Brainiac Challenge

Application Specification

Generated by Gemini for AUCDT ICT

August 1, 2025

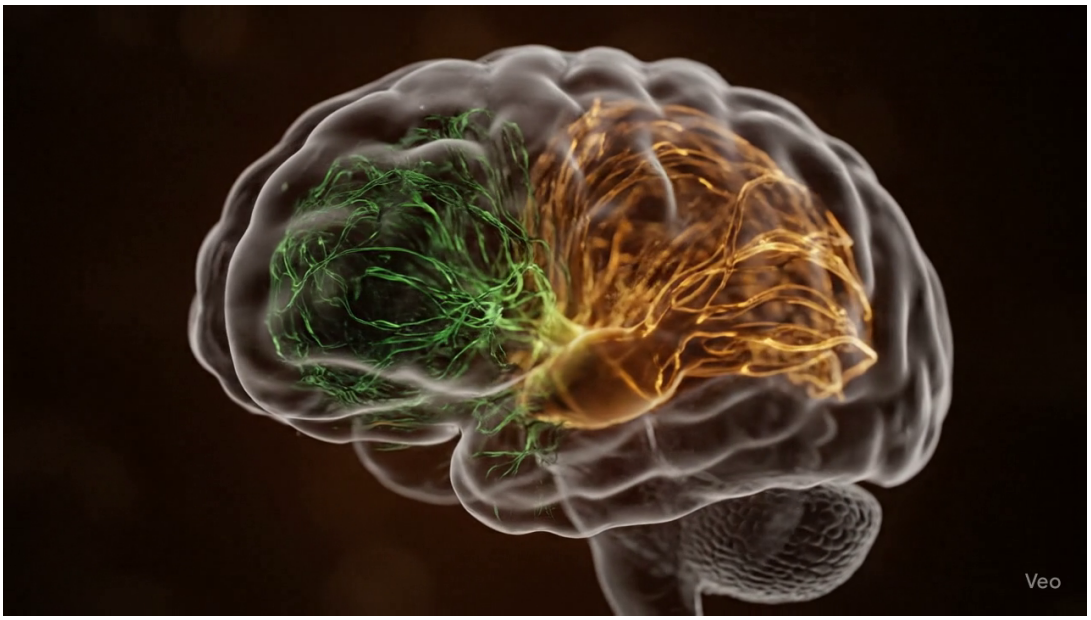

Figure 1: A demonstration of an embedded video. Click the image to play.

Contents

Abstract

This document provides a detailed technical and functional specification for the **Brainiac Challenge**, a dynamic, AI-powered quiz generation application. The application is designed to be an engaging educational tool for students across all academic levels, from Primary School to University. It leverages the Google Gemini API for real-time question generation and Google Firebase for user authentication and data persistence. Key features include customisable quiz parameters, level-adaptive topic selection, a persistent audit log of all generated quizzes, and data import/export capabilities. The application is sponsored by and branded for the AsanSka University College of Communications (AUCDT).

1 Introduction

1.1 Purpose

The purpose of this document is to define the requirements and system architecture for the Brainiac Challenge application. It serves as a central guide for developers, stakeholders, and project managers, ensuring a common understanding of the application's scope, features, and technical implementation.

1.2 Scope

The application is a web-based, single-page application (SPA) that allows users to:

- Configure and generate customised quizzes on any topic.
- Select their educational level to receive relevant subject suggestions.
- Take quizzes in an interactive interface.
- View their score and performance feedback.
- Maintain a persistent, time-stamped log of all generated quizzes.
- Export and import their audit log data for backup and analysis.

1.3 Target Audience

The primary audience for this application is students of all academic levels, including:

- Primary School
- Junior High School (JHS)
- Senior High School (SHS)
- University

The application is also intended for educators and administrators who may use it as a teaching or assessment tool.

1.4 Sponsor

This project is sponsored by the **AsanSka University College of Communications (AUCDT)**. The application's user interface, branding, and colour scheme reflect the corporate identity of AUCDT. The sponsor's logo is featured prominently and links to the official AUCDT student portal.

