“Bangla Programming Language Interpreter”

Submitted By

Student Name	**Student ID**
Foysal Mahamud Fahim	232-15-334
Sharmin Akter Jame	232-15-380
Estiake Ahmed	232-15-430

LAB PROJECT REPORT

This Report Presented in Partial Fulfillment of the course

CSE314 Compiler Design Lab in the Department of Computer Science and Engineering

DAFFODIL INTERNATIONAL UNIVERSITY

Dhaka, Bangladesh

14/12/2025

DECLARATION

We hereby declare that this lab project has been done by us under the supervision of **Zarif Mahmud**, **Lecturer**, Department of Computer Science and Engineering, Daffodil International University. We also declare that neither this project nor any part of this project has been submitted elsewhere as lab projects.

Submitted To:

Zarif Mahmud
Lecturer
Department of Computer Science and Engineering
Daffodil International University

Submitted by

Foysal Mahamud Fahim ID:232-15-334 Dept. of CSE, DIU	
Sharmin Akter Jame ID:232-15-380 Dept. of CSE, DIU	Estiake Ahmed ID:232-15-430 Dept. of CSE, DIU

COURSE & PROGRAM OUTCOME

The following course have course outcomes as following:

Table 1: Course Outcome Statements

CO’s	Statements
CO1	**Define** and **Relate** classes, objects, members of the class, and relationships among them needed for solving specific problems
CO2	**Formulate** knowledge of object-oriented programming and Java in problem solving
CO3	**Analyze** Unified Modeling Language (UML) models to **Present** a specific problem
CO4	**Develop** solutions for real-world complex problems **applying** OOP concepts while evaluating their effectiveness based on industry standards.

Table 2: Mapping of CO, PO, Blooms, KP and CEP

CO	PO	Blooms	KP	CEP	CO
CO1	PO1	C1, C2	KP3	EP1, EP3	CO1
CO2	PO2	C2	KP3	EP1, EP3	CO2
CO3	PO3	C4, A1	KP3	EP1, EP2	CO3
CO4	PO3	C3, C6, A3, P3	KP4	EP1, EP3	CO4

The mapping justification of this table is provided in section **4.3.1**, **4.3.2** and **4.3.3**.

Table of Contents

Chapter 1

Introduction

This chapter gives an overview of the Compiler Design Lab project, including its objectives, scope, and significance. It explains the goals the project aims to achieve.

1.1 Introduction

Programming is traditionally taught using English-based syntaxes, which can create an additional learning barrier for beginners who are more comfortable with their native language. To address this challenge, this project focuses on the design and implementation of a Bangla-based Script Interpreter. The interpreter allows users to write programs using Bengali keywords such as **সংখ্যা** for variable declaration, **লিখ** for output, **নাও** for input, **যদি** for conditional statements, **যখন** and **যাবত** for loop control, and **শেষ** to terminate control blocks.

1.2 Motivation

The primary motivation behind this project is to reduce the language barrier in learning programming concepts. Many beginners face difficulties understanding programming due to unfamiliar English-based syntax. By introducing a Bangla script interpreter, this project aims to make programming more intuitive and learner-friendly. Additionally, the project serves as a hands-on application of compiler design theories learned in the course, strengthening practical understanding through real implementation.

1.3 Objectives

The main objectives of the project are:

- To design and develop a functional Bangla Script Interpreter.
- To support basic programming constructs such as variables, input/output, conditions, and loops.
- To implement symbol table management for variable storage and retrieval.
- To execute Bangla-based scripts correctly and efficiently.
- To provide a simple menu-driven interface for running scripts and accessing help.
- To enhance understanding of compiler design concepts through practical implementation.

1.4 Feasibility Study

The project is technically feasible as it is implemented using the C programming language, which provides efficient control over memory and execution. Required tools such as GCC compiler and standard C libraries are easily available. The system runs on common operating systems like Windows and Linux. From an academic perspective, the project scope is manageable within the lab timeframe and aligns well with the course objectives. No specialized hardware is required, making the project economically feasible as well.

