6. Project Implementation

In this chapter, we provide an in-depth exploration of the various modules, components, and functionalities implemented within the project. From user authentication and profile management to question generation, test session management, and feedback evaluation, each aspect of the system is meticulously designed to offer a seamless and effective user experience.

The project implementation phase involves the integration of multiple technologies and frameworks, including Google Firebase for user data management, HTML, CSS, and JavaScript for frontend development, and Node.js for backend logic. Additionally, the integration of Google Gemini LLM Model API for question generation and speech-to-text libraries for user interaction, enhances the functionality and capabilities of the system.

Throughout this chapter, we provide detailed insights into the architecture, design decisions, and technical considerations that shape the implementation of the project. Furthermore, we discuss the deployment process, testing methodologies, and ongoing maintenance strategies employed to ensure the reliability, scalability, and security of the application.

By offering a comprehensive overview of the project implementation process, this chapter aims to provide readers with a thorough understanding of the technical underpinnings and practical considerations involved in bringing the project to fruition. Additionally, it serves as a valuable resource for developers and future researchers seeking to explore and extend the capabilities of the proposed system.

**6.1 Overview of Project Modules**

1. **User Authentication Module**:
   * This module handles user authentication and authorization.
   * Users can register, log in, and manage their accounts securely.
   * Firebase Authentication is utilized for user authentication.
2. **Profile Management Module**:
   * Allows users to create and update their profiles.
   * Users can input their detailed application form (DAF) information, which serves as the basis for generating personalized mock interview questions.
   * Profile data is stored in Firebase Firestore database.
3. **Question Generation Module**:
   * This module generates personalized UPSC mock interview questions based on the user's profile information.
   * Utilizes prompts constructed from the user's DAF to call the Google Gemini LLM Model API.
   * The Gemini LLM Model API generates questions, answers, and approaches tailored to the user's background.
4. **Test Session Management Module**:
   * Manages the entire mock interview test session.
   * Presents generated questions to the user one by one.
   * Provides options for the user to answer questions via typing or speech-to-text.
   * Captures and stores user responses for evaluation.
5. **Response Evaluation Module**:
   * Compares the user's responses with the model-generated answers.
   * Evaluates the tone, understanding, and approach of the user's responses.
   * Provides feedback to the user based on the comparison.
6. **Session History Module**:
   * Maintains a record of all test sessions conducted by the user.
   * Allows users to view their past test sessions along with feedback provided.
   * Enables users to track their progress over time and identify areas for improvement.
7. **User Interface (UI) Module**:
   * Implements the frontend interface of the web application.
   * Designs a user-friendly interface for easy navigation and interaction.
   * Utilizes HTML, CSS, and JavaScript to create responsive and visually appealing UI components.
8. **Backend Integration Module**:
   * Handles backend logic and integration with external services.
   * Implements server-side functionalities using Node.js.
   * Facilitates communication between the frontend and backend components of the application.
9. **Deployment and Testing Module**:
   * Handles deployment of the web application to a hosting platform such as Firebase Hosting and Vercel.
   * Sets up automated testing processes to ensure the reliability and stability of the application.
   * Performs unit tests, integration tests, and end-to-end tests to validate the functionality of each module.

**6.2 Tools and Technologies Used**

1. **Google Firebase:**

* Utilized for storing user data, managing user profiles, and maintaining session history.
* Features like Firestore database for structured data storage and Firebase Authentication for user authentication are employed.

1. **HTML, CSS, JavaScript (Frontend):**

* HTML for structuring the web application's layout and content.
* CSS for styling the user interface, ensuring a visually appealing and consistent design.
* JavaScript for implementing client-side interactivity, such as handling user inputs, displaying dynamic content, and integrating with backend services.

1. **Speech-to-Text JavaScript Library:**

* Integrated for providing users with the option to convert spoken responses into text format during the mock interview.
* Enables a convenient and accessible method for users to interact with the application.

1. **Google Gemini LLM:**

* Leveraged for generating personalized UPSC mock interview questions, answers, and approaches based on user inputs.
* Provides advanced natural language processing capabilities to generate high-quality content tailored to each user's profile.

1. **Server-side Technologies:**

* Node.js is used for handling server-side logic and API endpoints.
* These technologies facilitate communication between the frontend and backend, as well as interaction with external services like Google Firebase and Gemini API.

1. **Version Control:**

* Git for version control, allowing for collaborative development, code management, and tracking of project changes.
* GitHub for hosting the project repository, enabling team members to contribute and review code changes.

1. **Development Environment:**

* Integrated Development Environment (IDE) such as Visual Studio Code for writing, editing, and debugging code efficiently.
* Tools like ESLint and Prettier for code formatting and ensuring code quality and consistency.

1. **Deployment Platform:**

* Deployment of the web application was done on platforms like Firebase Hosting and Vercel for easy and scalable hosting.
* Continuous integration and deployment (CI/CD) pipelines were set up for automated testing and deployment processes.

**6.3 Algorithm Details**

**6.3.1 Algorithm 1:**

**Generating Personalized UPSC Mock Interview Questions**

1. **Input:**

Detailed Application Form (DAF) submitted by the candidate.

1. **Prompt and Input Merging:**
2. Extract relevant information from DAF (e.g., educational background, work experience, hobbies, achievements).
3. Use this information to send to prompts for the Google Gemini API.
4. **Call Google Gemini API:**
5. Use prompts to request generation of questions, answers, and approaches from the Gemini model.
6. Receive responses containing the generated content.
7. **Output:**
8. Display the generated questions, their answers, and approaches to the candidate.
9. Allow the candidate to review and confirm willingness to take the test.
10. **End Algorithm.**

**6.3.2 Algorithm 2:**

**Conducting Personalized UPSC Mock Interview**

1. **Input:**

Candidate's choice to appear for the test.

1. **Test Session:**

If the candidate chooses to appear:

Provide questions generated in Algorithm 1 one-by-one.

Offer options to answer: typing or speech-to-text.

Capture candidate's responses.

If the candidate chooses not to appear:

Provide generated questions.

Offer option to view answers.

1. **Comparison and Feedback:**

Compare candidate's responses with Gemini-generated answers.

Evaluate tone, understanding, and approach of candidate's responses.

Provide feedback on each response based on the evaluation.

1. **Update Session History:**

Store the test session details for the candidate.

Allow candidate to view past sessions and results.

1. **Option to Retake:**

Offer candidate the option to retake the test.

Allow modifications to DAF if desired.

1. **End Algorithm.**