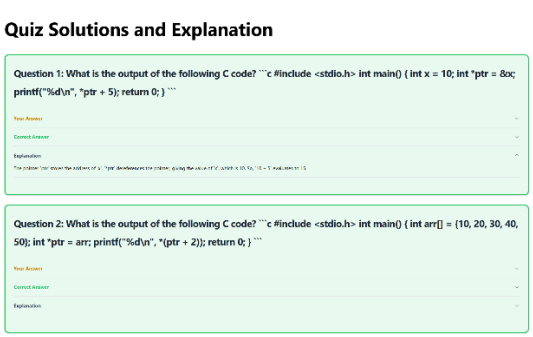


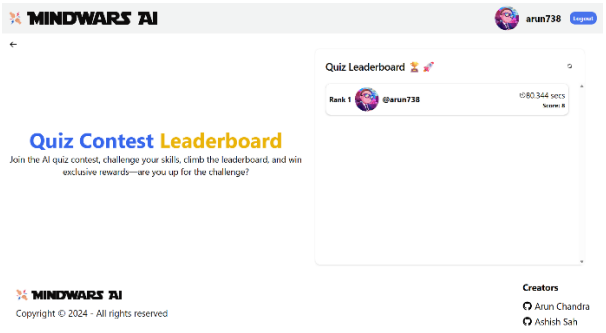






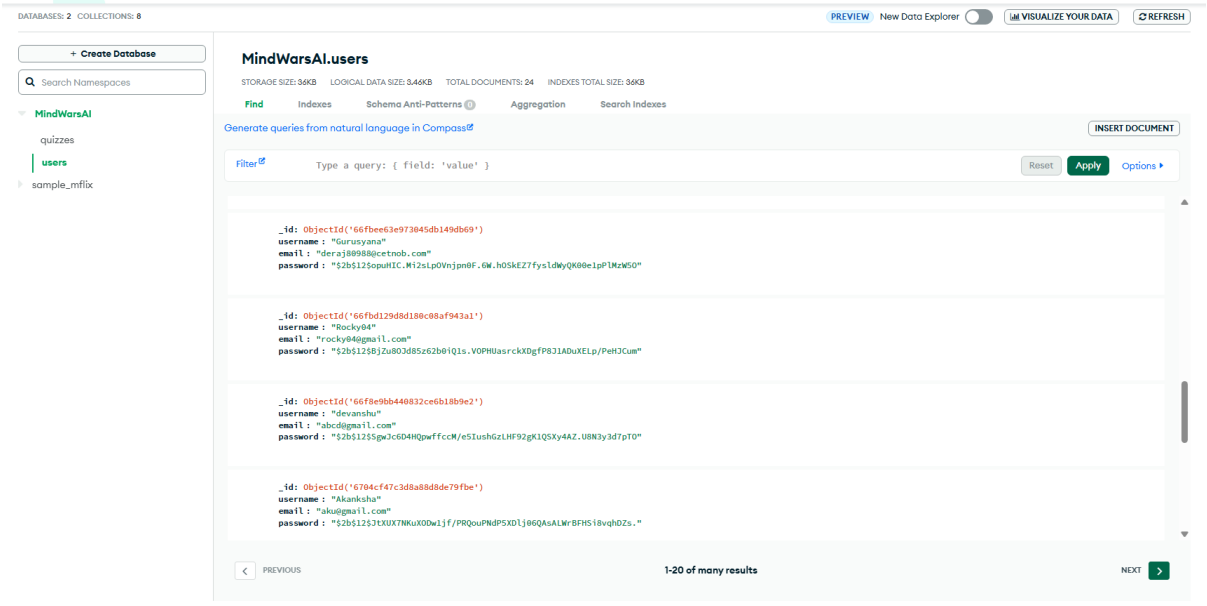


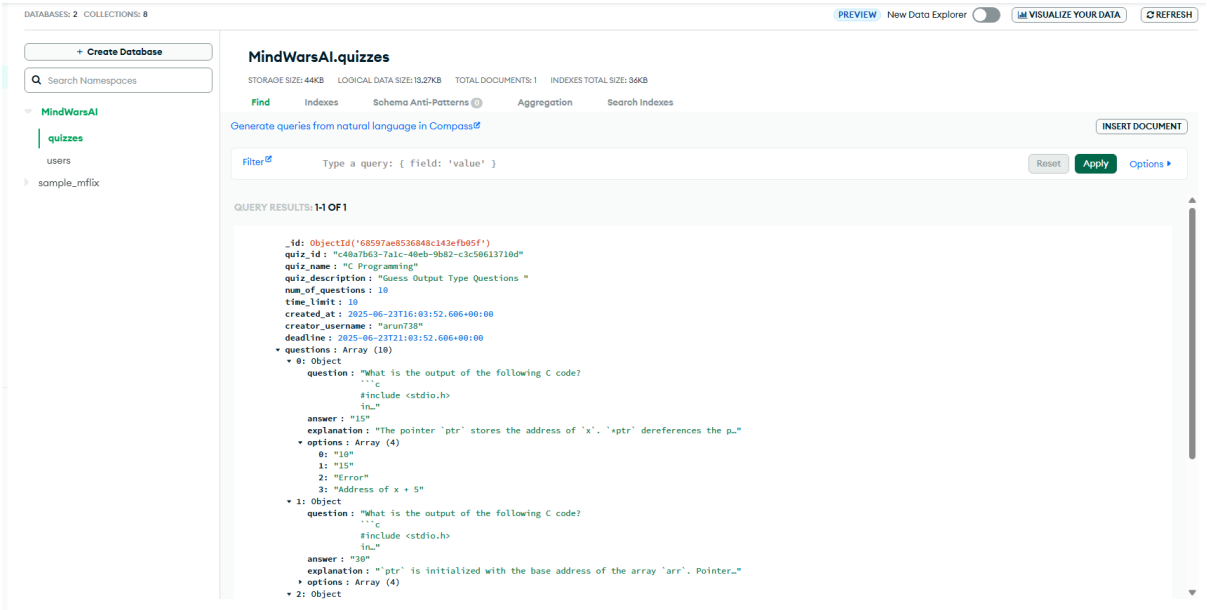




11. Appendix

MongoDB Collections:





End