Software Requirements Specification for LearnifyAI

1. Introduction

1.1 Purpose

The purpose of this document is to provide a complete and detailed specification of the requirements for the LearnifyAI project. LearnifyAI is an AI-powered, web-based learning platform designed to provide an interactive and personalized educational experience for both students and teachers. This document outlines the functional and non-functional requirements of the system, serving as the foundational agreement between stakeholders on what the product will do.

1.2 Document Conventions

This document follows the IEEE standard for SRS documentation. Technical terms, acronyms, and abbreviations are defined in the Glossary section (Appendix A). Functional requirements are uniquely identified with the prefix "REQ-".

1.3 Intended Audience and Reading Suggestions

This document is intended for project stakeholders, including developers, testers, project managers, and designers.

- **Developers & Testers:** Should focus on Sections 3, 4, and 5 for detailed functional and non-functional requirements.
- **Project Managers & Stakeholders:** Can review Sections 1 and 2 for a high-level overview of the project's scope, features, and objectives.

1.4 Project Scope

The LearnifyAI project aims to create a comprehensive e-learning tool that facilitates quiz-based learning. The system will support two primary user roles: Teachers and Students. The platform is designed for use by individual educators, tutors, and self-learners who want to leverage AI for creating and taking educational quizzes.

Key objectives **include:**

- Secure user registration and role-based access.
- A robust quiz creation and management system for teachers, including AI-powered question generation from various sources (text, PDF).

- An engaging and secure quiz-taking environment for students, complete with anti-cheating measures.
- A classroom system for teachers to manage groups of students.
- An AI Study Buddy to provide on-demand learning support.

1.5 References

- Firebase Documentation: https://firebase.google.com/docs
- Google AI for Developers (Gemini API): https://ai.google.dev/
- Tailwind CSS Documentation: https://tailwindcss.com/docs

2. Overall Description

2.1 Product Perspective

LearnifyAI is a standalone, serverless web application. It is self-contained and operates entirely on the client-side, leveraging Firebase for backend services like authentication, database, and storage. It does not require any local software installation besides a modern web browser.

2.2 Product Features

The major features of LearnifyAI are:

1. **User Authentication System:** Secure registration and login for students and teachers with validation and CAPTCHA.
2. **Quiz Management (Teacher):** A full suite of tools for the creation, editing, and deletion of quizzes.
3. **AI-Powered Quiz Generation:** Automatic question generation from text and PDF content using the Gemini API.
4. **Classroom Management:** A system for teachers to create classrooms, manage student join requests, and assign quizzes.
5. **Quiz Taking System (Student):** A secure, fullscreen environment for students to take quizzes with anti-cheating features.
6. **AI** Study **Buddy:** A conversational AI tutor that can analyze text, images, and PDFs to answer student questions.
7. **Profile Management:** Personalized profile pages for both user roles to track statistics and manage personal information.

2.3 User Classes and Characteristics

- **Teachers:** Tech-savvy educators who are comfortable using web applications. They need an efficient way to create assessments, manage student groups, and view performance data without a steep learning curve.

- **Students:** Learners of varying ages and technical abilities. They need an intuitive and engaging interface to take quizzes, review their results, and get help with difficult topics.

2.4 Operating Environment

- **Frontend:** HTML5, CSS3, Tailwind CSS, JavaScript (ES6 Modules)
- **Backend Services:** Firebase (Authentication, Firestore, Storage)
- **Hosting:** Firebase Hosting
- **Compatibility:** The application will operate on any modern web browser (e.g., Chrome, Firefox, Safari, Edge) on any desktop or mobile operating system.

2.5 Design and Implementation Constraints

- The application must be a client-side, single-page application (SPA) style project.
- All backend logic must be handled by Firebase services on the Spark plan.
- PDF processing for AI generation is handled client-side using the PDF.js library.

2.6 User Documentation

This Software Requirements Specification (SRS) document serves as the primary user and system documentation.

