**🧾 Software Requirements Specification (SRS) Document**

**Project Title: Micro Code Review Assistant**

**Prepared By: *Jawad Ahmad, Muneeb Shafique, Dayyan Riaz, Huzaifa Muzammil***

**Supervisor: *Mam Iris***

**Institute of Data Science**

**University: *University Of Engineering and Technology, Lahore***

**Date: *17th Oct, 2025***

**1. Introduction**

**1.1 Purpose**

The purpose of this document is to define the **functional and non-functional requirements** for the development of the *Micro Code Review Assistant* — a web-based platform that allows users to write, execute, and receive AI-generated code reviews in real-time.

This document will serve as a guide for developers, testers, and stakeholders to ensure that the system is developed, tested, and maintained according to the agreed specifications.

**1.2 Scope**

The **Micro Code Review Assistant** aims to assist:

* **Students** preparing for coding interviews
* **Developers** improving code quality and logic
* **Online coding challenge participants** looking for quick, AI-assisted code reviews

The system provides:

* Online code editing and execution environment
* AI-powered review feedback
* Logical and performance insights
* Review history storage and retrieval

The system will be deployed as a **web application**, supporting code execution through **Piston/Judge0 API**, and AI feedback via **Hugging Face inference API**.

**1.3 Definitions, Acronyms, and Abbreviations**

| **Term** | **Definition** |
| --- | --- |
| AI | Artificial Intelligence |
| LLM | Large Language Model |
| API | Application Programming Interface |
| UI | User Interface |
| UX | User Experience |
| IDE | Integrated Development Environment |
| JSON | JavaScript Object Notation |
| CRUD | Create, Read, Update, Delete |

**1.4 References**

* Hugging Face Model Inference API Documentation
* Piston API Documentation (for code execution)
* Angular 17 Official Documentation
* FastAPI Documentation

**1.5 Overview**

The remainder of this document specifies the **system features**, **functional and non-functional requirements**, **system models**, and **constraints** to ensure complete and professional documentation of the *Micro Code Review Assistant*.

**2. Overall Description**

**2.1 Product Perspective**

The system is an **independent web application** that integrates:

* **Frontend:** Angular
* **Backend:** FastAPI/Flask
* **Execution Engine:** Piston/Judge0
* **AI Engine:** Hugging Face model (e.g., StarCoder, CodeT5)
* **Database:** SQLite (local) / Supabase (cloud)

The architecture follows a **client-server model** with RESTful communication between frontend and backend.

**2.2 Product Functions**

1. **Code Input and Execution**
   * Users write code using a web-based code editor.
   * Code is executed in a sandbox environment.
2. **AI Code Review**
   * AI model analyzes code logic, structure, and output.
   * Returns improvement suggestions and explanations.
3. **Submission History**
   * Stores code, output, and AI feedback in the database.
   * Allows users to revisit past reviews.
4. **Language Support**
   * Supports multiple languages (Python, C++, Java, JavaScript).
5. **Optional Authentication**
   * JWT or Firebase-based login system (optional for semester project).

**2.3 User Classes and Characteristics**

| **User Class** | **Description** | **Technical Expertise** |
| --- | --- | --- |
| Student / Interviewee | Uses app for code practice and AI feedback | Beginner to Intermediate |
| Developer | Tests logic and receives advanced feedback | Intermediate |
| Instructor | Uses it for code review demonstration | Intermediate |

**2.4 Operating Environment**

| **Component** | **Technology** | **Environment** |
| --- | --- | --- |
| Frontend | Angular 17 | Web Browser |
| Backend | FastAPI / Flask | Python 3.11+ |
| Database | SQLite / Supabase | Local or Cloud |
| AI Engine | Hugging Face Model API | Cloud |
| Execution Engine | Piston API | Cloud Sandbox |
| Deployment | Railway / Render / Vercel | Cloud-based Hosting |

**2.5 Design and Implementation Constraints**

* Execution API (Piston/Judge0) limits requests per second.
* Free-tier Hugging Face model inference may have latency.
* Hosting must remain under free limits (Railway, Render).
* Data privacy and secure code execution must be ensured.

**2.6 User Documentation**

* In-built “Help” modal for user guidance
* README.md (for developers)
* API reference endpoints documentation via Swagger (if FastAPI used)

**2.7 Assumptions and Dependencies**

* Active internet connection required.
* AI and code execution APIs must remain operational.
* Angular CLI and Node.js must be installed for frontend development.

**3. System Features**

**3.1 Feature 1: Code Editor & Execution**

**Description:**  
Allows users to write and execute code in multiple languages.

**Functional Requirements:**

* [FR-1.1] The system shall provide a syntax-highlighted code editor.
* [FR-1.2] The user shall select a programming language before execution.
* [FR-1.3] The system shall send the code to Piston API for execution.
* [FR-1.4] The execution output shall be displayed in real time.

**3.2 Feature 2: AI Review and Feedback**

**Description:**  
Provides automated code review via AI model inference.

**Functional Requirements:**

* [FR-2.1] The backend shall send the user’s code and output to AI service.
* [FR-2.2] The AI shall return structured feedback (readability, optimization, logic).
* [FR-2.3] The frontend shall display feedback alongside the code.

**3.3 Feature 3: Review History Management**

**Description:**  
Stores and retrieves users’ previous submissions.

**Functional Requirements:**

* [FR-3.1] The system shall store code, feedback, and timestamps.
* [FR-3.2] Users shall be able to view past submissions.
* [FR-3.3] The system shall allow filtering by date or language.

**3.4 Feature 4: API Sharing (Optional Advanced Feature)**

**Description:**  
Expose AI review capabilities as a RESTful API for other developers.

**Functional Requirements:**

* [FR-4.1] The backend shall expose a /ai-review endpoint.
* [FR-4.2] Third-party developers shall be able to send code and receive feedback.

**4. External Interface Requirements**

**4.1 User Interfaces**

* **Angular-based UI** with responsive layout
* **Tabs:** Editor | Feedback | History | Settings
* **Color Scheme:** Dark & Light theme switch

**4.2 Software Interfaces**

* Piston API — code execution
* Hugging Face Inference API — AI review
* SQLite/Supabase — data persistence

**4.3 Communication Interfaces**

* REST APIs over HTTPS
* JSON as data format

**5. Non-Functional Requirements**

| **Type** | **Description** |
| --- | --- |
| **Performance** | Response under 5s for AI review; 2s for code execution |
| **Scalability** | Supports up to 50 concurrent users |
| **Security** | HTTPS, CORS enabled, limited API keys |
| **Maintainability** | Modular architecture with isolated services |
| **Usability** | Intuitive UI; zero learning curve |
| **Portability** | Works on all major browsers and devices |
| **Availability** | 95% uptime (limited by hosting tier) |

**6. System Architecture Overview**

**Layers:**

1. Presentation (Angular)
2. Application (FastAPI)
3. Intelligence (Hugging Face AI Engine)
4. Data (SQLite/Supabase)

**7. Future Enhancements**

* Add **Real-Time Collaboration** (multiple users on same code)
* Add **Voice Feedback using TTS**
* Integrate **AI Code Generation** for incomplete code
* Advanced **Plagiarism Checker** for submissions

**8. Appendices**

* A: API Endpoint Reference
* B: Data Schema Diagram
* C: Model Evaluation Notes
* D: Deployment Instructions

✅ **Professional Summary:**  
This SRS follows IEEE 830:1998 standard format used by software engineers and project reviewers in academia and industry. It will serve as your **final formal documentation** for submission or defense.

