3. Software Requirements Specification

## **1. Purpose**

The purpose of this project is to develop and implement an innovative application that utilizes Large Language Model (LLM) to address the specific challenges faced by UPSC (Union Public Service Commission) aspirants during their interview preparation. This application aims to provide personalized guidance by generating interview questions tailored to each user’s Detailed Application Form (DAF) data. Additionally, it offers self-assessment tools through speech to text and response analysis, enabling users to record their responses and receive feedback to improve their speech, articulation, and communication skills, which are crucial for success in UPSC interviews. Ultimately, the project’s goal is to enhance the interview preparation process and increase the chances of success for UPSC candidates [**3**] [**4**] [**5**].

## **2. Document Conventions**

This document uses the following conventions: -

* + - references: Used figures and equations wherever appropriate, and due references are given to them with clickable jump to the referred context.
    - temperature: It is a hyperparameter that regulates the randomness, or creativity, of the AI’s responses. A higher temperature value typically makes the output more diverse and creative but might also increase its likelihood of straying from the context.

**3.1 Assumptions and Dependencies**

In the development of this project, several assumptions and dependencies played a crucial role in shaping the project’s requirements and outcomes. These factors, while not necessarily known facts, are important considerations that could impact the project if they are incorrect, not shared, or subject to change. Below, we outline the key assumptions and dependencies:

**Assumptions:**

**1. Availability of Language Model:**

- Assumption: A suitable large language model, such as Gemini or its successor, will be available for integration into the project.

- Impact of Change: If access to a language model is restricted or its capabilities change significantly, it could affect the project’s feasibility and functionality.

**2. Third-Party APIs and Services:**

- Assumption: Integration with third-party services for functionalities like user authentication or natural language processing will be feasible and stable.

- Impact of Change: Unavailability or changes in third-party APIs could require re-engineering of the software.

**Dependencies:**

**1. LLM API Google Gemini:**

- Dependency: The project relies on LLM (Google Gemini) for question generation and training.

- Impact of Change: Changes in these frameworks may require modifications in code and prompts.

**2. Web Development Technologies:**

- Dependency: Web development technologies (e.g., JavaScript and Bootstrap) are essential for the front-end of the software.

- Impact of Change: Updates or shifts in web technologies may necessitate adjustments in the user interface.

Understanding these assumptions and dependencies is essential for project planning, risk assessment. Monitoring and adapting to changes in these factors will be crucial to the project’s success and ongoing development.

**3.2 Functional Requirements**

### **3.2.1. Question Generation Description and Priority:**

REQ-1: The system must use generative AI and a large language model to create questions.

REQ-2: Questions must be relevant to the user’s educational background and areas of interest.

REQ-3: The system should consider user performance history when generating questions for appropriate difficulty.

**3.2.2. User Profile Management**

REQ-4: Users can edit their profile information at any time.

REQ-5: Profile information includes educational background, areas of interest, and topic preferences.

REQ-6: Users can specify their UPSC interview date to tailor question generation.

**3.2.3. User Registration and Authentication**

REQ-7: Users must provide a valid email address and a secure password for registration.

REQ-8: User passwords should meet minimum security requirements (length, complexity).

REQ-9: The system must verify the uniqueness of email addresses during registration.

REQ-10: Upon successful registration, users receive a confirmation email.

**3.3 External Interface Requirements**

**3.3.1 User Interfaces**

In the development of this project, the user interfaces play a crucial role in facilitating user interactions with the software. The user interfaces are designed to provide a seamless and intuitive user experience. Below, we describe the logical characteristics of each interface between the software product and the users:

**1. Web-Based User Interface (GUI):**

**• Components:** The primary user interface is a web-based graphical user interface (GUI) developed using Python Django and Bootstrap.

**• Screen Layout:** The GUI follows responsive design principles, ensuring optimal display on various screen sizes and devices.

**• Standard Buttons and Functions:** Common GUI elements such as navigation menus, user profiles, previous sessions, and help options are available on every screen.

**• Sample Screens:** Sample screen images showcasing the software’s main functionalities will be included in the user documentation to assist users in getting started.

**• Error Message Display:** Standard error message display standards will be followed, providing clear and user-friendly error messages that guide users in resolving issues.

