Software Requirements Specifications

Voice Coding

Project Code:

Internal Advisor:
Mr. Mudassir Ali Zaidi

Project Manager:
Dr. Muhammad Ilyas

Project Team:

Alina Ahmed	BSCS51F22S044	Team Lead
Laiba Humayun	BSCS51F22S031	Team Member
Maryum Javed	BSCS51F22S009	Team Member

Submission Date:
October 20, 2025

_____
Project Manager's Signature

Document Information

Category	Information
Customer	University of Sargodha
Project	Voice Coding
Document	Requirement Specifications
Document Version	1.0
Identifier	
Status	Draft
Author(s)	Alina Ahmed Laiba Humayun Maryum Javed
Approver(s)	Professor Dr. Muhammad Ilyas
Issue Date	October 20, 2025
Document Location	
Distribution	1. Professor Mudassir Ali Zaidi 2. Professor Dr. Muhammad Ilyas

Definition of Terms, Acronyms and Abbreviations

Term/Acronym	Description
API	Application Programming Interface, a service for communication between systems
Web Speech API	Browser feature for speech recognition
Monaco Editor	Code editor component from VS Code
Paiza.io / JDoodle	Free online C++ compiler APIs
RSI	Repetitive Strain Injury
IDE	Integrated Development Environment
HTML / CSS / JS	Frontend web languages
SRS	Software Requirements Specification

Table of Contents

1. Introduction

1.1 Purpose of Document

This Software Requirements Specification (SRS) describes the functional and non-functional requirements for the Voice Coder web application. It serves as a guide for stakeholders, developers, and evaluators, defining the project's functionality, limitations, and design goals.

1.2 Project Overview

Voice Coder is a web-based application that enables users to write C++ programs using voice commands. The system translates spoken instructions into syntactically correct C++ code, compiles it in real time, and displays the output. The objective is to improve accessibility for users with physical disabilities or RSI and promote hands-free programming.

1.3 Scope

The system will:

- *Convert voice commands into valid C++ code*
- *Support basic to intermediate C++ constructs*
- *Provide real-time cloud-based compilation*
- *Display program outputs and errors*
- *Operate on modern browsers without installation*

The system will not:

- *Support programming languages other than C++*
- *Offer advanced IDE features such as debugging or version control*
- *Work offline or in languages other than English*
- *Handle complex C++ metaprogramming*

2. Overall System Description

It describes the environment, in which the system will be developed and used, the anticipated users of the system and the known constraints, assumptions and dependencies.

2.1 User characteristics

Primary Users: *Students with disabilities, programmers with RSI, and computer science educators.* ***Secondary Users:*** *Accessibility researchers and developers interested in voice-based programming.*

2.2 Operating environment

- ***Hardware:*** *PC, Laptop, or Tablet with microphone*
- ***Operating System:*** *Windows 10+, macOS 10.14+, Linux distributions*
- ***Browser:*** *Chrome 70+, Firefox 75+, Edge 79+, Safari 13+*
- ***Dependencies:*** *Web Speech API, Paiza.io API, Monaco Editor*

2.3 System constraints

- *Requires browser support for Web Speech API and internet connection*
- *Limited to C++ language features supported by Paiza.io*
- *Needs microphone, at least 2 GB RAM, quiet environment, and clear English speech*

3. External Interface Requirements

This section is intended to specify any requirements that ensure that our system will connect properly to external components.

3.1 Hardware Interfaces

The system is fully web-based and only requires a microphone and internet connection.

3.2 Software Interfaces

- *Frontend: HTML, CSS, JavaScript*
- *Speech Recognition: Web Speech API*
- *Compilation: Paiza.io or JDoodle API*
- *Code Editor: Monaco Editor*

3.3 Communications Interfaces

The system will communicate securely with compiler APIs via HTTPS, ensuring that voice and code data are processed temporarily and not stored permanently.

4. Functional Requirements

4.1 Voice Command Recognition

The system must capture voice input and convert it to text using the Web Speech API.

4.2 Command Parsing and Code Generation

The application will interpret recognized text into predefined C++ templates.
Example:
"*Include Iostream****"*** *→ #include <iostream>*
"*For loop from i to 10****"*** *→ for(int i=0; i<10; i++){}*

4.2 Code Display and Editing

The recognized code must appear in the Monaco Editor, allowing manual edits or corrections before execution.

4.5 Error Handling

The system must display clear messages for invalid or unrecognized commands.

4.6 User Interaction and Accessibility

The system must have accessible buttons labeled "Start Recording," "Stop," and "Run Code."

5. Non-functional Requirements

5.1 Performance Requirements

- *The web-based system must process spoken commands and generate corresponding C++ code within 2–4 seconds of input under standard internet conditions.*
- *The speech recognition module (Web Speech API) should maintain a minimum accuracy rate of 90–95% in transcribing programming-related keywords and control structures.*
- *The code parser should efficiently handle sequential voice inputs and correctly format constructs like loops, conditionals, and functions with minimal syntax errors.*
- *The integration with cloud compiler APIs (JDoodle/Paiza.io) should allow code execution and output retrieval within 5 seconds per request.*
- *The system should sustain performance under moderate load, supporting multiple concurrent users without API timeout or browser lag.*

5.2 Safety Requirements

- *The platform will execute all user-generated C++ code within a secure cloud sandbox environment provided by JDoodle or Paiza.io, preventing unauthorized access to the host system.*
- *No locally executed code or external file access will be permitted, reducing the risk of data loss or malware execution.*
- *The system should automatically log and handle errors gracefully, ensuring that invalid or incomplete voice commands do not crash the application or produce undefined code structures.*
- *Temporary data and cached voice recordings will be cleared automatically after each session to prevent data leakage or misuse.*
- *User code and results will remain session-specific and non-persistent, ensuring that no residual data from one user can interfere with another's session.*

5.3 Security Requirements

- *All communication between the client browser and external APIs (Web Speech API, JDoodle, Paiza.io) must use secure HTTPS/TLS encryption to protect data integrity.*
- *The system should conform to OWASP Top 10 web security guidelines to prevent common vulnerabilities such as Cross-Site Scripting (XSS), injection attacks, and cross-origin requests.*

- *User data — including speech transcripts and code — must not be stored or shared with third parties without consent.*
- *Any authentication tokens or API keys used to access cloud compilers must be secured and hidden from the client side.*
- *The platform should perform input sanitization to ensure only safe code constructs are sent for compilation and execution.*

5.4 User Documentation

Comprehensive user documentation will be provided, including:

- *A web-based user manual describing all features (voice input, code generation, compilation, error handling).*
- *Voice command reference sheet listing supported C++ commands and phrases.*
- *Documentation will be updated iteratively to match each milestone and final implementation phase.*

6. Assumptions and Dependencies

- *The system assumes the user has access to a stable internet connection (minimum 2 Mbps) for cloud-based speech recognition and compilation.*
- *It is assumed that the user's browser supports Web Speech API (Google Chrome, Microsoft Edge, or Firefox latest version).*
- *The Web Speech API, JDoodle, and Paiza.io services are expected to remain operational and free to use throughout the project duration.*
- *The user is expected to speak commands in clear English, following the predefined syntax pattern designed for accurate parsing.*
- *The system depends on the availability and reliability of external APIs for speech recognition and code compilation; any downtime or API policy change may affect performance.*
- *The parser logic and C++ syntax mapping are designed according to C++17 standards; future changes in the C++ language may require codebase updates.*
- *The environment assumes modern web browsers and standard hardware specifications (8 GB RAM, Intel i5 or equivalent).*
- *It is assumed that the voice recognition accuracy might vary depending on background noise, accent, or pronunciation; users are expected to test and adjust microphone settings for best results.*
- *The development process relies on iterative testing, meaning partial functionality will be refined progressively until final integration and optimization.*

7. References

- Mozilla Developer Network (MDN). *Web Speech API Documentation.* https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API

- Monaco Editor – Microsoft Open Source Project. https://microsoft.github.io/monaco-editor

- Paiza.io API Documentation. https://paiza.io/en

- JDoodle API Documentation. https://www.jdoodle.com/compiler-api

- IEEE Std 830-1998. *IEEE Recommended Practice for Software Requirements Specifications.* IEEE Computer Society.

- Pressman, R. S. (2019). *Software Engineering: A Practitioner's Approach.* McGraw-Hill.