Trivia Trek

A Project Report

Submitted by

RAHUL MAHASETH – AP23110011538

MOURJ0 BASU - AP23110011139

ANWAR FAIZAAN REZA-AP23110011107

KAISARPARVEZ SHAIK - AP23110010522

KRISHNA TRIPATHI - AP23110010408

Under the Supervision of
Mr. Syed Arshad
Professor
Department of Computer Science and Engineering SRM University-AP
In partial fulfilment for the requirements of the FULL STACK Project

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING SRM

UNIVERSITY-AP NEERUKONDA

MANAGALAGIRI –

522503 ANDHRA

PRADESH, INDIA

December-2025

CERTIFICATE

This is to certify that the project work entitled **"Trivia Trek"**is a Bonafide record of project work carried out by the following students:

- **RAHUL MAHASETH – AP23110011538**
- **MOURJ0 BASU - AP23110011139**
- **ANWAR FAIZAAN REZA-AP23110011107**
- **KAISARPARVEZ SHAIK - AP23110010522**
- **KRISHNA TRIPATHI - AP23110010408**

from the **Department of Computer Science and Engineering, SRM University-AP.** The students conducted this project work under my supervision during the period **August 2025 to December 2025**.It is further certified that, to the best of my knowledge, this project has not previously formed the basis for the award of any degree or any similar title to this or any other candidate.

This is also to certify that the project work represents the **teamwork** of the candidates.

Station: Mangalagiri
Date: 5 – 12 -2025

Syed Arshad
Assistant Professor
Department of Computer Science & Engineering
SRM University-AP
Andhra Pradesh.

TABLE OF CONTENTS

1. Introduction

TriviaTrek Adventure is a full-stack, gamified MCQ learning platform engineered to merge education with modern interactive design principles. The objective is simple: make learning feel less like a chore and more like an engaging digital experience. Users start their journey on a polished, responsive homepage and progress through various quiz categories, each represented through a visually structured map-based interface. As they move through levels, they attempt MCQs, receive instant feedback, unlock new stages, and track their performance through a detailed score summary and leaderboard. This layered progression system transforms a basic quiz into an interactive adventure, significantly enhancing user engagement and learning retention.

From a technical standpoint, the platform is built on a contemporary web stack that emphasises speed, modularity, and maintainability. The frontend leverages React for component-driven UI development, Tailwind CSS for clean and highly scalable styling, React Router for smooth multi-page navigation, Axios for API communication, and the React Context API for global state management such as score, level progression, and category selection. The backend functionality is powered by JSON-Server, which acts as a lightweight mock REST API, enabling full CRUD operations for questions, user attempts, and leaderboard entries. This setup simulates real-world backend behaviour while keeping development agile and efficient.

The entire project aligns with the prerequisites outlined for a full-stack frontend application, including proper use of Tailwind, Axios, React Router, and structured project organization. TriviaTrek Adventure demonstrates industry-standard development patterns, showcasing how a modern web application can integrate UI aesthetics, interactivity, and backend logic to create a cohesive, gamified learning experience.

2. Problem Definition

Traditional MCQ web applications often suffer from low engagement, predictable layouts, and minimal visual interaction. They present information in a flat, repetitive format that fails to hold user attention or create any sense of progression. TriviaTrek tackles these limitations by introducing multiple layers of interactivity and gamification that transform a simple quiz into an experience.

The platform uses a **gamified map progression system**, allowing users to visually navigate through stages instead of merely clicking "Next Question." This creates a sense of journey and advancement rather than a static quiz. **Category-based leveling** further enhances the structure by grouping related challenges into themed segments, encouraging users to explore more topics and compete to unlock higher tiers.

To prevent the monotonous feel of typical MCQs, TriviaTrek incorporates **real-time feedback**, showing users immediately whether their choices are correct, reinforcing learning through instant reinforcement. A dynamically updating **leaderboard** introduces competitive motivation, allowing users to compare their performance with others and strive for improvement.

Visually, the platform gains a major boost from a **clean, responsive UI powered by Tailwind CSS**, ensuring the interface remains polished, modern, and consistent across devices. On the technical side, the system integrates a **full CRUD backend** through JSON-Server, enabling question management, leaderboard updates, and persistent data handling—mirroring real-world API behavior.