**2. Database Connectivity:**

**• Firebase Integration:** Firebase is used for database connectivity, providing real-time data synchronization and secure storage of user-related information and generated interview questions.

**3. Question Generation Interface:**

**• Google’s Gemini Large Language Model:** The interface with Google’s Gemini Large Language Model variant, ‘gemini-1.5-pro-latest’ is integrated into the software to facilitate question generation.

**• User Interaction:** Users can input relevant information and parameters for question generation, such as topic preferences and interview context according to DAF.

**4. User Authentication and Security:**

**• Login and Registration:** User authentication interfaces will be provided to ensure secure access to the software using Firebase Authentication.

**• Data Privacy:** Interfaces for managing user data privacy and consent settings will be included, in compliance with data protection regulations such as GDPR (General Data Protection Regulation).

**5. User Feedback and Support:**

**• Feedback Forms:** Users can provide feedback and report issues through dedicated interfaces.

**• Help and Support:** Standard help and support options will be accessible from the GUI, directing users to resources and assistance.

The user interfaces are designed to focus on usability, accessibility, and a consistent user experience. The design adheres to GUI standards, responsive design principles, and best practices to enhance user satisfaction and productivity. Detailed user interface specifications will provide in-depth guidance for interface design and implementation.

**3.3.2 Hardware Interfaces**

In the development of this project, hardware interfaces play a vital role in connecting the software with various hardware components of the system. These interfaces enable data transfer, control interactions, and communication protocols. Below, we describe the logical and physical characteristics of each interface between the software product and the hardware components of the system:

**1. Web Browser Interface:**

**• Device Types:** The primary interface for users is web-based and accessible through standard web browsers on various devices, including desktop computers, laptops, tablets, and smartphones.

**• Data Interactions:** Users interact with the software via the web browser interface to access question generation, user profiles, and other functionalities.

**• Communication Protocol:** The software communicates with web browsers using HTTP/HTTPS protocols to ensure secure data transmission.

**2. Firebase Database Interface:**

**• Device Types:** The software interacts with Firebase’s cloud-based database service, which is accessed over the internet.

**• Data Interactions:** The software reads and writes data to Firebase, including user profiles, generated questions, and application state.

**• Communication Protocol:** Data interactions with Firebase are secured using HTTPS, and Firebase provides real-time synchronization for data updates.

**3. API Interface for Language Model (Google’s Gemini Large Language Model):**

**• Device Types:** The software interfaces with Google’s Gemini Large Language Model (gemini-1.5-pro-latest) over the internet.

**• Data Interactions:** Users input parameters and data to request question generation from the language model.

**• Communication Protocol:** The software sends requests to the language model API using HTTP/HTTPS protocols, and the model responds with generated questions.

**4. User Authentication Interface:**

**• Device Types:** User authentication interfaces are accessible through web browsers on various devices.

**• Data Interactions:** Users log in or register using the authentication interface to access their accounts and personalized features.

**• Communication Protocol:** Secure authentication protocols, including OAuth or token- based authentication, may be employed to ensure user data protection in the long run.

The hardware interfaces enable seamless interactions between the software and various hardware components and devices. These interfaces adhere to standard communication protocols, ensuring data security, real-time updates, and accessibility across a variety of platforms and devices.

**3.3.3 Software Interfaces**

For this project, software interfaces are crucial in establishing connections between the project and other specific software components. These interfaces define the data flow, messages, and interactions with various software components. Below, we describe the software interfaces and connections:

**1. Firebase Realtime Database Interface:**

**• Description:** The software interacts with Firebase Realtime Database, which serves as the primary data storage solution.

**• Data Items:** Data items exchanged with the database include user profiles, generated interview questions, user preferences, and application state.

**• Purpose:** The database stores user-related information and serves as a repository for generated questions, ensuring real-time synchronization and accessibility from multiple devices.

**2. Google’s Gemini Large Language Model (gemini-1.5-pro-latest) API Interface:**

**• Description:** The software interfaces with Google’s Gemini Large Language Model API for question generation.

**• Data Items:** Data items sent to the API include user input parameters, such as interview context and topic preferences. The request requires 2 parameters, model, and prompt. **Model**: This is required and must match one of the embedding models available.   
**Prompt**: This field is required. It must contain at least one message and can optionally include a context for the chat as well as examples [1].

