COLLEGE CODE : 9623

COLLEGE NAME : Amrita College of Engineering And Technology

DEPARTMENT : Computer Science and Engineering

STUDENT NM-ID: E4405866C4E8C2C81634C314A03DF444

ROLL NO : 23CS028

DATE : 08-10-2025

Completed the project named as

Phase 1 Problem Understanding and Requirements

PROJECT NAME: INTERACTIVE QUIZ APP

SUBMITTED BY,

NAME : Brylin Shijo.S

MOBILE NO: 9600709838

Phase 4: Enhancements & Deployment

1. Additional Features

The goal here is to introduce significant, non-core functionality that increases the application's utility, competitiveness, and engagement factor.

User Authentication and Profiles: Implement a full Sign-up/Login system (e.g., using Firebase Auth, Auth0, or a custom backend) to allow users to save their progress.

User Profiles: Create a profile page displaying a user's high scores, quiz history, and badges/achievements earned.

"Guest" Mode: Ensure the application is still usable for quick, anonymous play without an account.

New Quiz Modes: Diversify the gameplay experience beyond a standard linear quiz.

Time-Attack Mode: A mode where users race against a strict clock to answer as many

questions as possible.
Survival Mode: A mode where three incorrect answers end the game (or a similar mechanic).
Administrative Panel (Minimal): Create a basic interface for a designated 'admin' user to add, edit, or remove quiz questions without needing to manually modify the database directly. This showcases full CRUD (Create, Read, Update, Delete) capability.
Question Flagging/Reporting: Implement a mechanism for users to flag questions they believe contain errors or inappropriate content.
2. UI/UX Improvements
This involves a comprehensive overhaul to ensure the application is visually appealing, intuitive to use, and accessible on different devices.
Design System Implementation: Standardize the look and feel by establishing a clear color palette, typography rules, and a set of reusable components (e.g., buttons, cards, input fields).
Accessibility (A11y) Review: Ensure the application adheres to basic accessibility standards, particularly:

Keyboard Navigation: All elements are reachable and operable via keyboard.
Color Contrast: Text-to-background contrast is high enough for readability (WCAG standards).
Screen Reader Support: Use of proper HTML semantics and ARIA attributes.
Responsive Design: Thoroughly test and fix layout issues on a variety of devices (mobile, tablet, desktop) to ensure a seamless experience across all screen sizes.
Micro-interactions and Feedback: Add subtle animations, sound effects (optional), and clear visual cues for user actions:
Correct/Incorrect Feedback: Immediate, clear visual feedback after answering a question.
Loading States: Distinct loading spinners or skeleton screens for content fetching.
Error Handling: User-friendly messages for network issues, failed login attempts, etc.
3. API Integrations and Data Enhancements
Focus on leveraging external services or creating more sophisticated data handling within the application.
External API Integration (Optional but

Recommended): Integrate a third-party API (if applicable) to fetch content, like trivia categories, or external images related to the questions. Example: Integrating a "Quote of the Day" API on the dashboard.
Sophisticated Scoring Algorithm: Move beyond simple +1 per correct answer.
Implement Time-Based Scoring: Award bonus points for faster correct answers.
Implement Difficulty-Based Scoring: Assign higher point values to questions marked as 'Hard'.
Testing and Documentation of New Endpoints: Write unit and integration tests for all newly created backend endpoints supporting the additional features (e.g., profile updates, score saving). Update the Postman/OpenAPI documentation accordingly.
4. Performance & Security Checks
This is a critical step to ensure the application is fast, reliable, and protected against common vulnerabilities before the final deployment.
Security Audit and Hardening:
Input Validation: Ensure all user input (login

forms, profile updates) is validated on both the client and server sides to prevent Cross-Site Scripting (XSS) and SQL Injection.
Data Encryption: Confirm that sensitive data (passwords, API keys) is properly hashed (e.g., with bcrypt) and that all communication is over HTTPS.
Rate Limiting: Implement limits on API requests to prevent abuse of the backend services.
Performance Optimization:
Code Splitting/Lazy Loading: Optimize frontend loading times by only loading necessary components initially.
Image Optimization: Ensure all application images are properly compressed and served in modern formats (e.g., WebP).
Query Optimization: Review database queries to ensure they are efficient and indexed to minimize latency (especially for fetching high scores and quiz history).
Pre-Deployment Testing:
Stress Testing: Simulate multiple concurrent users to check if the application's backend can handle the load.

Browser Compatibility: Test the application on the latest versions of Chrome, Firefox, Safari, and Edge.
Final Regression Test: A full run-through of all core features (from Phase 3) to ensure the new Phase 4 enhancements haven't accidentally broken existing functionality.