2 Functional Requirements

2.1 User Authentication

- **FR-1.1:** The system shall use Google Firebase Authentication to manage users.
- **FR-1.2:** On first visit, users shall be signed in anonymously to provide a unique user ID without requiring account creation. This ensures a seamless user experience while enabling persistent data storage per user.
- **FR-1.3:** The user's unique ID (UID) from Firebase shall be used to associate all generated data, specifically the audit log, with that user.

2.2 Quiz Configuration (Settings Page)

- **FR-2.1:** Users must be able to select an academic **Level** from a predefined list (e.g., Primary School, JHS, SHS, University, General).
- **FR-2.2:** Upon selection of a Level, the **Topic** input shall dynamically update to a dropdown list of relevant subjects for that level.
- **FR-2.3:** The Topic dropdown must include an "Other..." option. When selected, a text input field shall appear, allowing the user to enter a custom topic.
- **FR-2.4:** Users must be able to specify the **Number of Questions** for the quiz, with a range from 1 to 20.
- **FR-2.5:** Users must be able to select a **Difficulty** for the quiz from three options: Easy, Medium, or Hard.

2.3 Quiz Generation and Interface

- **FR-3.1:** The application shall construct a detailed prompt for the Google Gemini API based on the user's selected settings (Level, Topic, Number of Questions, Difficulty).
- **FR-3.2:** The API request shall specify a structured JSON schema for the response to ensure data consistency.
- **FR-3.3:** The quiz interface shall display one question at a time, with multiple-choice options presented in a randomised order for each session.
- **FR-3.4:** Users shall select one answer per question. The interface will prevent changing answers after the quiz is submitted.

2.4 Scoring and Results

- **FR-4.1:** Upon submission, the application shall calculate the user's score as a percentage of correct answers.
- **FR-4.2:** A results screen (Score Card) shall display the final percentage score.
- **FR-4.3:** The results screen shall provide dynamic, encouraging feedback based on the user's score.
- **FR-4.4:** The user shall be given options to "Try Again" (retake the same quiz with shuffled options) or start a "New Challenge" (return to the settings page).

2.5 Audit Logging

- **FR-5.1:** The application must provide access to an "Audit Log" page from the main settings screen.
- **FR-5.2:** Every time a quiz is successfully generated, an audit entry must be created.
- **FR-5.3:** Each audit entry must be stored as a document in a dedicated Firestore collection and contain:
  - A unique document ID.
  - A precise ISO 8601 timestamp.
  - The complete text of the prompt sent to the Gemini API.

  - The user's selected settings (topic, level, etc.).
  - The full, raw JSON response received from the Gemini API.

- **FR-5.4:** The Audit Log page shall display all log entries in reverse chronological order (most recent first).
- **FR-5.5 (Export):** Users must be able to export the entire audit log as a single JSON file.
- **FR-5.6 (Import):** Users must be able to import an audit log from a JSON file. The system must check for duplicate entries based on the timestamp and only import new, unique records.

3 Non-Functional Requirements

3.1 User Interface and Branding

- **NFR-1.1:** The application shall have a clean, modern, and responsive user interface that is intuitive on desktop and mobile devices.
- **NFR-1.2:** The UI shall be styled using Tailwind CSS.
- **NFR-1.3:** The application's colour palette, typography, and button styles shall adhere to the AUCDT branding guidelines, primarily using gold (#B8860B), green (#006400), and deep brown (#5C4033).
- **NFR-1.4:** The AUCDT logo on the settings page must be a hyperlink to `https://portal.aucdt.edu.gh`.

3.2 Technology Stack

- **NFR-2.1: Frontend Framework:** React v18 (loaded via UMD scripts).
- **NFR-2.2: Language:** JavaScript (ES6+) with JSX, transpiled in-browser by Babel.
- **NFR-2.3: AI Service:** Google Gemini API (gemini-2.5-flash-preview-05-20 model) for content generation.
- **NFR-2.4: Backend Services:** Google Firebase for Authentication (Anonymous) and Database (Firestore).
- **NFR-2.5: Styling:** Tailwind CSS v3.

3.3 Performance

- **NFR-3.1:** The application shall display a loading indicator while waiting for the API response to inform the user of background activity.
- **NFR-3.2:** The audit log shall load and display entries efficiently, with real-time updates from Firestore.