**• Purpose:** The interface enables the software to request and retrieve personalized interview questions generated by the language model, enriching the user experience.

**3. User Authentication Interface:**

**• Description:** The software integrates with user authentication services, such as OAuth or custom authentication systems.

**• Data Items:** Data items exchanged include user login credentials, tokens, and authentication responses.

**• Purpose:** User authentication interfaces ensure secure access to user accounts, safeguarding user data and personalization features.

**4. Web Browser Interface (Client-Side):**

**• Description:** The software provides a user interface accessible via web browsers on various devices.

**• Data Items:** Data items transmitted to the client-side interface include Javascript, Boot- strap and user input.

**• Purpose:** The web browser interface enables users to interact with the software, input preferences, view generated questions, and provide feedback.

The software interfaces enable seamless interactions between the project and various software components, including databases, APIs, user interfaces, and authentication services. These interfaces define the data flow, purpose, and nature of communications, ensuring the effective functioning of the software.

**3.3.4 Communication Interfaces**

In the development of this project, various communication interfaces are essential to facilitate data exchange, user interactions, and system functionality. These interfaces encompass a range of communication functions and protocols, each serving a specific purpose. Below, we describe the communications interfaces and associated requirements:

**1. HTTP/HTTPS Protocol:**

**• Protocol Type:** Hypertext Transfer Protocol (HTTP) and its secure version, HTTPS.

**• Description:** The software utilizes HTTP/HTTPS for communication between web browsers and the server.

**• Requirements:** All communication over HTTP is secured via HTTPS to ensure data privacy and integrity.Messages are formatted as HTTP requests and responses, following standard conventions.

**Security:** HTTPS ensures data encryption and secure transmission of user interactions and data.

**2. API Communication (RESTful):**

**• Protocol Type:** Representational State Transfer (REST) architecture.

**• Description:** The software interacts with external APIs, such as Google’s Gemini Large Language Model API, using RESTful communication.

**• Requirements:** Data exchange occurs via HTTP/HTTPS requests.Data is formatted as JSON (JavaScript Object Notation) for compatibility and ease of parsing.

**Security:** API keys and authentication tokens are employed for secure API access.

**3. Data Synchronization (Firebase Realtime Database):**

**• Protocol Type:** Firebase Realtime Database synchronization protocol.

**• Description:** Data synchronization between the software and Firebase is essential for real-time updates and data consistency.

**• Requirements:** The software utilizes Firebase’s synchronization mechanisms to ensure data consistency across devices.

**• Synchronization Mechanisms:** Firebase provides real-time data synchronization, ensuring instant updates across connected clients.

**3.4 Nonfunctional Requirements**

**3.4.1 Performance Requirements**

**1. Response Time for Question Generation:**

The system should generate personalized UPSC mock interview questions within a maximum response time of 2 seconds per question.

**Rationale:** To ensure a seamless and responsive user experience during the interview preparation process.

**2. Scalability:**

The system should be able to handle a minimum of 100 concurrent users generating questions simultaneously without a significant degradation in performance.

**Rationale:** To accommodate peak usage times and provide a consistent user experience.

**3. Accuracy of Question Relevance:**

The generated questions should be contextually relevant to the user’s profile and preparation level with an accuracy rate of at least 90 percent.

**Rationale:** To ensure that the questions generated are valuable for the user’s UPSC interview preparation.

**4. Security and Privacy Compliance:**

The system should adhere to all relevant data security and privacy regulations, ensuring the protection of user data and confidentiality.

**Rationale:** To maintain user trust and legal compliance.

**5. Compliance with UPSC Syllabus Changes:**

Ensure that the system can adapt to changes in the UPSC syllabus within one week of the official announcement.

**Rationale:** To provide users with updated and relevant content as per UPSC requirements.

**3.4.2. Safety Requirements**

Safety Requirements for this project are as follows:

**1. Data Privacy and Confidentiality:**

* Ensure compliance with data protection regulations, including GDPR, and safeguard users’ personal information.
* Implement strong encryption mechanisms to protect user data during storage and transmission.

**2. Content Moderation:**

* Implement content moderation mechanisms to prevent the generation of offensive, inappropriate, or harmful questions.
* Establish guidelines and policies for identifying and filtering out potentially harmful content.