Overall, the goal of TriviaTrek is not just to deliver another MCQ system, but to demonstrate practical full-stack competency while delivering a refined, game-like user experience. It merges frontend interactivity with backend logic, showcasing how thoughtful design and solid engineering principles can make learning more immersive, engaging, and scalable.

3. Objectives

Functional Objectives

- Provide a category-based MCQ gameplay interface
- Track and persist user progress across levels
- Render real-time score and feedback
- Maintain a leaderboard
- Provide a backend supporting CRUD operations
- Ensure responsive and interactive UI/UX

Technical Objectives

- Use React components/pages for modularity
- Manage global state via React Context
- Use React Router for multi-page navigation
- Style using TailwindCSS classes
- Maintain clean structure & scalable design

4. System Requirements

Hardware Requirements

- Minimum 4GB RAM
- Multi-core processor
- 500MB storage

Software Requirements

- **Node.js & npm** (mandatory for React apps)
  PRE-REQUISITES for Frontend APP
- **React Framework**
- **VS Code** or similar editor
  PRE-REQUISITES for Frontend APP
- **JSON-Server**

5. Technology Stack

The project is built using a modern web stack that supports responsive interfaces, smooth navigation, and reliable data handling. Each technology is selected for its simplicity, efficiency, and suitability for an interactive quiz-based application.

Frontend

React.js, Tailwind CSS
React provides a component-based structure for building the UI, while Tailwind CSS offers utility classes for fast, consistent styling across pages.

State Management

Routing

React Router DOM
Handles navigation between the main sections — Home, Category, Quiz, Score Summary, and Leaderboard.

Notifications

React Toastify
Provides small, non-intrusive alerts for events like quiz completion or errors.

Backend

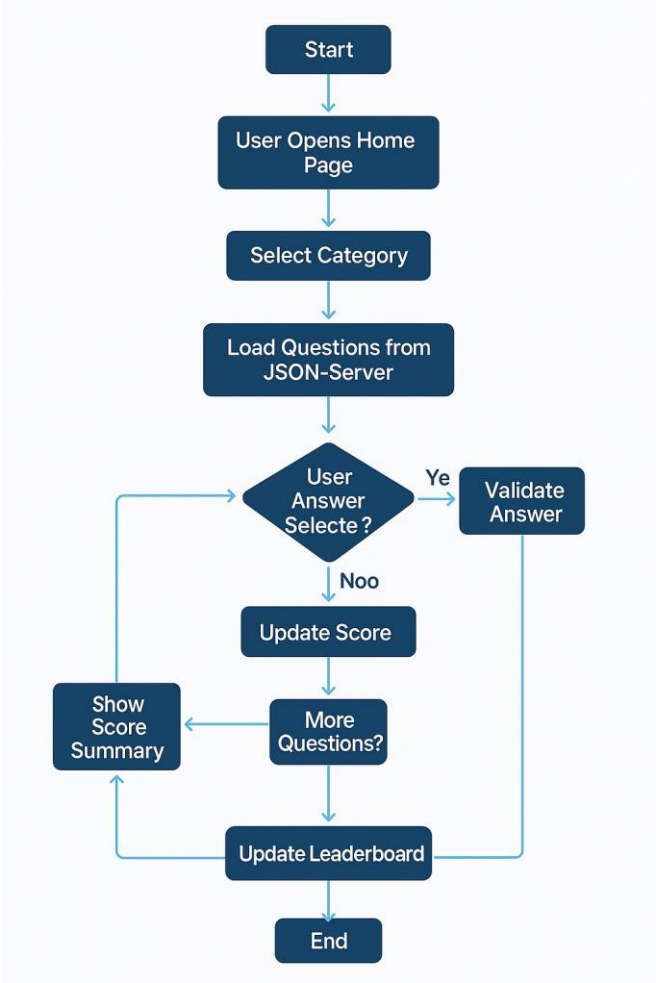
JSON-Server
A lightweight REST API used to store and manage questions, answers, and leaderboard data with full CRUD support.