2.7 Assumptions and Dependencies

- Users must have a stable internet connection.
- Users must use a modern web browser that supports JavaScript ES6 modules and the Fullscreen API.
- The functionality of AI features is dependent on the availability and terms of the Google Gemini API.

3. System Features

3.1 User Authentication System

- **Description and Priority:** High. This is the entry point for all users.
- **Stimulus/Response Sequences:**
    - **Stimulus:** A new user fills out the sign-up form with valid data and solves the CAPTCHA.
    - **Response:** The system creates a new user account, stores their role in the database, and redirects them to their respective dashboard.
- **Functional Requirements:**
    - **REQ-1.1:** Users shall be able to sign up with a full name, email, and password.
    - **REQ-1.2:** Users must select a role (Student or Teacher) during sign-up.

    - **REQ-1.3:** The system shall validate that the email is in a standard format and the password is at least 6 characters long.
    - **REQ-1.4:** The sign-up form shall include a simple math-based CAPTCHA to prevent bots.
    - **REQ-1.5:** Registered users shall be able to log in with their email and password.
    - **REQ-1.6:** Upon login, users shall be redirected to their respective dashboards.

3.2 Quiz Management System (Teacher)

- **Description and Priority:** High. This is a core feature for teachers.
- **Stimulus/Response Sequences:**
    - **Stimulus:** A teacher clicks the "Delete" button for a quiz and confirms the action.
    - **Response:** The system removes the quiz document from the database and refreshes the quiz list.
- **Functional Requirements:**
    - **REQ-2.1:** Teachers shall be able to create public quizzes or quizzes assigned to a specific classroom.
    - **REQ-2.2:** Teachers shall be able to add, edit, and remove questions from a quiz.
    - **REQ-2.3:** Teachers shall be able to set an optional due date for classroom quizzes.
    - **REQ-2.4:** Teachers shall be able to view a list of all their created quizzes, including the number of attempts.
    - **REQ-2.5:** Teachers shall be able to delete quizzes.

3.3 AI-Powered Quiz Generation

- **Description and Priority:** High. This is a key differentiator of the application.
- **Stimulus/Response Sequences:**
    - **Stimulus:** A teacher uploads a PDF, selects the number of questions, and clicks "Generate."
    - **Response:** The system extracts the text from the PDF, sends it to the Gemini API, and populates the quiz creation form with the generated questions.
- **Functional Requirements:**
    - **REQ-3.1:** Teachers shall be able to generate quiz questions from a block of text or a topic.
    - **REQ-3.2:** Teachers shall be able to generate quiz questions by uploading a PDF file.
    - **REQ-3.3:** Teachers shall be able to specify the number of questions and the skill level for the AI to generate.

3.4 Quiz Taking System (Student)

- **Description and Priority:** High. This is the core student experience.
- **Stimulus/Response Sequences:**

    - **Stimulus:** A student clicks "Start Quiz," enters fullscreen, and then presses the 'Esc' key.
    - **Response:** The system detects the fullscreen exit, automatically submits the quiz with any answered questions, and redirects the student to the results page.
- **Functional Requirements:**
    - **REQ-4.1:** Students shall be able to view a list of available public and classroom-specific quizzes.
    - **REQ-4.2:** Before starting a quiz, students must be presented with a lobby screen detailing the rules.
    - **REQ-4.3:** The quiz must be taken in a secure, fullscreen environment.
    - **REQ-4.4:** If a student exits fullscreen mode, the quiz shall be submitted automatically.
    - **REQ-4.5:** Text selection, copying, and right-clicking shall be disabled during the quiz.
    - **REQ-4.6:** Upon submission, students shall be redirected to a results page showing their score and a review of their answers.

