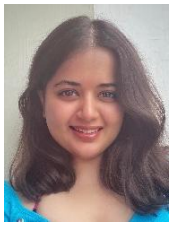
Graphic Era deemed to be University
DEHRADUN

PROJECT AND TEAM INFORMATION

Project Title

NL2C- Bridging Human Logic to C code

Student/Team Information

Team Name: Team # (Mentor needs to assign)	The Code Rangers
Team member 1 (Team Lead) (Solanki, Aakriti : 220221054 : aakritisolanki.188@gmail.com)	
Team member 2 (Nitya: 220221996: nityabudhraja555@gmail.com)	

Team member 3 (Pandey, Achyut : 220211170 : achyutpandey018@gmail.com)	
Team member 4 (Sharma, Saransh : 22021535 : saranshsharma669@gmail.com)	

PROJECT PROGRESS DESCRIPTION

Project Abstract

NL2C (Natural Language to C) is an educational compiler-like tool that converts user-defined pseudocode written in plain English into valid C code. It aims to help beginners transition from natural logic to structured programming by combining compiler techniques (like lexical analysis, parsing, and code generation) with NLP. The user enters logic in a loose pseudocode format, and the system translates it into syntactically correct C code. This tool enhances the learning experience for novice programmers by reducing syntactic complexity and promoting algorithmic thinking.

Updated Project Approach and Architecture

NL2C follows a classic compiler design approach with three main phases: lexical analysis, parsing, and code generation. Users provide pseudo code or natural language input, which is first tokenized using a custom lexer built with SLY (Python Lex-Yacc). The parser, also using SLY, maps these tokens to grammar rules for control structures, loops, assignments, and I/O operations. These are then translated into corresponding C language syntax. The system supports a flexible vocabulary, enabling alternative keywords like "define" or "allot" via NLP.

NLP and Error Handling:

The system integrates SpaCy for natural language similarity matching, expanding user input flexibility. It also includes basic error feedback mechanisms, such as identifying undeclared variables during parsing.

Libraries and Tools Used

- SLY: For lexer and parser construction
- SpaCy: For NLP similarity and keyword mapping
- Pandas, Regex, Rich: For data handling, pattern extraction, and better output formatting
- AStyle: For formatting generated C code
- Streamlit: For creating a web-based GUI interface

Communication Protocols:

When used with a GUI (e.g., Streamlit), input/output communication happens over HTTP via form submission. In the CLI version, I/O is file or terminal-based, making it platform-independent and easy to test.

Tasks Completed

Task Completed	Team Member
Lexer and parser implementation using SLY	Saransh and Achyut
NLP similarity integration via Spacy	Nitya
Grammar and Syntax rule for pseudocode	Aakriti and Achyut
Code Formatting via Astyle	Aakriti
Streamlit GUI prototype	Nitya
Undeclared variable highlighting	Aakriti
Test programs like prime checker implemented	Entire Team

Challenges/Roadblocks

1. SpaCy Model Compatibility: Model version mismatches and large downloads caused setup issues.
2. Pydantic Version Issues: Opyrator was incompatible with newer Pydantic v2.x, so we shifted to Streamlit.
3. Token Conflicts: Grammar led to shift/reduce conflicts (~164) which required careful resolution.
4. AStyle on Windows: Executable path issues required local file use instead of global install.
5. Testing NLP Mapping: Ensuring accurate keyword similarity for diverse inputs was time-intensive.

   We handled these by using specific dependency versions and thorough step-by-step debugging.

Tasks Pending

Task Pending	Team Member (to complete the task)
Robust Error Message Rendering in GUI	Nitya and Aakriti
Export Feature for Generated C code	Saransh and Aakriti
Dockerization for Deployment	Achyut

Project Outcome/Deliverables

- Functional NLP-to-C compiler with lexer, parser, and code generator
- Command-line interface for translation

- Streamlit-based GUI
- Undeclared variable detection
- Sample pseudocode programs and their C equivalents
- Full documentation and presentation

Progress Overview

~80% of the core functionality is complete. Parser, code generator, and NLP module are functional. GUI is in place but needs polish. Error feedback for undeclared variables works; future work will expand this to structural errors. We are on track.

Codebase Information

GitHub Repo: https://github.com/testgithubrit11189/compiler_project
Branch: main
Important Commits:

- Converter-func.py: Initial token and parser logic
- Nlp-suggestion.py: SpaCy integration
- GUI-interface: code using app.py
- error-feedback:will be included in the converter-func.py

Testing and Validation Status

Test Type	Status (Pass/Fail)	Notes
Prime Number pseudocode	Pass	Generates correct C code
Streamlit Interface	pass	UI working,but will be enhanced
Declaring Invalid Variable	In Progress	Will throw undeclared variable error

Deliverables Progress

Deliverable	Status
Lexer and Parser	Completed
NLP Integration	Completed
Undeclared Variable Handling	Completed
Streamlit Interface	In Progress
Final Documentation & Dockerization	Pending