1.5 Gap Analysis

Existing programming tools and compilers are primarily based on English syntax, which can be challenging for beginners who are more comfortable with Bangla. There is a noticeable gap in tools that support native-language-based programming education. This project addresses that gap by introducing Bangla keywords and commands while still maintaining core programming logic. However, compared to full-scale compilers, the interpreter has limited optimization and language features, highlighting areas for future enhancement.

1.6 Project Outcome

The outcome of this project includes:

- A working Bangla Script Interpreter capable of executing Bangla programs.
- Support for variable declaration and assignment using **সংখ্যা** and other commands.
- Handling of user input via **নাও** and output via **লিখ**.
- Conditional execution using **যদি** statements.
- Loop control using **যখন** (while loops) and **যাবত** (for loops) with **শেষ** to end blocks.
- Execution of arithmetic operations and assignments correctly.
- A user-friendly, menu-driven interface with help options.
- Demonstration of compiler design concepts such as symbol table management, parsing, and execution control.

Chapter 2

System Architecture

This chapter describes the overall system design and architecture of the Compiler Design Lab project. It outlines the major components, their interactions, and the technologies used to build an efficient and scalable database system.

2.1 Requirement Analysis & Design Specification

This section focuses on analyzing the requirements of the Bangla Script Interpreter and specifying its design to ensure efficient development and usability.

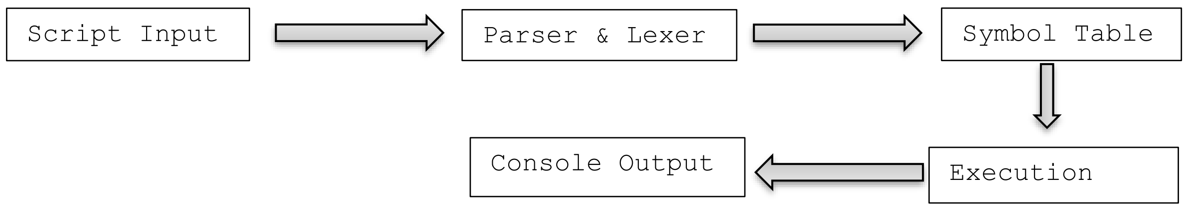
2.1.1 Requirements

The requirements of the system include:

- Support for Bangla keywords such as **সংখ্যা, লিখ, নাও, যদি, নইলে, যখন, যাবত, শেষ**.
- Execution of arithmetic operations and variable assignments.
- Input and output handling via console.
- Conditional execution using **যদি** statements.
- Loop execution using **যখন** and **যাবত** with **শেষ** to terminate blocks.
- User-friendly menu-driven interface with options to run scripts, open editor, and access help.
- Platform compatibility with Windows and Linux.
- Lightweight implementation using C programming for educational purposes.

2.1.2 System Components Diagram

Since the interpreter does not use a database, the ERD is not applicable. Instead, a simplified system components diagram can represent the main modules:



This diagram shows the flow from script input, parsing, variable management (symbol table), execution, to output display.

Language Sketch: (Keywords & Semantics)

Bangla Keyword	Syntax / Semantics	Comment
সংখ্যা	সংখ্যা name = int	Declare a new variable or update an existing variable.
লিখ	লিখ "text" or লিখ var	Print a string or variable value to the console.
যদি	যদি var op int ... নইলে ... শেষ	Conditional block; executes code if the condition is true, else executes alternative block.
যখন	যখন var op int করো ... শেষ	Loop construct; executes the code repeatedly while the condition holds.
নাও	নাও var	Take input from the user and store it in the value

2.1.3 GUI Design

The system employs a console-based menu interface that includes:

- **Language Selection Menu:** Choose between Bangla, C (coming soon), and C++ (coming soon).
- **Main Menu:** Options to run script files, open script editor, view help, or exit.
- **Interactive Prompts:** Console prompts for **নাও** (input) and informative messages for **লিখ** (output).
- **Colorful UI Elements:** ANSI escape codes to highlight banners, messages, and errors for better readability.