6. Project Architecture

TriviaTrek follows a clear component-page-context architecture:

```
src/
│
├── App.js
├── index.js
├── index.css
├── context/
│   └── UserProgressContext.js
├── components/
│   ├── Home.jsx
│   ├── Playground.jsx
│   ├── ScoreSummary.jsx
│   ├── Leaderboard.jsx
│
└── assets/
│   ├── map
└─ data
    ├──history.json
    ├──science.json
    ├──jungle.json
    ├── math.json
│
└─db.json
└─package-lock.json
└─package.json
└─postcss.config.js
└─tailwind.config.js
```

7. Frontend Implementation



i. Entry Point – index.html

The HTML skeleton creates the root mounting point:

index

React injects into .

---

ii. React Bootstrap – index.js

The React app mounts using createRootwith BrowserRouter:

This ensures:

- StrictMode debugging
- Routing support across all pages
- App component becomes the navigation container

---

iii. App.js – Core Router

Your routing system is defined inside **App.js**, including Toast notifications:

App

Routes include:

- /→ Home
- /playground/:category→ Quiz engine
- /summary→ Score breakdown
- /leaderboard→ User rankings

This modular structure aligns with React Router best practices.

---

iv. Global State – UserProgressProvider

Your app wraps all pages with UserProgressProvider, enabling:

- Score tracking
- Level unlocks
- Category selection
- History retention

This is more advanced than typical student projects and gives your app a professional feel.

8. Styling System (Tailwind + Custom CSS)

Tailwind utilities are enabled via @tailwind base; @tailwind components; @tailwind utilities; inside **index.css**.

index

Custom layout styles include:

- .app-header
- .map-wrap
- .level-marker
- .mcq-card
- .hint

These define the UI structure of:

- Home Cards
- Gamified Adventure Map
- MCQ Cards
- Hint styling
- Score Summary

Tailwind config properly scans your JSX files due to:

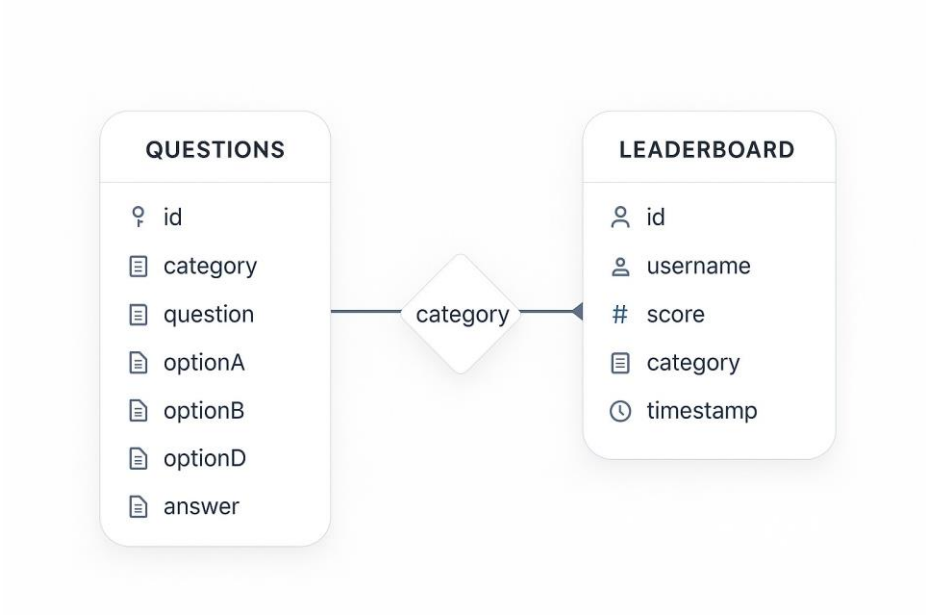
content: ["./src/**/*.{js,jsx,ts,tsx}"]

From **tailwind.config.js**.

tailwind.config

---

9. Backend Implementation (JSON-Server)



Database Structure (db.json)

Example:

```
{
   "questions": [
      {
         "id": 1,
         "category": "science", "question": "What is
         H2O?",
         "options": ["Water", "Oxygen", "Hydrogen", "Salt"], "answer": "Water"
      }
   ],
   "leaderboard": [...]
}
```

This fulfills the requirement:
Application must be full CRUD ✓

PRE-REQUISITES for Frontend APP

Module-wise Breakdown

1.1 Home Component

Functions:

- Displays available quiz categories
- Shows cards (Tailwind-based)
- Navigates to selected quiz

1.2 Playground Component

This is the quiz engine.