**3. User Authentication and Authorization:**

* Implement robust user authentication and authorization mechanisms to prevent unauthorized access and misuse of the system.
* Ensure that only authenticated users can access and generate mock interview questions.

**4. Response Handling:**

* Provide clear guidelines and disclaimers on how users should interpret and use the generated interview questions.
* Emphasize that the questions are for practice purposes only and should not be considered as official UPSC content.

**5. Compliance with Ethical Guidelines:**

* Adhere to ethical guidelines and best practices for AI and machine learning model development.
* Stay updated with evolving ethical standards and adapt the system accordingly.

By addressing these safety requirements, the project can ensure that users have a safe and secure experience while using the personalized UPSC mock interview question generation system.

**3.4.3. Security Requirements**

Security Requirements for this project are as follows:

**1. User Authentication:**

* Implement secure user authentication mechanisms, including password hashing and multi-factor authentication (MFA) to protect user accounts.
* Enforce strong password policies and regular password expiration.

**2. Data Encryption:**

* Encrypt sensitive data, including user profiles, interview question histories, and authentication tokens, using industry-standard encryption algorithms and protocols (e.g., TLS/SSL).
* Ensure that data at rest and in transit is protected.

**3. Access Control:**

* Enforce strict access control policies to limit system access to authorized personnel only.
* Implement role-based access control (RBAC) to manage permissions effectively.

**4. Secure APIs:**

* Ensure that all APIs used by the system are protected against unauthorized access and abuse.
* Use API keys or tokens for authentication and rate limiting to prevent abuse.

**5. Vulnerability Scanning and Penetration Testing:**

* Conduct regular vulnerability scans and penetration tests to identify and remediate potential security weaknesses in the system.
* Address identified vulnerabilities promptly.

**6. Incident Response Plan:**

* Develop and maintain an incident response plan that outlines procedures for detecting, reporting, and mitigating security incidents.
* Establish a dedicated incident response team responsible for addressing security breaches.

**7. End-of-Life Security:**

* Develop a plan for securely decommissioning the system at the end of its lifecycle, including data sanitization and disposal.

## **3.4.4. Software Quality Attributes**

Software Quality Attributes for this project are as follows:

1. **Adaptability:**

* Ensure that the system can adapt to changing user needs, such as updates in UPSC exam patterns and question trends.
* Implement a flexible architecture that allows for easy integration of new features and content.

1. **Availability:**

* Maintain a high level of system availability to ensure users can access the platform when needed.
* Implement redundancy and failover mechanisms to minimize downtime.

1. **Correctness:**

* Verify the correctness of generated questions and system responses to provide accurate and reliable mock interview experiences.
* Implement automated testing and validation processes to catch and rectify errors.

1. **Flexibility:**

* Design the system with a modular and extensible architecture that allows for the addition of new features and components without major code overhauls.
* Enable users to customize their mock interview experiences to align with their specific needs and preferences.

1. **Reliability:**

* Build a reliable system that consistently delivers accurate and relevant mock interview questions.
* Implement error-handling mechanisms to gracefully handle unexpected issues and maintain system stability.

1. **Robustness:**

* Ensure that the system can withstand unexpected inputs and recover gracefully from errors.
* Implement thorough testing, including boundary testing and stress testing, to assess system robustness.

By focusing on these software quality attributes, the project aims to deliver a robust, adaptable, and user-friendly UPSC mock interview question generation system that meets the highest standards of quality and user satisfaction.

**3.5 System Requirements**

**3.5.1 Database Requirements**

**Database Type:** NoSQL Database (Firebase Firestore)

**Data Model:** The system will primarily store two types of data:

* UPSC Interview Question Dataset: This data will include past UPSC interview questions and relevant information. The specific structure of this data will depend on the chosen data source and may include fields such as question text, topic area, year, and difficulty level.
* Candidate Data: This data will be sourced from the candidate's Detailed Application Form (DAF) and will likely include information such as educational qualifications, work experience, and keywords extracted from the DAF content.

**3.5.2 Software Requirements**

The software environment will require a recent web browser and an internet connection for users to access the application.

**3.5.3 Hardware Requirements**

The system should have minimal hardware requirements as it is a web-based application. Standard specifications for a personal computer or mobile device with a web browser and internet connection will suffice. This means users can access the application from a variety of devices including laptops, desktops, tablets, and smartphones.