

VOICE CODING

Project Code:

Project Advisor:

Mr. Mudassar Ali Zaidi

Project Manager:

Professor Dr. Muhammad Ilyas

Project Team:

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

Submission Date:

October 6 , 2025

Table of Contents

Abstract.....	3
Background and justification.....	4
Methodology.....	5
Scope.....	6
High level project plan.....	7
References	9

1. Abstract

Traditional programming requires extensive keyboard use, creating barriers for individuals with physical disabilities or repetitive strain injuries who cannot type efficiently. 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.

Implementation Steps:

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

Expected Benefits: Provides inclusive programming education tools, reduces physical barriers in software development, and offers hands-free coding alternative for all programmers.

2. Background and Justification

Existing Work: Current programming environments (VS Code, CodeBlocks) rely entirely on keyboard input. 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.

Enhancement Approach: Our web-based solution requires no installation or payment, using free APIs and browser technologies. We focus specifically on C++ programming education with optimized voice commands for common programming constructs, 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. It enhances existing approaches through simplified access, educational focus, and elimination of financial and technical barriers.

3. Methodology

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

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.

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.

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) {}`). 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.

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. Scope

In Scope:

- Real-time voice-to-code conversion
- Integration with free cloud compiler APIs
- Syntax highlighting and command mapping
- Browser-based execution (no installation required)
- Accessibility for individuals with physical disabilities

Out of Scope:

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

5. 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.

Milestone 2 — Voice Recording & Basic Commands

Objectives: Implement voice capture, transcription to editor, and basic command parsing (symbols and keywords).

Milestone 3 — Advanced Command Processing

Objectives: Support control structures, function declarations, templates; complete voice-driven code generation for simple programs.

Milestone 4 — Integration & End-to-End Flow

Objectives: Integrate free compilation API (Paiza.io), “Compile & Run” workflow, error handling, UI polish.

Milestone 5 — User Experience Enhancements

Objectives: Command help, syntax highlighting, undo/redo, tutorial/demo mode.

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.

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).
“<https://developer.mozilla.org/en->
2. JDoodle Documentation. “JDoodle API for Online Compiler and Code Execution.”<https://docs.jdoodle.com/compiler-api>
3. Microsoft Visual Studio Code. “Monaco Editor Documentation.”: <https://microsoft.github.io/monaco-editor/docs.html>
3. World Wide Web Consortium. (n.d.). Web Accessibility Initiative (WAI). <https://www.w3.org/WAI/>
4. Atlassian. (n.d.). Project management.
<https://www.atlassian.com/project-management>
C++ language constructs
5. cppreference.com. (n.d.). C++ language reference.
<https://cppreference.com>
6. LearnCpp.com. (n.d.). Learn C++.
<https://www.learncpp.com>
7. Open Web Application Security Project. (n.d.).
OWASP Top Ten. <https://owasp.org/www-project-top-ten/>
8. Paiza.IO. (n.d.). Security & sandboxing (vendor docs).
<https://paiza.io>
9. JDoodle. (n.d.). Security & sandboxing (vendor docs).
<https://www.jdoodle.com>
10. <https://serenade.ai/>
11. <https://github.com/pedrooaugusto/speech-to-code>