Software Requirements Specification (SRS) for UpAlgo

1. Introduction

1.1 Purpose

The purpose of this document is to define the detailed requirements for "UpAlgo," an online coding preparation platform that aims to provide users with coding challenges, questions, and test cases to enhance their coding skills, similar to LeetCode.

1.2 Scope

The scope of UpAlgo includes the development of a web-based platform that offers coding challenges, question management, user authentication, and a coding environment with Python support. The platform will also feature a user-friendly interface and an admin panel for content management.

1.3 Document Conventions

- **Bold Text**: Denotes section headings.
- `Monospace Text`: Denotes code snippets, variable names, or file paths.

2. System Overview

2.1 System Description

UpAlgo is a web application built using Django (Python), JavaScript, and HTML. It allows registered users to access coding challenges, submit solutions, and view results. The system provides an admin panel for question management.

2.2 Users

UpAlgo targets the following user roles:

- **Guest**: Non-registered visitors who can access limited information.
- **Registered User**: Users with an account who can solve questions, access test cases, and track their progress.
- **Admin User**: Administrators with privileges to manage questions, users, and content.

3. Functional Requirements

3.1 User Authentication

- Users must be able to register, log in, and log out using their email or GitHub accounts through GitHub OAuth.
- User passwords must be securely stored and hashed.

3.2 Question Management
- Admin users can add, edit, and delete coding questions.
- Each question must include a title, description, input format, output format, and a set of test cases.
- Test cases should cover various scenarios and edge cases.
- Questions can be marked as "premium" for paid access.
- Questions should be categorized and searchable.

3.3 User Dashboard
- Registered users have a dashboard displaying their progress, solved questions, and statistics.
- Progress should include metrics such as the number of questions solved, success rate, and progress over time.

3.4 Question Access
- Users can browse a list of available questions, filtered by categories, tags, or difficulty levels.
- Premium questions are accessible only after a successful Stripe payment.
- Users can view detailed descriptions and test cases for each question.

3.5 Coding Environment
- A coding environment is provided to users to write, compile, and run Python code.
- Users can submit their code for evaluation against the provided test cases.
- Detailed error messages and results are displayed.
- Users can switch between code editors, including a code editor with Python syntax highlighting and code suggestions.

3.6 Payment Integration
- Stripe will be integrated as a third-party payment gateway for processing premium content payments.
- Users can make payments securely for premium question access.

4. Non-Functional Requirements

4.1 Performance
- The platform should have low latency and fast response times.
- The system should handle concurrent users efficiently.

4.2 Security
- User data, including passwords, must be securely stored and encrypted.

- Authentication through GitHub OAuth should be implemented securely.
- Payment processing via Stripe should follow security best practices.

4.3 Usability
- The user interface should be intuitive, with clear navigation and responsive design.
- Code evaluation should provide meaningful error messages.
- The platform should support multiple languages in the future.

4.4 Scalability
- The system should be designed to scale horizontally to accommodate increased user traffic.

5. Constraints

5.1 Technology Stack
- The platform will be built using Django (Python), JavaScript, and HTML.
- The system will be hosted on a cloud platform (e.g., AWS, Azure).

5.2 Supported Browsers
- The system should be compatible with modern web browsers, including Chrome, Firefox, Safari, and Edge.

6. Glossary

- **Django**: A Python web framework.
- **GitHub OAuth**: A service that allows users to log in with their GitHub accounts securely.
- **Stripe**: A third-party payment processing service.
- **HTML** Hypertext Markup Language.
- **JavaScript**: A programming language used for web development.

7. Appendix

7.1 Wireframes (if available)
- Attach wireframes or UI mockups that illustrate the expected design and layout of the platform.

7.2 Data Models (if available)
- Include data models or database schemas that outline the structure of the application's data.

7.3 Third-party Services Documentation

- Provide links to the documentation of GitHub OAuth and Stripe for integration details.