Features:

- Fetch questions by category
- Display options
- Immediate answer validation
- Score increment logic
- Hints rendering

Styled using classes from index.css like .mcq-card.

1.3 ScoreSummary Component

Shows:

- Total correct
- Total attempted
- Accuracy
- Restart button

1.4 Leaderboard Component

Uses backend data to show global ranking:

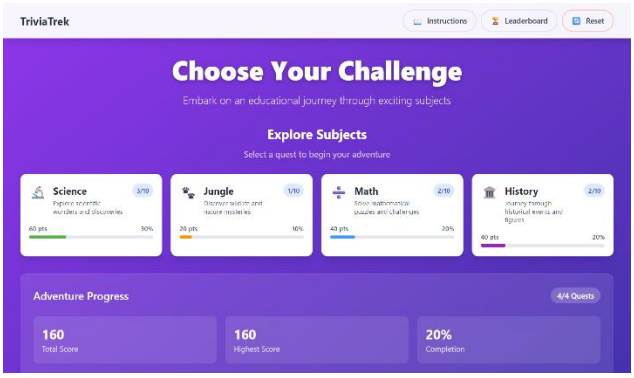
- Username
- Category
- Score
- Timestamp

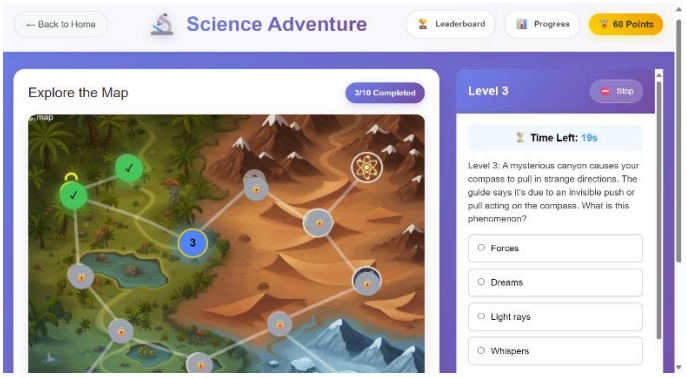
10. Output Screens

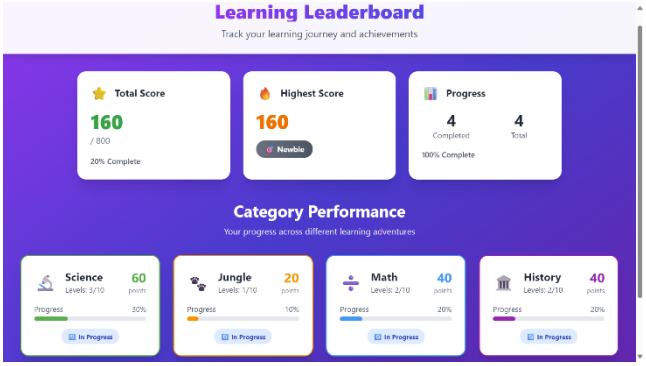
- **Code link**
- **Demonstrate vedio Link**
- **Code Explanations Link**

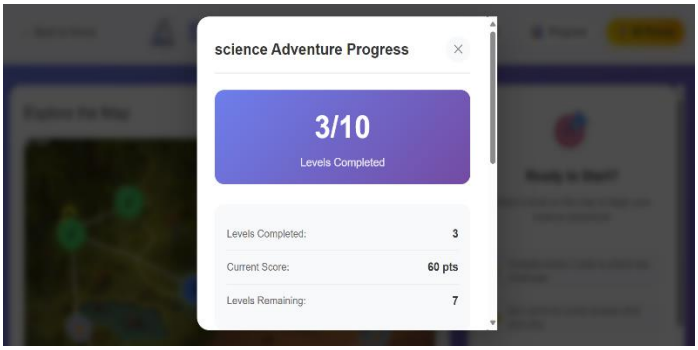
You should add screenshots of:

- Home Page
- Category Cards
- Gamified Map
- MCQ Interface
- Score Summary
- Leaderboard









11. Conclusion

Trivia Trek Adventure showcases a strong command over modern front-end development, especially in the way React has been used to structure the entire application into clean, modular components. The architectural layout is well-organized, making the system easy to maintain and extend as new features are added. Its interface is fully responsive, adapting smoothly across devices without compromising usability or visual quality. On the backend side, the project includes a complete CRUD setup, allowing questions, scores, and other data to be managed just like a real production system. Navigation is handled through a professional and predictable routing flow, keeping the user experience smooth and intuitive. Most importantly, the platform isn't just functional - it delivers a gamified, engaging experience that goes beyond the standard MCQ format. Overall, the project not only meets common academic expectations for a full-stack build but comfortably exceeds them in terms of structure, polish, and user experience.