2.2 Use of Modern Tools

The project is developed using modern development tools and practices:

- **GCC Compiler:** For compiling C code on multiple platforms.
- **Text Editor:** Notepad (Windows) or Nano (Linux) for editing scripts.
- **UTF-8 Encoding:** To support Bangla characters correctly.
- **Version Control (Optional):** Git can be used to manage code versions and collaborative development.
- **ANSI Escape Codes:** For enhanced console UI with colored output and banners.

Chapter 3

Implementation and Results

This chapter details the step-by-step implementation process of the Compiler Design Lab project. It presents the development phases, key features, and the results obtained from testing and evaluation.

3.1 Implementation

The Bangla Script Interpreter is implemented in C language with a focus on simplicity and educational clarity. Key implementation details include:

- **Input Parsing:** Reads script lines from a file and trims unnecessary spaces.
- **Symbol Table:** Uses in-memory arrays to store variable names and values for fast retrieval and assignment.
- **Command Handling:** Supports Bangla keywords like **নাও, সংখ্যা, লিখ, যদি, নইলে, যাবত, যখন, শেষ**.
- **Control Structures:** Implements conditional statements and loops by parsing operators and evaluating expressions.
- **Menu System:** Provides a user-friendly console interface to select language, run scripts, open editor, or view help.
- **Error Handling:** Detects missing script files and displays appropriate error messages.
- **UTF-8 Support:** Ensures proper display of Bangla characters in console.

Module	**Screenshot (Code)**

File Handling	`FILE *fp = fopen("script.txt", "r");`
	`if(!fp) {`
	`    printf("Error opening file\n");`
	`}`
	`// Read lines`
	`char line[512];`
	`while(fgets(line, sizeof(line), fp)) {`
	`    // Process each line`
	`}`
	`fclose(fp);`

Variable Declaration (সংখ্যা)	`if (strncmp(line, "সংখ্যা", strlen("সংখ্যা")) == 0)`
	`{`
	`    char name[50]; int value;`
	`    if (sscanf(line + strlen("সংখ্যা"), " %s = %d", name, &value)`
	`    {`
	`        setVarValue(name, value);`
	`    }`
	`}`

Printing (লিখ)	`else if (strncmp(line, "লিখ", strlen("লিখ")) == 0) {`
	`    char *p = line + strlen("লিখ");`
	`    while (*p) {`
	`        if (*p == '"') {`
	`            p++;`
	`            while (*p && *p != '"') {`
	`                putchar(*p);`
	`                p++;`
	`            }`
	`            if (*p == '"') p++;`
	`            printf(" ");`
	`        } else if (*p == ' ') {`
	`            p++;`
	`        } else {`
	`            char word[50]; int i=0;`
	`            while (*p && *p != ' ' && *p != '"') word[i++] = *p++;`
	`            word[i] = 0;`
	`            if (strlen(word) > 0) {`
	`                int val;`
	`                if (getVarValue(word, &val)) printf("%d ", val);`
	`                else printf("%s ", word);`
	`            }`
	`        }`
	`    }`
	`    printf("\n");`
	`}`

Conditional (যদি)	`else if(strstr(line,"যদি")!=NULL && strstr(line,"লিখ")!=NULL){`
	`    char msg[200], op_str[3];`
	`    sscanf(line,"যদি %s %s %s লিখ \"%[^\"]\"",a,op_str,b,msg);`
	`    int v1=getVarValue(name,var,count,a);`
	`    int v2=getVarValue(name,var,count,b);`
	`    int cond=0;`
	`    if(strcmp(op_str,"<")==0) cond=v1<v2;`
	`    else if(strcmp(op_str,">")==0) cond=v1>v2;`
	`    else if(strcmp(op_str,"==")==0) cond=v1==v2;`
	`    else if(strcmp(op_str,"!=")==0) cond=v1!=v2;`
	`    else if(strcmp(op_str,"<=")==0) cond=v1<=v2;`
	`    else if(strcmp(op_str,">=")==0) cond=v1>=v2;`
	`    if(cond) printf("%s\n",msg);`
	`}`

