

Analysis of Programming Languages (CIT4004)

Semester 1 – 2025/2026

Draft Student Group Project (20%)

Secure Policy Language (SPL)

Lecturer: David W. White (dwwwhite@utech.edu.jm)

Date Given: Week of October 13, 2025

Due Date: Week of November 11, 2025

Design a new programming language that addresses the critical need for granular, human-readable access control policies which are often used in cloud infrastructure or enterprise systems. Your compiler should successfully parse a Secure Policy Language and interface with a large language (LLM) model like Microsoft's Copilot, OpenAI's ChatGPT or Google's Gemini, etc., to scan the compiled policy set for potential security risks. Your new Secure Policy programming language should demonstrate the characteristics of a good programming language that you have studied in this class.

This project offers rich complexity. Your group must master the mechanics of lexical analysis and parsing (compiler basics), design an intuitive language (syntax/semantics), and apply machine learning techniques (AI component) to solve a real meaningful problem. The analysis of existing programming languages such as Python, Java, and C++ SQL, and specialized simulation languages will be essential for informing your own design choices in this project.

This project requires your group to develop a new programming language to perform arithmetic expressions with correct operator precedence (PEMDAS/BODMAS) using the principles you have learned in this class. Your group will build a compiler or interpreter for your secure policy language, which will allow users to define who can access, modify, or delete resources (files, databases, APIs) within an organization, similar to Amazon's AWS IAM policies or Microsoft's Azure Role-Based Access Control (RBAC) and Azure Policy services. Your language will employ AI to scan the parsed policy set for potential security risks, policies granting access to too many resources, identify logical overlaps or contradictions (e.g., "Allow read if Monday" and "Deny read if Monday") and rules that inadvertently grants for example Developer access to DB_Finance delete privileges, providing the line number and a risk score.

The syntax of your secure policy language should be attribute-based and declarative, focused on subjects, objects, actions, and environment conditions. The semantics of your secure policy language should allow Boolean logic evaluation, policy inheritance, and the principle of least privilege. The compiler/interpreter for your secure policy language must perform lexical, syntactic and semantic analysis, and must incorporate the results of the semantic analysis into the core logic of your compiler/interpreter and allow the resulting target code to be executed.

Your programming language must feature the characteristics of a good programming language that you studied in class. The grammar of your language can be specified in regular CFG, BNF, EBNF, or PEG formats. You are free to use any development language of your choice, and you may also use compiler development tools such as LEX/YACC (generates C code), FLEX/BISON (generates C/C++ code), JLEX/CUP (generates Java code), Jack

(generates Java code), PLY (generates Python code), or similar tools. Special marks will be awarded for projects deployed on the cloud.

You may use Microsoft's Azure OpenAI Service, OpenAI's ChatGPT, Google's Gemini or a similar LLM to provide additional information for your language in line with the university's AI use policy. You may also use Microsoft's Azure, Amazon's AWS, or similar platform to deploy your language on the cloud. Note that you have free student access to certain elements of the Microsoft Azure platform once you log in using your UTech credentials.

Organize yourselves into groups of three (3) to five (5) persons per group, and produce a project report and a project implementation, both of which will be graded and will represent the Individual Assignment and Assignment 1 grade on your portal respectively.

Required

To complete this project successfully, you should:

- Produce a project report containing the components listed below in the grading scheme.
- Develop an executable that performs lexical, syntactic and semantic analysis on input source code written in your programming language
- Provide users with an interface to your programming language
- Generate executable target code
- Allow your programming language to be accessed and/or deployed on the cloud
- Upload your project report, source code and working application code to the project assignment space on the course portal
- Make a 10-minute online presentation on your project to the class in the allotted tutorial time, involving all members of the project team and with cameras on

At a minimum, your completed secure policy language project must allow the following:

Feature	Example Code	Compiler/Interpreter Task
Subjects/Roles	ROLE Admin {can: *} USER JaneDoe {role: Developer}	Define hierarchical roles and map users to permissions.
Objects/Resources	RESOURCE DB_Finance {path: /data/financial}	Define the assets being protected, often with wildcards (*).
Permissions	ALLOW action: read, write ON resource: DB_Finance IF (time.hour > 9 AND time.hour < 17)	Enforce access rules based on defined conditions (time, IP, device).
Denial	DENY action: delete ON resource: /data/* IF (user.role == Guest)	Implement explicit denial rules (Deny overrides Allow).