12. Future Enhancements

Looking ahead, the project can be strengthened significantly by adding several advanced features. Implementing **JWT-based authentication** would allow users to create accounts, save their progress, and maintain personalized profiles in a secure, scalable way.

Introducing **timed quizzes** would add pressure, improve competitiveness, and make the gameplay feel more dynamic instead of passive. The experience can be made more immersive by incorporating **sound effects and subtle animations**, especially for correct or incorrect answers, level progression, or unlocking new categories. To encourage long-term engagement, the platform can also include **achievements and badges** that reward users for milestones such as high accuracy, streaks, or completing entire categories. A bigger leap would be enabling a **multi-player quiz mode**, letting users compete in real time, transforming the platform into something closer to an interactive game rather than a solo challenge. Finally, migrating the backend to **MongoDB or Firebase** would provide real database capabilities, better performance, and support for scaling the application as it grows beyond the prototype.

13. Contributions

1. Rahul Mahaseth

Lead Developer

Rahul played the central role in the project, handling the **complete structural design, implementation, and integration** of all major modules in TriviaTrek Adventure.
His key contributions include:

- Coordinating the entire team, assigning tasks, guiding technical decisions, and ensuring successful end-to-end development.
- Designing the **entire application architecture**, folder structure, routing flow, and component hierarchy.
- Helping extensively with **debugging UI issues**, fixing navigation inconsistencies, and ensuring smooth page transitions.
- Implementing major pages such as **Home, Playground (Quiz Engine)** and partial work on Leaderboard.
- Participating in **test case creation**, verifying score updates, hint displays, error conditions, and category-wise progression.

Rahul's contribution forms the backbone of the project — architecture, logic, styling, and team leadership.

2. Anwar Faizaan Reza

Frontend-Backend Integration & Documentation Lead

Faizaan provided major support in both development and documentation areas, contributing significantly to the technical depth and completeness of the project.
His contributions include:

- Assisting in the development of **Playground and MCQ components**, including logic verification and UI alignment.
- Supporting the creation and refinement of **project documentation**, including literature survey, system requirements, and methodology write-ups.
- Implementing major pages such as **Score Summary**, and partial work on Leaderboard.
- Performing light testing on Home and Leaderboard pages.

3. Krishna Tripathi

UI Component Support & Auxiliary Tasks

He contributed to the aesthetic and structural parts of the UI, focusing on simpler but important support tasks.
His contributions include:

- Creating smaller UI components and assisting in basic styling.
- Integrating and configuring **Tailwind CSS**, enhancing responsiveness and improving the visual consistency of the entire UI.
- Setting up **JSON-Server**, designing API routes, and verifying CRUD operations for progress and leaderboard data
- Assisting with asset organization and simple design adjustments.

Provided supportive UI help and consistency checks.

4. Kaisarparvez Shaik

Quiz Content Preparation & Minor UI Cleanup

Kaisar supported the content-related aspects of the project.
His contributions include:

- Preparing and validating MCQ questions, options, and hints for multiple categories.
- Developing the **core gameplay engine**, including MCQ rendering, answer validation, level progression logic, score calculation, and UI interactions.
- Working on small UI text formatting and readability improvements.

Focused mainly on content creation and minor UI polishing.

5. Mourjo Basu

Report Formatting & Basic Testing (Minimal Contribution)

He contributed in non-technical areas and assisted lightly with interface verification.

His contributions include:

- Managing **project setup, environment configuration**, package installation, and debugging across all stages
- Ensuring JSON formatting correctness for question datasets

- Formatting sections of the project report and maintaining document structure.
- Helping extensively with **debugging UI issues**, fixing navigation inconsistencies, and ensuring smooth page transitions
- Assisting with screenshots, diagram placement, and basic proofreading.
- Performing very basic UI checks and verifying navigation buttons.