Loop (যখন)	`else if(strncmp(line,"যাবত",4)==0){`
	`    char loopVar[50], startStr[50], condVar[50], cmp[3], endStr[50];`

	```
	    sscanf(line,"যাবত %s = %s থেকে %s %s %s পর্যন্ত",
	        loopVar, startStr, condVar, cmp, endStr);

	    // Get start value
	    int startVal;
	    if(isdigit(startStr[0]) || (startStr[0] == '-' && isdigit(startStr[1]))) {
	        startVal = atoi(startStr);
	    } else {
	        startVal = getVarValue(name,var,count,startStr);
	    }
	}
	```

Scanf (নাও)	
	```
	if (sscanf(line, "নাও %s", vname) == 1) {
	    int temp;
	    printf("%s এর মান দিন: ", vname);
	    scanf("%d", &temp);
	    getchar();

	    int idx = getVarIndex(name, count, vname);
	    if (idx == -1) {
	        strcpy(name[count], vname);
	        var[count] = temp;
	        count++;
	    } else {
	        var[idx] = temp;
	    }
	}
	```

3.2 Output

The system produces the following outputs:

- **Console Output:** Executes the Bangla script commands and displays results using **লিখ**.
- **Interactive Prompts:** Requests user input with **নাও** commands.
- **Error Messages:** Displays messages in red for missing files or invalid commands.
- **Script Execution Logs:** Shows the running script filename and successful execution completion message.

Example Output:

```
====================================
    BD SCRIPT INTERPRETER v4.1
====================================
Running script: fact.txt
n এর মান দিন: 5
5! = 120

Script finished successfully.
```

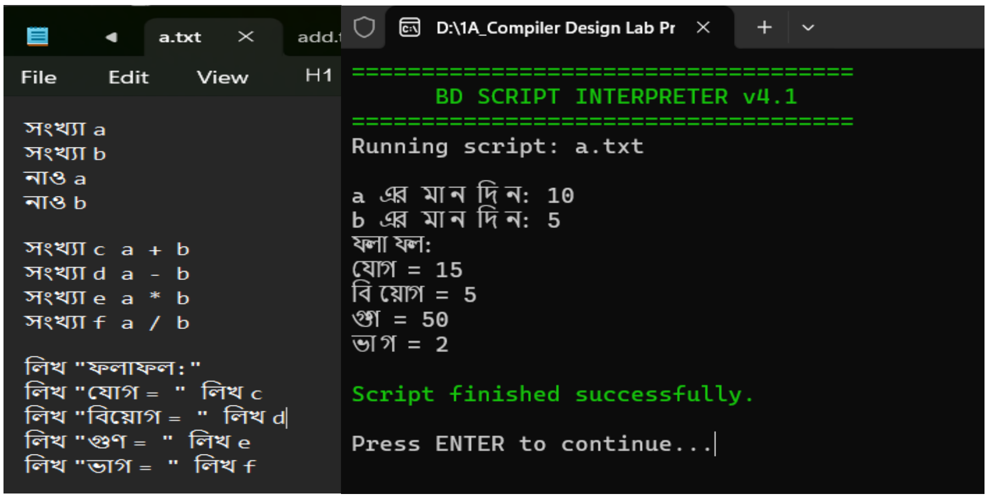
Github Link: https://github.com/fahim-j/Bangla-Compiler-Compiler-Design-Lab-Project-

Live Demo Link: https://banglaprograminglanguageinterpreter.netlify.app/

3.3 Results and Discussion

Input Program

Code: **Output:**



```
সংখ্যা a
সংখ্যা b
নাও a
নাও b

সংখ্যা c  a  +  b
সংখ্যা d  a  -  b
সংখ্যা e  a  *  b
সংখ্যা f  a  /  b

লিখ "ফলাফল:"
লিখ "যোগ =  "  লিখ c
লিখ "বিয়োগ =  "  লিখ d
লিখ "গুণ =  "  লিখ e
লিখ "ভাগ =  "  লিখ f
```

