Voice Coding

(Project Proposal)

Project Code

VC-WA01

Project Advisor

Mr. Mudassar Ali Zaidi

Project Manager

Dr. Muhammad Ilyas

Project Team

Alina Ahmed (S044)	Team Lead
Maryum Javed(S009)	Team Member
Laiba Humayun(S031)	Team Member

Submission Date

October 6, 2025

Table of Contents

Abstract

Traditional programming requires extensive keyboard use, creating barriers for individuals with physical disabilities or repetitive strain injuries who cannot type efficiently [1], [4]. We came with the idea to develop a voice- controlled web application that converts spoken commands into valid C++ code, implements real-time compilation, and provides accessible programming capabilities without keyboard dependency [1], [2].

Implementation Steps:

- Design web interface with integrated code editor [3]
- Implement voice recognition using Web Speech API [1]
- Develop C++-specific command parser [5]
- Integrate cloud compilation services [2], [8]
- Conduct usability testing [4]

Expected Benefits

Provides inclusive programming education tools, reduces physical barrier in software Development, and offers hands-free coding alternative for all programmers [4].

Background and Justification

Existing Work:

Current programming environments (VS Code, CodeBlocks) rely entirely on keyboard input [3]. Some voice programming tools exist (Serenade, Talon Voice, Speech 2 code) but require paid subscriptions, complex setups, or local installations, limiting accessibility for students and individuals with disabilities [10], [11].

Enhancement Approach:

Our web-based solution requires no installation or payment, using free APIs and browser technologies [1], [2]. We focus specifically on C++ programming education with optimized voice commands for common programming constructs [5], [6], making it immediately usable in academic environments.

Justification:

This project addresses the critical gap in accessible programming tools by providing a zero-cost, web-based platform that enables voice-controlled C++ coding [4]. It enhances existing approaches through simplified access, educational focus, and elimination of financial and technical barriers [4], [5].

Project Methodology

The project will follow an iterative and incremental development model, ensuring each feature is built, tested, and refined step-by-step [12].

Phase 1 - Interface Design:
The development begins with designing a simple and accessible web interface using HTML, CSS, and JavaScript. The layout includes a code editor (Monaco Editor), microphone control buttons, and an output console for displaying program results [3].

Phase 2 - Speech Recognition Integration:
The Web Speech API will be implemented to convert user voice into text. When a user speaks commands such as "include iostream" or "for loop," the system will capture and process them for code generation [1].

Phase 3 - Command Parsing and Code Generation:
A custom JavaScript parser will interpret the recognized text and map it to corresponding C++ syntax structures (e.g., "if x greater than y" → if (x > y) {}) [5]. This parser ensures accurate and consistent translation between voice input and code output.

Phase 4 - Code Compilation and Output:
The generated code will be compiled using Paiza.io or JDoodle free online compiler APIs. These services execute the C++ code in the cloud and return the output [2], [8], [9].

Phase 5 - Testing and Improvement:
Each feature will be tested individually to ensure proper performance. User testing will focus on speech accuracy, code correctness, and response time [4].

Project Scope

In Scope:

- Real-time voice-to-code conversion [1]
- Integration with free cloud compiler APIs [2], [8]
- Syntax highlighting and command mapping [3], [5]
- Browser-based execution (no installation required)
- Accessibility for individuals with physical disabilities [4]

Out of Scope:

- Support for multiple programming languages (limited to C++)
- Offline voice recognition Complex AI-based natural language processing

High level Project Plan

Milestone 1 - Foundation & Setup
Objectives: Project kickoff, repository creation, basic web scaffold, Monaco Editor integration, and initial Web Speech API prototype [1], [3].

Milestone 2 - Voice Recording & Basic Commands
Objectives: Implement voice capture, transcription to editor, and basic command parsing (symbols and keywords) [1], [5].

Milestone 3 - Advanced Command Processing
Objectives: Support control structures, function declarations, templates; complete voice-driven code generation for simple programs [5], [6].

Milestone 4 - Integration & End-to-End Flow
Objectives: Integrate free compilation API (Paiza.io), "Compile & Run" workflow, error handling, UI polish [2], [8].

Milestone 5 - User Experience Enhancements
Objectives: Command help, syntax highlighting, undo/redo, tutorial/demo mode [4].

Milestone 6 - Advanced Features
Objectives: Context-aware parsing, auto-completion, command history, customization.

Milestone 7 - Testing & Optimization
Objectives: Accents and browser testing, performance tuning, bug fixes, stress tests [9].

Milestone 8 - Documentation & Final Presentation
Objectives: Comprehensive docs, user manual, demo video, final report, slides, rehearsal.

Time Allocation by Activity:

- Planning & Design: 10%
- Frontend Development (UI, Editor, Voice UI): 25%
- Voice Processing & Command Parsing: 20%
- Integration & Compilation (Paiza.io): 15%
- UX Enhancements & Accessibility: 10%
- Testing, Debugging, & Optimization: 15%
- Documentation & Presentation: 5%

Hardware/Environment:

- Modern browser (Chrome/Edge/Firefox) for testing

Success Metrics:

- Voice-to-code for basic C++, end-to-end compile/run
- Functional UI with editor and output display

Final Product:

- Coverage of core C++ constructs via voice
- High accuracy voice recognition and robust error handling
- Complete documentation and demo-ready presentation

References

[1] Mozilla Developer Network (MDN), "Web Speech API."
https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API

[2] JDoodle, "JDoodle API for Online Compiler and Code Execution."
https://docs.jdoodle.com/compiler-api

[3] Microsoft, “Monaco Editor Documentation.”
https://microsoft.github.io/monaco-editor/docs.html

[4] World Wide Web Consortium (W3C), “Web Accessibility Initiative (WAI).”
https://www.w3.org/WAI/

[5] cppreference.com, “C++ Language Reference.”
https://cppreference.com

[6] LearnCpp.com, “Learn C++.”
https://www.learncpp.com

[7] Open Web Application Security Project (OWASP), “OWASP Top Ten.”
https://owasp.org/www-project-top-ten/

[8] Paiza.IO, “Security & Sandboxing.”
https://paiza.io

[9] JDoodle, “Security & Sandboxing.”
https://www.jdoodle.com

[10] Serenade.
https://serenade.ai/

[11] Speech-to-Code GitHub Repository.
https://github.com/pedrooaugusto/speech-to-code

[12] Atlassian, “Project Management.”
https://www.atlassian.com/project-management