High Level Design Document

Introduction

This High Level Design (HLD) document outlines the architecture and core components for **Quizzy - Interactive JavaScript Quiz**. Quizzy is an educational web application that tests users on JavaScript fundamentals, built using React, TypeScript, and Tailwind CSS. The design focuses on dynamic question rendering, user interaction, and real-time scoring.

---

1. System Architecture Overview

Architecture Description:
Quizzy is a single-page application (SPA) with a modular frontend architecture. All logic and data are managed client-side. The system is composed of distinct React components, each responsible for a specific function.

Module/Component	Role/Responsibility
App Shell	Initializes app, manages routing/layout
Quiz Engine	Loads questions, manages quiz state and flow
Question Renderer	Dynamically displays questions and options
Answer Handler	Captures user input, validates answers
Scoring Module	Calculates and displays user score
UI Components	Provides styled buttons, progress bars, etc.

---

2. Component Interactions

Sequence Step	Interaction Description
1. App Initialization	App Shell loads Quiz Engine and UI Components
2. Quiz Start	Quiz Engine fetches questions, passes to Question Renderer
3. User Answers Question	Answer Handler receives input, updates Quiz Engine
4. Score Update	Scoring Module updates score, triggers UI update
5. Quiz Completion	Final score displayed, option to restart quiz

---

3. Data Flow Overview

Data Entity	Source Component	Destination Component	Purpose
Questions	Quiz Engine	Question Renderer	Display next question

User Answer	Answer Handler	Quiz Engine	Validate and update state
Score	Scoring Module	UI Components	Show current/final score
Quiz State	Quiz Engine	All Components	Manage progress and transitions

---

4. Technology Stack

Layer/Aspect	Technology/Framework
Frontend	React (with Hooks)
Language	TypeScript
Styling/UI	Tailwind CSS
State Management	React Context/Local State
Build Tooling	Vite or Create React App

---

5. Scalability & Reliability

- **Scalability:**
  The SPA architecture supports easy addition of new question sets or quiz features. Component-based design enables modular enhancements.
- **Reliability:**
  All logic is client-side, minimizing external dependencies. TypeScript ensures type safety and reduces runtime errors.
- **Security:**
  No sensitive data is handled. Input validation prevents malformed answers.

End of Document