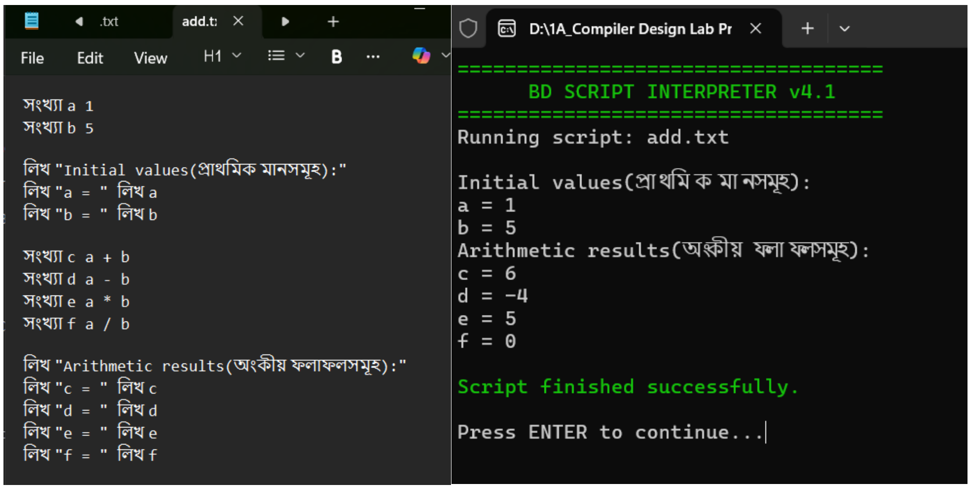
```
======================================
      BD SCRIPT INTERPRETER v4.1
======================================
Running script: a.txt

a এর মান দিন: 10
b এর মান দিন: 5
ফলা ফল:
যোগ = 15
বি য়োগ = 5
গুণ = 50
ভাগ = 2

Script finished successfully.

Press ENTER to continue...
```

Fig-1



```
সংখ্যা a  1
সংখ্যা b  5

লিখ "Initial values(প্রাথমিক মানসমূহ):"
লিখ "a = "  লিখ a
লিখ "b = "  লিখ b

সংখ্যা c  a + b
সংখ্যা d  a - b
সংখ্যা e  a * b
সংখ্যা f  a / b

লিখ "Arithmetic results(অংকীয় ফলাফলসমূহ):"
লিখ "c = "  লিখ c
লিখ "d = "  লিখ d
লিখ "e = "  লিখ e
লিখ "f = "  লিখ f
```

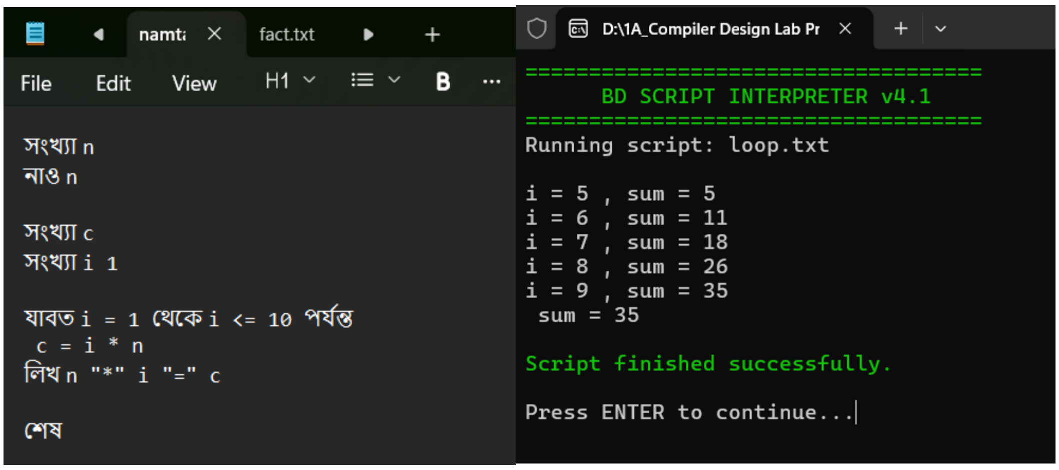
```
======================================
      BD SCRIPT INTERPRETER v4.1
======================================
Running script: add.txt

Initial values(প্রা থমি ক মা নসমূহ):
a = 1
b = 5
Arithmetic results(অঙ্কীয় ফলা ফলসমূহ):
c = 6
d = -4
e = 5
f = 0

Script finished successfully.

Press ENTER to continue...
```

