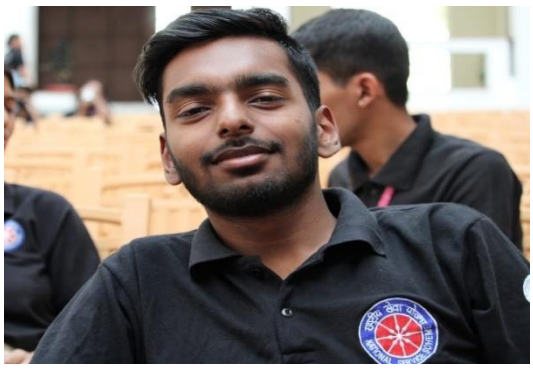
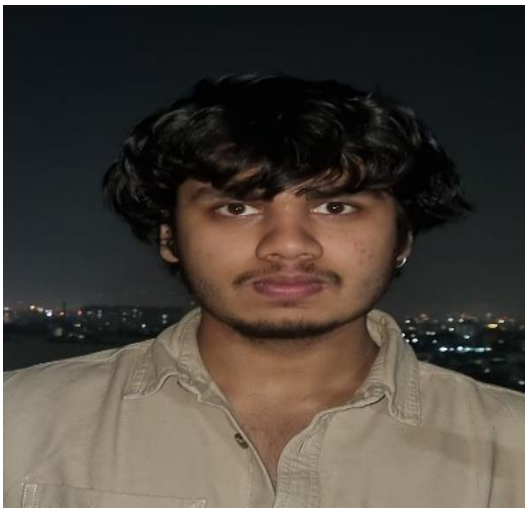
Graphic Era Hill University
DEHRADUN • BHIMTAL • HALDWANI

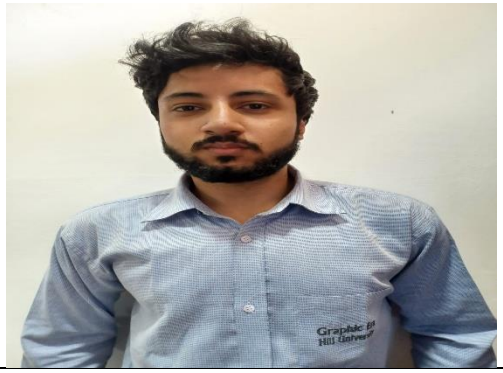
PROJECT AND TEAM INFORMATION

Project Title

Lexical Analysis using Python

Student/Team Information

Team Name	MEGARUSHERS
Team Member 1 (Team Lead) Samarth Agarwal Student ID: - 22011896 samarth2404agarwal@gmail.com	
Team Member 2 Kunwardeep Singh Student ID: - 22011787 kunwar2104@gmail.com	

Team Member 3 Lakshaydeep Chaudhary Student ID: - 2219016 lakshay71003@gmail.com	

PROJECT PROGRESS DESCRIPTION

Project Abstract

The Lexical Analyzer is a Python-based educational tool designed to tokenize source code into meaningful tokens such as keywords, identifiers, operators, literals, constants, and comments. It simulates the front-end functionality of a compiler by generating a symbol table, constant table, and parsed table. The tool also performs error detection for invalid characters, nested comments, and unbalanced braces. The project has been completed as a standalone command-line application along with a basic web interface using Flask for tokenization. It serves as a practical guide to understanding compiler design concepts.

Updated Project Approach and Architecture

Backend:

- ✔ Lexer implemented in Python using regular expressions (re module).
- ✔ Maintains Symbol Table, Constant Table, Parsed Table.
- ✔ Advanced error handling: detection of nested comments, invalid tokens, unmatched braces.

Frontend:

- ✔ Flask API integrated for communication between backend and frontend.
- ✔ Basic HTML/CSS/JS Web Interface allowing code input and display of tokenized output.

Technologies Used: Python 3.x, Flask, HTML, CSS, JavaScript, Regex, File I/O.

Tasks Completed

Task Completed	Team Member
Regex-based Tokenizer	Samarth Agarwal
Symbol & Constant Tables	Kunwardeep Singh
Error Handling Logic	Lakshaydeep Chaudhary
Flask API Integration	Kunwardeep Singh
Web Interface Development	Lakshaydeep Chaudhary
Comprehensive Testing	Samarth Agarwal

Challenges/Roadblocks

- ✔ Complex Regular Expressions: Resolved via refining patterns and priorities.
- ✔ Nested Comments Detection: Solved using counter and flag-based logic.
- ✔ Symbol Table Duplication: Prevented via pre-insertion checks.
- ✔ Flask API Integration: Completed by linking backend analyzer with frontend via POST requests.

Tasks Pending

None. All tasks successfully completed.

Project Outcome/Deliverables

- ✔ Fully functional Lexical Analyzer supporting tokenization of C-like code.
- ✔ SymbolTable.txt, ConstantTable.txt, ParsedTable.txt generated correctly.
- ✔ Error detection and handling for invalid inputs completed.
- ✔ Flask-based API integrated.
- ✔ Web-based frontend implemented (HTML/CSS/JS).
- ✔ All test cases verified successfully.
- ✔ Final documentation completed.

Progress Overview

- ☑ 100% Complete
- ✔ Lexer, Tables, Error Handling, Flask API, Frontend – all completed as planned.
- ✔ Testing and documentation done.
- ✔ No pending or delayed work.

Codebase Information

Repository: https://github.com/MegarusherSamarth/PBL/tree/main/Compiler%20Design

Testing and Validation Status

Test Type	Status	Notes
Tokenization Test	Pass	Keywords, Identifiers, Operators identified correctly.
Error Handling	Pass	Invalid characters, nested comments, unbalanced braces detected.
Symbol Table	Pass	Lexemes recorded without duplication.
Flask API Test	Pass	Backend-frontend integration successful.
Web Interface	Pass	Input/output flow correct and user-friendly.

Deliverables Progress

Deliverable	Status
Lexer Core (Python)	Completed
Symbol/Constant Table Generation	Completed
Error Detection/Handling	Completed
Flask API Integration	Completed
Web Frontend (HTML/CSS/JS)	Completed
Testing & Validation	Completed
Documentation	Completed