# Detailed and Optimized Prompt for Creating an Advanced Formula 1 Quiz Application

# **General Objective**

Create an interactive multiple-choice quiz application focused on Formula 1, targeting two distinct user groups: "General Audience" and "Enthusiasts," featuring simplified user profile management, detailed performance statistics, and a local leaderboard.

#### **Personas**

- Casual User (General Audience): Limited general knowledge of Formula 1.
- Expert User (Enthusiasts): In-depth, technical knowledge of Formula 1.

#### **Task**

- Develop a comprehensive web application using MVC architecture.
- Implement two clearly distinguished modes.
- Integrate user management, score saving, and local leaderboard.
- Ensure performance tracking by specific themes.

#### Context

The application should include the following themes:

- Drivers
- Circuits
- Rules
- General Quiz (mixed questions)

Each theme should allow individual score tracking.

# **Expected Deliverable Formats**

1. Class Diagram (Mermaid)

```
+theme: String
        +questions: List~Question~
        +currentScore: int
        +currentQuestionIndex: int
   class Leaderboard {
        +users: List~User~
        +sortByPoints(): void
   User --> Leaderboard
   Quiz --> Question
   2. Component Diagram (Mermaid)
flowchart TD
   Frontend(UI Vue.js/React) --> Controller(Node.js)
   Controller --> Model
   Controller --> Storage(LocalStorage)
   Model --> User
   Model --> Quiz
   Model --> Leaderboard
   3. Sequence Diagram for Quiz Management (Mermaid)
sequenceDiagram
    participant User
    participant Controller
    participant Quiz
   participant Storage
   User ->> Controller: Starts quiz
   Controller ->> Storage: Load questions
   Storage -->> Controller: Return questions
   Controller ->> Quiz: Initialize quiz
    loop For each question
        Quiz ->> Controller: Display question
        User ->> Controller: User response
        Controller ->> Quiz: Verify answer
        Quiz ->> Controller: Update score
   Controller ->> Storage: Save score
   4. Use Case Diagram (Mermaid)
useCaseDiagram
   actor General Audience User
    actor Enthusiast User
    rectangle Application {
        General Audience User --> (Take simple quiz)
        Enthusiast User --> (Take technical quiz)
        (Take simple quiz) --> (View scores and leaderboard)
        (Take technical quiz) --> (View scores and leaderboard)
```

### **MVC Software Architecture**

}

• **Model**: Stores and manages user data, quizzes, and scores.

General Audience User --> (Select theme)

Enthusiast User --> (Select theme)

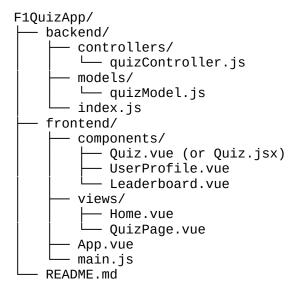
• **View**: Interactive, clear, and responsive user interface.

• **Controller**: Application logic, user interactions, and model/view updates.

## **Recommended Technologies**

- Frontend: React.js or Vue.js (responsive, fast, and easy-to-manage interfaces).
- Backend: Node.js (easy to implement, performant).
- Storage: LocalStorage (no external database needed, quick data access).

## **Initial File Structure**



## **Justification of Choices**

- **Mermaid**: Easy integration and visualization directly within documentation and version control.
- **MVC**: Clear separation of concerns, facilitating maintenance and evolution of the application.
- **JavaScript stack**: Accessible, popular, excellent performance, and an active community.
- **LocalStorage**: Reduces complexity, ideal for simple, lightweight applications without a dedicated backend.