Fig-2



```
সংখ্যা n
নাও n

সংখ্যা c
সংখ্যা i 1

যাবত i = 1 থেকে i <= 10 পর্যন্ত
 c = i * n
লিখ n "*" i "=" c

শেষ
```

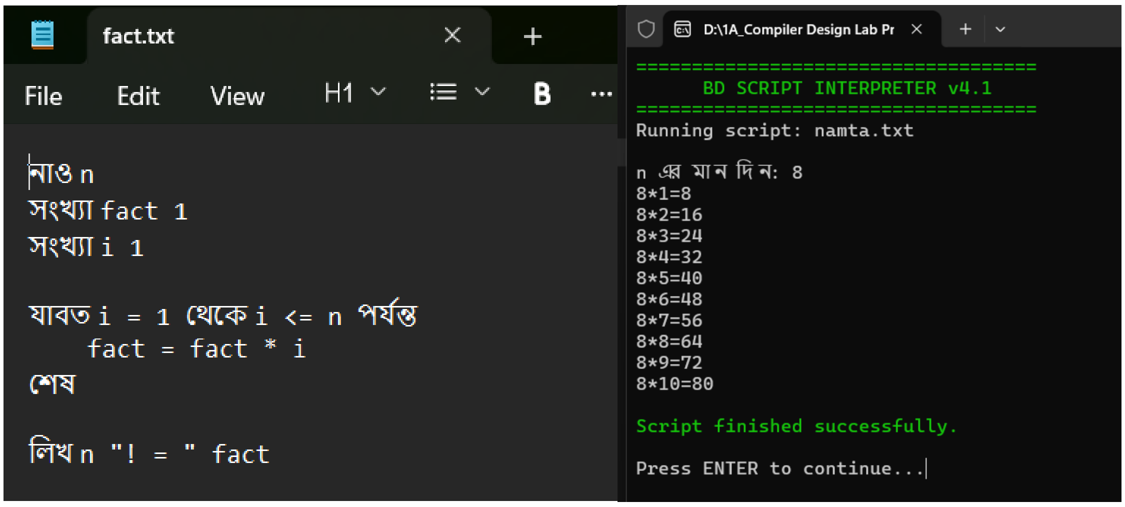
```
=====================================
      BD SCRIPT INTERPRETER v4.1
=====================================
Running script: loop.txt

i = 5 , sum = 5
i = 6 , sum = 11
i = 7 , sum = 18
i = 8 , sum = 26
i = 9 , sum = 35
 sum = 35

Script finished successfully.

Press ENTER to continue...
```

Fig-3



```
নাও n
সংখ্যা fact 1
সংখ্যা i 1

যাবত i = 1 থেকে i <= n পর্যন্ত
    fact = fact * i
শেষ

লিখ n "! = " fact
```

```
=====================================
      BD SCRIPT INTERPRETER v4.1
=====================================
Running script: namta.txt

n এর মান দিন: 8
8*1=8
8*2=16
8*3=24
8*4=32
8*5=40
8*6=48
8*7=56
8*8=64
8*9=72
8*10=80

Script finished successfully.

Press ENTER to continue...
```

Fig-4

Discussion

- The project demonstrates the practical implementation of a **Bangla-based scripting language interpreter**.
- It provides a hands-on experience in **compiler design concepts** such as parsing, symbol table management, and execution control.
- Limitations include **no persistent database**, **limited error detection**, and **console-only interface**.
- Future enhancements can include GUI development, advanced data structures, and support for additional programming constructs.
- Overall, the interpreter successfully bridges the gap between **language accessibility** and **learning programming concepts in Bangla**

Chapter 4
Engineering Standards and Mapping

This chapter discusses the engineering standards followed throughout the project and maps the project activities to the course and program outcomes. It highlights how the project meets academic and professional requirements

4.1 Impact on Society, Environment and Sustainability

This section describes the overall effects of the Bangla Script Interpreter project on society, environment, ethics, and long-term sustainability.

4.1.1 Impact on Life

- Makes programming **accessible to Bangla-speaking students and learners**.
- Helps beginners understand basic programming concepts without the barrier of English syntax.
- Encourages educational growth and computational thinking in local language.

4.1.2 Impact on Society & Environment

- **Society:** Promotes digital literacy and programming skills among students who prefer Bangla.
- **Environment:** The project is software-based and lightweight; no physical resources are consumed apart from normal computer usage.
- Enhances awareness of **local language computing**, encouraging software solutions in Bangla.

4.1.3 Ethical Aspects

- Promotes inclusivity by providing access to programming in native language.
- No data collection or privacy concerns as the interpreter works offline.
- Encourages responsible coding practices among learners.

4.1.4 Sustainability Plan

- Maintainable **codebase** in C language, easy to update and enhance.
- Can be extended with GUI, additional Bangla keywords, or support for advanced programming constructs.
- Open-source potential allows community contributions and long-term usability.
- Minimal system requirements ensure it can run on a wide range of computers, ensuring accessibility and continued use

4.2 Complex Engineering Problem

4.2.1 Mapping of Program Outcome

In this section, the mapping of the Bangla Programming Language Interpreter project with targeted Program Outcomes (POs) is provided.

Table 4.1: Justification of Program Outcomes

POs	**Justification of Mapping (Project Perspective)**
PO3	The Bangla Programming Language Interpreter project involves designing and implementing a script interpreter with parsing, symbol table management, and execution control, addressing complex problem-solving and system design challenges (PO3).
PO5	The Bangla Programming Language Interpreter incorporates modern programming concepts, C language implementation, and software development tools, ensuring application of contemporary practices in real-world scenarios (PO5).
PO9	Through collaborative project work, all members were engaged in teamwork, task allocation, and documentation (PO9).
PO10	The Bangla Programming Language Interpreter project requires presenting the developed interpreter, preparing technical documentation, and communicating design ideas clearly (PO10).

4.2.2 Complex Problem Solving

Table 4.2: Mapping with complex problem solving.

EP1 Dept of Knowledge	**EP2** Range of Conflicting Requirements	**EP3** Depth of Analysis	**EP4** Familiarity of Issues	**EP5** Extent of Applicable Codes	**EP6** Extent Of Stakeholder Involvement	**EP7** Inter-dependence
✓	✓		✓			

EP1: Bangla Programming Language Interpreter project requires designing and implementing an interpreter that parses scripts, handles variables, executes control structures, and manages I/O operations. These tasks involve tackling complex software engineering challenges, enhancing problem-solving skills.

EP2: We utilize C language and standard programming libraries to implement the interpreter efficiently. Applying these fundamental tools in a practical environment strengthens our ability to use established engineering tools effectively.

EP4: We critically evaluate the interpreter by testing scripts, debugging errors, and optimizing code execution. This ongoing evaluation helps refine the solution to meet performance and accuracy standards.

4.2.3 Complex Engineering Activities

Table 4.2: Mapping with complex engineering activities.

EA1	EA2	EA3	EA4	EA5
✓	✓	✓	✓	✓

EA1: In the Bangla Programming Language Interpreter project, we systematically gather project requirements, design parsing logic, implement symbol tables, and manage script execution. This preparation helps organize technical information logically, ensuring clear understanding of the project structure and functionality.