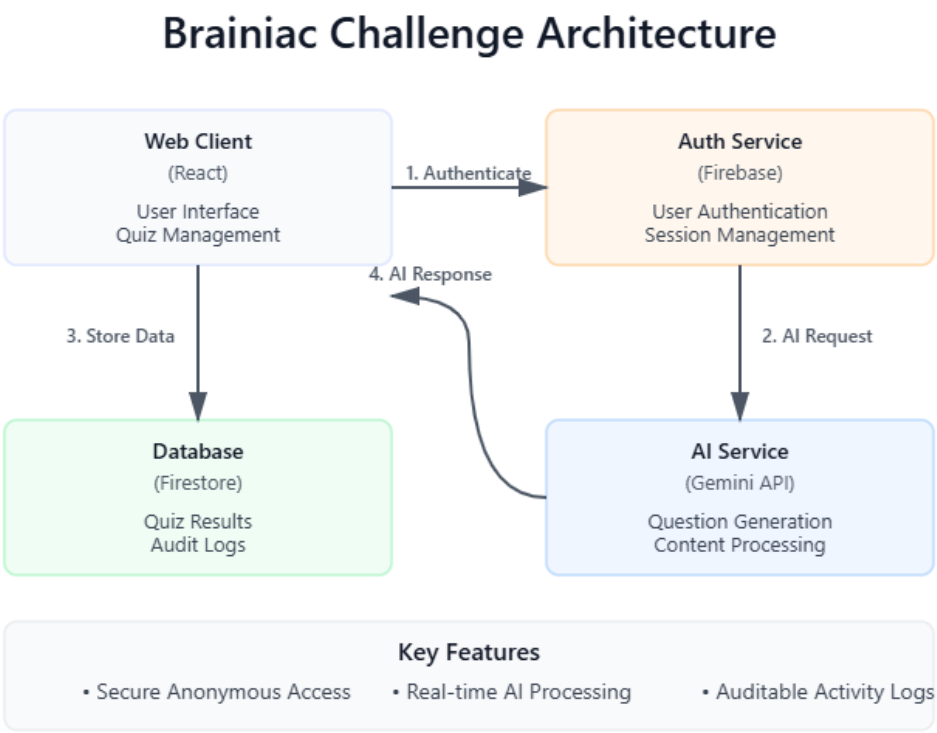

Figure 2: High-Level System Architecture

4 System Architecture

The Brainiac Challenge application is a client-side, single-page application that interacts with Google Cloud services.

1. **Client (Browser):** The user interacts with the React application running in their web browser. The UI is rendered and managed by React.
2. **Firebase Authentication:** On page load, the client communicates with Firebase Auth to sign the user in anonymously, receiving a UID.
3. **Gemini API:** When a user starts a challenge, the client constructs a prompt and sends a secure HTTPS request to the Google Gemini API endpoint.
4. **Firebase Firestore:**
   - After receiving a response from Gemini, the client writes a new document to the user's audit log collection in Firestore, using the UID to ensure data privacy.
   - When the user visits the Audit Log page, the client establishes a real-time listener to the Firestore collection to display and update the logs.

A API Payload Example

The following is an example of the JSON payload sent to the Google Gemini API for quiz generation.

```
{
  "contents": [
```

```
    {
      "parts": [
        {
          "text": "Generate a SHS exam with 10 multiple-choice questions for a SHS student. The topic is Integrated Science. The difficulty is Medium. For each question, provide 4 options and one correct answer. Ensure the options are diverse and plausible."
        }
      ]
    }
  ],
  "generationConfig": {
    "responseMimeType": "application/json",
    "responseSchema": {
      "type": "OBJECT",
      "properties": {
        "questions": {
          "type": "ARRAY",
          "items": {
            "type": "OBJECT",
            "properties": {
              "question": { "type": "STRING" },
              "options": {
                "type": "ARRAY",
                "items": { "type": "STRING" }
              },
              "answer": { "type": "STRING" }
            },
            "required": ["question", "options", "answer"]
          }
        }
      },
      "required": ["questions"]
    }
  }
}
```

Listing 1: API Payload Example