# PROJECT AND TEAM INFORMATION

# **Project Title:**

**Team Name** 

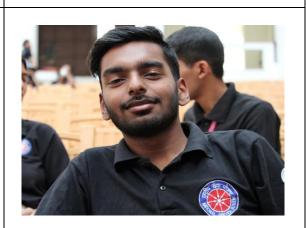
Lexical Analyzer using Python

# Student / Team Information:

# Team Member 1 (Team Lead)

Samarth Agarwal Student ID: - 22011896 samarth2404agarwal@gmail.com

# **MEGARUSHERS**



# 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 tool designed to tokenize source code into meaningful tokens (e.g., keywords, identifiers, operators) as part of a compiler's front-end. It supports error handling, symbol/constant tables, and comment detection, serving as an educational tool for understanding compiler design principles.

## **Updated Project Approach and Architecture:**

#### Backend:

- Core Lexer: Implemented in Python using regex (re module) for tokenization.
- Tables: Symbol table, constants table, and parsed table for tracking lexemes.
- **Error Handling**: Detects invalid characters, nested comments, and unbalanced braces. **Frontend** (*Planned*):
- **Web Interface**: HTML/CSS/JavaScript + Flask API for user interaction (future milestone). **Technologies**: Python, regex, file I/O, modular OOP design.

## Tasks Completed:

Task Completed	Team Member	
Regex-based tokenizer	Samarth Agarwal	
Symbol/constant tables	Kunwardeep Singh	
Error handling (comments/braces)	Lakshaydeep Chaudhary	

## Challenges / Roadblocks:

- 1. **Complex Regex Patterns**: Debugging token misclassification. *Solution*: Refined regex groups and priorities.
- 2. **Nested Comments**: Tracking /\* \*/ pairs. *Solution*: Counter (nc) and flag-based logic.
- 3. Symbol Table Duplicates: Avoided redundant entries via pre-check.

# Tasks Pending:

Task Pending	Team Member
Flask API integration	Kunwardeep Singh
Web interface (HTML/CSS/JS)	Lakshaydeep Chaudhary
Comprehensive test cases	Samarth Agarwal

# **Project Outcomes / Deliverables:**

- 1. Functional Lexer: Tokenizes C-like code (keywords, identifiers, etc.).
- 2. **Reports**: Generates symbolTable.txt, constantTable.txt, parsedTable.txt.
- 3. Error Handling: Logs invalid chars, unclosed comments, unbalanced braces.
- 4. **Documentation**: Code comments and usage guide (in progress).

## **Project Overview:**

Ahead of Schedule: Core lexer, tables, error handling.

- On Track: File I/O, reports.
- Behind Schedule: Frontend (pending Phase 3).

## **Codebase Information:**

- Repository: https://github.com/MegarusherSamarth/PBL/tree/main/Compiler%20Design
- Branch: main.
- **Key Commits**: Regex patterns, symbol table, error handling.

# **Testing and Validation Status:**

Test Type	Status	Notes
Tokenization	Pass	Validates keywords, IDs, etc.
Error Handling	Pass	Catches invalid
		chars/comments.
Symbol Table	Pass	Tracks lexemes accurately.

# **Deliverables Progress:**

- Completed: Lexer core, tables, error handling.
- In Progress: Documentation, test cases.
- Pending: Frontend, Flask API.