EA2: We use console-based visual outputs and step-by-step demonstrations to explain the working of the interpreter. This helps users understand script execution and language features, improving communication skills.

EA3: Teamwork is essential for dividing tasks such as parser development, arithmetic and control-flow implementation, and testing. Collaboration ensures efficient delivery and contribution from all team members.

EA4: Critical thinking and problem-solving are applied to debug complex script scenarios, handle invalid commands, and optimize interpreter performance, strengthening engineering judgment.

EA5: Professional ethics are followed while designing the interpreter. Proper code documentation, clarity in communication, and adherence to software development best practices are maintained, promoting accountability and professionalism.

4.3 Project Management and Team Work

Team Structure and Roles:

Date	Task/Activity	Assigned Team Member	Status	Comments/Remarks
November 10, 2025	**Project Planning and Research**	All Team Members	Completed	Initial research, objectives, planning.
November 18, 2025	**System Design**	Sharmin Akter Jame	Completed	Schema successfully drawn
November 24, 2025	**GUI design**	Estiake Ahmed	Completed	Console-based GUI design completed.
November 25, 2025	**Code Development**	All Team Members	Completed	Started coding the interpreter.
December 05, 2025	**Final Code Selection and Edits**	Foysal Mahamud Fahim	Completed	Fixed all debug issues.
December 06, 2025	**Manual Testing**	Sharmin Akter Jame	Completed	Final checks and testing.
December 10, 2025	**Final Report Writing and Edits**	All Team Members	Completed	Write report and make edits.
December 15, 2025	**Final Review and Submission**	All Team Members.	Completed	Final checks and submission of project.

Time and Cost:

Overall, time needed is **30-35 days**, and the total cost of this project is **15k**.

Chapter 5

Conclusion

5.1 Summary

The Bangla Programming Language Interpreter project successfully demonstrates the implementation of a simple scripting language using Bangla keywords. The interpreter handles variable declarations, arithmetic operations, control structures like loops and conditionals, and input/output operations. It provides a user-friendly console interface, allowing Bangla-speaking learners to write and execute programs in their native language. The project enhances understanding of compiler concepts such as parsing, symbol table management, and execution flow.

5.2 Limitation

Despite its advantages, the system has certain limitations:

- Currently, the interpreter is **console-based** with no graphical user interface.
- **Limited error handling**; complex syntax errors may not be fully detected.
- Does not support advanced programming constructs like functions, arrays, or user-defined procedures.
- Script execution is **linear and in-memory**, with no persistent storage or file-based variable saving.
- Only supports integer variables; floating-point and string manipulations are not implemented.

5.3 Future Work

The system can be enhanced in the future by:

- Develop a **graphical user interface (GUI)** to enhance usability.
- Extend the interpreter to support **advanced programming constructs**, including functions, arrays, and user-defined procedures.
- Add **robust error handling** and debugging features.
- Include **floating-point numbers, string operations, and more Bangla keywords**.
- Enable **persistent storage** to save variable states and program outputs.
- Convert the project into an **educational IDE** for Bangla-speaking students with tutorials, sample programs, and interactive learning.

References

Books:

1. John R. Levine, Tony Mason, and Doug Brown. *Lex & Yacc*. O'Reilly Media, 1992.
2. Alfred V. Aho, Monica S. Lam, Ravi Sethi, and Jeffrey D. Ullman. *Compilers: Principles, Techniques, and Tools (2nd Edition)*. Pearson, 2006.
3. Keith D. Cooper and Linda Torczon. *Engineering a Compiler*. Morgan Kaufmann, 2011.

Online Resources:

- Official GCC Documentation: https://gcc.gnu.org/onlinedocs/
- Tutorialspoint: Compiler Design: https://www.tutorialspoint.com/compiler_design/index.htm
- GeeksforGeeks: Compiler Design Concepts: https://www.geeksforgeeks.org/compiler-design-tutorials/
- YouTube Channels: Compiler design tutorials, parser and interpreter examples, and C language implementation guides.
- ChatGPT and AI-based coding assistants for explanations, debugging, and example code generation.