3.5 Classroom Management System

- **Description and Priority:** Medium. This feature enhances teacher-student interaction.
- **Stimulus/Response Sequences:**
    - **Stimulus:** A student clicks "Request to Join" on an available classroom.
    - **Response:** The system creates a "joinRequest" document in the database with a 'pending' status. The teacher will see a new pending request on their classroom page.
- **Functional Requirements:**
    - **REQ-5.1 (Teacher):** Teachers shall be able to create, view, and manage their classrooms.
    - **REQ-5.2 (Teacher):** Teachers shall be able to view and approve/deny join requests from students.
    - **REQ-5.3 (Student):** Students shall be able to browse available classrooms and request to join.
    - **REQ-5.4 (Student):** Students shall be able to view the classrooms they are members of.

3.6 AI Study Buddy

- **Description and Priority:** Medium. This is a value-add feature for students.
- **Stimulus/Response Sequences:**
    - **Stimulus:** A student uploads an image of a math problem and asks, "How do I solve this?"
    - **Response:** The system sends the image and the text prompt to the Gemini API. The AI's step-by-step explanation is then displayed in the chat interface.
- **Functional Requirements:**

    - **REQ-6.1:** Students shall be able to engage in a conversational chat with an AI tutor.
    - **REQ-6.2:** The AI shall maintain context throughout a single session (temporary memory).
    - **REQ-6.3:** Students shall be able to upload an image or PDF for the AI to analyze in the context of their questions.

4. External Interface Requirements

4.1 User Interfaces

The application will provide a modern, responsive Graphical User Interface (GUI) built with HTML and styled with Tailwind CSS. The design is clean and professional, with a "glassmorphism" effect on the authentication pages for a modern aesthetic.

4.2 Hardware Interfaces

There are no specific hardware interface requirements.

4.3 Software Interfaces

- **Firebase SDKs:** The application will interface with Firebase Authentication, Firestore, and Storage via their respective web SDKs.
- **Google Gemini API:** The application will make HTTPS requests to the Google Gemini API (`gemini-1.5-flash-latest` model) to power all AI features.

4.4 Communications Interfaces

All communication between the client and Firebase/Gemini API will be conducted over HTTPS.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

- The application should load quickly, with initial page loads under 3 seconds on a standard broadband connection.
- Real-time updates from Firestore (e.g., new classroom posts) should appear without requiring a page refresh.
- AI-generated responses from the Study Buddy should ideally be returned to the user within 5 seconds.

5.2 Safety Requirements

Not applicable for this type of software.

5.3 Security Requirements

- User authentication is mandatory for accessing any page other than the landing and auth pages.
- Sensitive information, such as API keys and Firebase configuration, must be stored in files excluded from the public Git repository via `.gitignore`.
- The system shall enforce role-based access control, preventing students from accessing teacher-only pages and functionalities.
- The sign-up page must include a CAPTCHA to mitigate automated bot registrations.

5.4 Software Quality Attributes

- **Maintainability:** The code will be organized into a clear file structure, with CSS and JavaScript modularized into separate files for easier maintenance.
- **Portability:** As a web application, it is highly portable and can be accessed from any device with a modern web browser.
- **Usability:** The interface will be designed to be intuitive and user-friendly, requiring minimal training for both students and teachers. Consistent navigation and clear feedback (loaders, modals) are provided throughout the app.

6. Other Requirements

None.

Appendices

A. Glossary

- **SRS:** Software Requirements Specification
- **UI:** User Interface
- **API:** Application Programming Interface
- **Firestore:** The NoSQL database service provided by Firebase.
- **Gemini:** The family of generative AI models developed by Google.
- **SPA:** Single-Page Application
- **CAPTCHA:** Completely Automated Public Turing test to tell Computers and Humans Apart.

B. Analysis Models

Use case diagrams for user interactions.

C. Issues List

A list of notable issues that were resolved during development:

- **Fullscreen API Limitations:** Addressed browser security rules that prevent automatic fullscreen by implementing a user-triggered flow.
- **CSS Loading Failures:** Resolved issues with incorrect relative paths by using absolute paths and embedding CSS where necessary.
- **Asynchronous Data Loading:** Fixed race conditions where the UI would try to render before data was fetched from Firestore.
- **AI Model Errors:** Updated the Gemini model name from `gemini-pro` to `gemini-1.5-flash-latest` to resolve