Graphic Era Hill University
DEHRADUN • BHIMTAL • HALDWANI

PROJECT AND TEAM INFORMATION

Project Title

Mini Compiler: A Web-Based Custom Language Compiler with React & Flask

Student/Team Information

Team Name:	CodeMasters
Team member 1 (Team Lead)	Chandra, Vinod – 22011515 itsvinod14@gmail.com 
Team member 2	Rana, Rahul – 220113260 rahulrana89546@gmail.com 

Team member 3	Chauhan, Divyansh – 22011456 divyanshchauhan349@gmail.com 

PROJECT PROGRESS DESCRIPTION

Project Abstract

The goal of this project is to design and build a Mini Compiler capable of translating source code into intermediate representation using dynamically provided grammar. This compiler allows users to input their own grammar rules and source code through a user-friendly web interface. The frontend, built using React and styled with CSS, includes features such as a Monaco Editor, error output, token streams, and result display panels. The backend is built using Flask in Python and uses the PLY library for performing lexical analysis and syntax parsing. The compiler is designed to handle dynamic grammar definitions, generate tokens, detect syntax errors, and produce intermediate representations such as three-address code. The project emphasizes modularity, extensibility, and real-time feedback, offering an educational and practical tool for understanding compiler design.

Updated Project Approach and Architecture

The compiler consists of a React frontend and a Flask backend. Users interact with the Monaco Editor to input both grammar and source code. This input is sent via a REST API (/compile) to the backend. On the backend, Python Lex-Yacc (PLY) is used to perform lexical analysis and parsing dynamically based on user-defined rules. The results (tokens, syntax errors, intermediate code) are returned to the frontend and displayed in real-time. No external database is used; all processing is done in-memory. The frontend is styled using plain CSS, with folders organized for maintainability. The backend has modular Python scripts for lexing, parsing, and code generation.

Tasks Completed

Task Completed	Team Member
Frontend setup with React and Monaco Editor	Member 2 (Rahul)
Dynamic grammar input support	Member 3 (Divyansh)
API integration with Flask backend (/compile)	Member 1 (Vinod)
Lexical analyzer using PLY	Member 1 (Vinod)
Syntax parser based on dynamic grammar input	Member 2 (Divyansh)
Intermediate code generation logic started	Member 3 (Rahul)

Challenges/Roadblocks

One of the main challenges is handling dynamic grammar parsing—since most compilers use predefined grammar, dynamically building a parser from user input introduces complex issues in token precedence, rule conflicts, and error recovery. Another challenge is real-time error reporting and visualization, especially for syntax trees. To address these, we are improving the backend error-handling routines and considering incorporating visual libraries for parse trees. Additionally, as no database is used, maintaining session state purely in-memory requires careful design to avoid data conflicts between multiple requests.

Tasks Pending

Task Pending	Team Member (to complete the task)
Parse tree visualization	Member 1 (Vinod)
Semantic analysis module	Member 1 (Vinod)
Styling and polishing UI	Member 2 (Rahul)
Report writing and documentation	Member 3 (Divyansh)

Project Outcome/Deliverables

1. A working mini compiler with UI for grammar and code input
2. Dynamic lexical and syntax analyzer based on user-defined grammar
3. Intermediate code generator for valid inputs
4. Error reporting with token stream display
5. REST API (/compile) for compiler services
6. Web-based frontend with Monaco Editor integration

Progress Overview

Roughly 70% of the core functionalities are complete. The compiler can already parse custom grammar and process code into tokens. Intermediate code generation and parse tree rendering are in progress. UI styling and final testing are pending but expected to complete on time. Some minor UI features were done ahead of schedule.

Codebase Information

GitHub Repository: https://github.com/VinodPandey14/Mini-Compiler
Branch: main
Important Commits:

- feat: added PLY lexer and dynamic parser support
- feat: connect frontend with backend /compile endpoint
- feat: integrate Monaco Editor in React

Testing and Validation Status

Test Type	Status (Pass/Fail)	Notes
Lexical analysis	Pass	Validated with sample grammars and inputs
Syntax parsing	Pass	Dynamic parsing works with multiple grammars
Intermediate code gen	Pass	Currently being finalized
API integration test	Pass	/compile endpoint works as expected
UI usability	In Progress	Needs style polish and visual validation

Deliverables Progress

Deliverable	Status
Frontend UI with Monaco Editor	Completed
Grammar and source code input module	Completed
Lexical analysis module	Completed
Syntax parsing module	Completed
Intermediate code generation module	Completed
Parse tree visualization	Pending
Semantic analysis	Pending
API communication (/compile)	Completed
Final testing and documentation	In Progress