Grading Scheme

Project Report (50 marks – 10%)

- Paradigm the language you developed belongs to (1/50 marks)
- Explaining whether your language is general purpose or domain specific (1/50 marks)
- Explaining whether your language is low level or high level (1/50 marks)
- Correct grammar for the language you developed (10/50 marks)
- Complete parse tree/AST for a sample program in your language (10/50 marks)
- Full list of tokens for the language you developed (5/50 marks)
- Regular expressions you used to recognize all the tokens for the language you developed (10/50 marks)
- Demonstration of scope and binding in sample code written in your programming language (5/50)
- Details on the programming language you used to develop your compiler (2/50 marks)
- The characteristics of a good programming language (from those you studied in class) that are evident in your designed programming language, and examples of how these characteristics affect the readability, writability and reliability of your designed programming language (5/50 marks)

The Application (50 marks – 10%)

- Integration with LLM to explain steps being executed (5/50 marks)
- Correctly perform lexical analysis and tokenization (5/50 marks)
- Correctly perform syntax analysis and parse tree/AST generation (7/50 marks)
- Correctly perform semantic analysis (7/50 marks)
- Target code runs and produces expected result (8/50 marks)
- Appropriate user interface and feedback to user (4/50 marks)
- Adequate error handling (7/50 marks)
- Effective cloud deployment (7/50 marks)

Important Note

Your completed project must run, you must upload your project report and application source code to the online course portal provided on UTech Online, and you must conduct the 10-minute online presentation in your tutorial class to receive a project grade. Place the names and id numbers of all your group members in the project documentation and code. Projects which do not run will not receive a passing grade. No individual projects will be

accepted, nor will projects be accepted by email. Plagiarism is considered as a very serious offense by the University and will be penalized as outlined “Academic Misconduct” section of the Student Handbook.

In the project report and during the group presentation, each group member must state which substantial portion of the project that group member worked on (this can't be just the documentation alone), and must explain that section. Group members must be present in-person in the tutorial class during the presentation to receive a grade, and no video recordings of the presenters will be accepted.

Useful Compiler Generation and LLM Resources

Lex and YACC primer/HOWTO by Bert Hubert

<https://tldp.org/HOWTO/Lex-YACC-HOWTO.html>

Lex and Yacc: A Brisk Tutorial by Saumya K. Debray

<https://www2.cs.arizona.edu/~debray/Teaching/CSc453/DOCS/tutorial-large.pdf>

PLY (Python Lex-Yacc) by David M. Beazley

<https://www.dabeaz.com/ply/ply.html>

Write text parsers with yacc and lex by Martin Brown

<https://developer.ibm.com/technologies/systems/tutorials/au-lexyacc/>

Using JFlex and CUP to implement a compiler

<https://www.cs.auckland.ac.nz/courses/compsci330s1c/lectures/330ChaptersPDF/Chapt1.pdf>

What is ANTLR (ANother Tool for Language Recognition)? by Terence Parr

<https://www.antlr.org/>

Syntax: a language agnostic parser generator by Dmitry Soshnikov

<https://dmitrysoshnikov.medium.com/syntax-language-agnostic-parser-generator-bd24468d7cf>

The Design of a Full Computer Language

<https://www.cs.auckland.ac.nz/courses/compsci330s1c/lectures/330ChaptersPDF/Chapt9.pdf>

How can I access the ChatGPT API?

<https://help.openai.com/en/articles/7039783-how-can-i-access-the-chatgpt-api>

Google Deepmind and Google Gemini

<https://deepmind.google/technologies/gemini/>

Microsoft Azure

<https://azure.microsoft.com/en-us/>