Department of Computer Science and Engineering

Project Proposal

Course Code: CSE314
Course Title: Compiler Design Lab

Submitted To

Ms. Tamanna Sultana

Lecturer,

Department of Computer Science and Engineering

Daffodil International University

Submitted By

Team Name: Team CLI (Clueless Legend of IT)

Student Name	Student ID
MD. Abdullah Khan	232-15-425
MD. Sazzad Islam	232-15-835
MD. Saif Ahmed	232-15-636
MD. Abdur Rahman Nayeem	232-15-143

Section: 65_L2

Group: 03

Date of Submission: 24th November, 2025

CLI

Custom Language Interpreter

A Visual Interface for a Custom Programming Language Using Flex, Bison, and Web Technologies

Prepared by: MD. Abdullah Khan

Date: 24th November, 2025

Table of Contents

Executive Summary

This project aims to develop a custom programming language named **CLI (Custom Language Interpreter)** along with a fully functional **web-based IDE** where users can write, compile, and execute code directly from a browser. The backend compiler will be built using **Flex** for lexical analysis and **Bison** for syntax parsing. The frontend interface will be developed using **HTML, CSS, and JavaScript** for a clean and user-friendly experience.

The goal of this project is to create a simple, interactive, and educational platform that demonstrates how compilers work internally and makes learning compiler design easier and more enjoyable for students.

Project Objectives

- ✓ Develop a working compiler for the CLI language using **Flex and Bison**.
- ✓ Design a modern and intuitive web interface for writing and running CLI code.
- ✓ Enable seamless data flow between the frontend and backend using JavaScript **fetch()** API.
- ✓ Make a beginner-friendly platform for understanding compiler fundamentals.
- ✓ Build a professional-quality project suitable for academic submission and portfolio use.

Problem Statement

Compiler design is often taught theoretically, leaving students without practical exposure to real compiler behavior. Most compiler implementations run through command-line interfaces, which can be difficult for beginners. To solve this, the proposed system provides a **visual, interactive, browser-based compiler environment** for a custom language—allowing students to write and run CLI code easily.

This makes compiler learning more interactive, intuitive, and accessible.

Project Scope

➢ **Included Scope**

- A functional programming language: **CLI (Custom Language Interpreter)**
- A compiler created using **Flex (lexer)** and **Bison (parser)**
- A responsive web interface for writing programs
- A backend system (implemented using Python Flask) that executes CLI code and sends output to the browser

- Error handling (lexical, syntax, and runtime)

➢ **Excluded Scope**

- Debugging features or syntax highlighting (future enhancements)
- Mobile application version
- File input/output system

Tools & Technologies

Part	Tools Used
Compiler	Flex (Lexer), Bison (Parser), C Programming
Frontend	HTML, CSS, JavaScript
Integration	JavaScript fetch() API
Execution	CLI-based CLI compiler **(Custom Language Interpreter)**
Backend Server	Python (Flask framework)

Project Modules

1. CLI Compiler Module

- Performs lexical analysis using Flex to generate tokens
- Uses Bison parser to validate grammar and structure
- Executes parsed code using C
- Displays output or detailed errors

2. Web Interface Module

- Text editor area for writing CLI code
- **Run** button to compile and execute code
- Output box for results or errors
- Clean, responsive, minimalistic design

3. Integration Module

- JavaScript fetch() sends user code to backend (Flask server in Python)
- Backend (Flask + C compiler engine) runs the CLI compiler and returns execution results
- Output is shown directly on the webpage in real time

4. User Interface Features

- Simple textarea-based code editor
- Execution button
- Output display panel
- Works on both desktop and mobile devices

Background Knowledge & Practical Foundations

A. C Programming & String Processing

- Numeric input processing (range operations)
- Manual string length calculation
- Reversing and concatenating strings using loops
- Counting vowels, words, lines, and articles
- Extracting prepositions from a text
- Caesar cipher shifting (+3)
- Email parsing (letters, digits, symbols)
- Identifying single-line and multi-line comments
- Word frequency calculation
- Regular Expression acceptance program in C

B. Compiler Theory Fundamentals

- Token, Pattern, Lexeme
- Types of compiler errors (lexical, syntax, semantic, logical, runtime)
- String recognition using regular expressions (abb, a* b+, a+)

C. Flex (Lexical Analysis)

- Keyword recognition (int, float, if, else, while)
- Identifier, number, float detection
- Operator and delimiter recognition
- Block comment removal
- String literal extraction
- Token counting & classification

D. Bison / YACC (Parsing)

- %union and typed tokens
- Passing identifiers via yylval.str
- Grammar rules for variable declarations

- Parser action code for printing variables
- Error handling using yyerror()

E. Lex–YACC Integration

- Source → Lex → YACC → Execution flow
- Token passing and grammar validation
- Semantic action execution

Specific Project Outcomes

- A functional **lexer** capable of recognizing CLI keywords, identifiers, numbers, operators, and comments.
- A complete **parser** capable of processing declarations, assignments, expressions, loops, and conditions.
- A **symbol table** storing variables and values.
- Execution engine that evaluates expressions and control structures.
- A fully working **web-based IDE** for writing and running CLI code.
- Meaningful **lexical, syntax, and runtime error reporting**.

Timeline

Week	Task	Output
1	Build MiniLang compiler (Flex/Bison)	Working compiler
2	Design frontend interface	Complete UI
3	Integrate frontend with backend	Browser-based code execution
4	Testing & Documentation	Final project with demo

Expected Results

- A fully functional web-based compiler for CLI language
- Simple and interactive learning tool for compiler design
- A polished academic and portfolio-ready project
- Base structure for adding future enhancements

Conclusion

This project combines the concepts of compiler design with modern web development to create an interactive coding environment for students. The CLI Web IDE helps users understand the internal workflow of compilers while providing a smooth and engaging interface. As both an academic submission and a portfolio project, it demonstrates strong technical skills and opens the door for future enhancements such as syntax highlighting, debugging, and advanced language features.

Our Project Diagram: