A Project Report for Assignment -1 of COP 701

On

Latex to Markdown Converter

Submitted to

Prof. Smruti Ranjan Sarangi

(Instructor COP 701)

by

Saharsh Laud (2024MCS2002)

Department of Computer Science and Engineering

Indian Institute of Technology, Delhi

Session: 2024-25

CONTENTS

1. INTRODUCTION

1.1 OBJECTIVE

The objective of this assignment is to get familiar with modern-day parsers and understand the process in-depth by actually implementing a latex to markdown parser.

It also aims at getting familiar with proper coding standards that are used in the industry and working on the development of multiple modules that are interconnected and work together to give the final product.

Another objective is to improve knowledge about lexical analysis, parsers, regular expressions, grammar and other key components related to compiler design.

1.2 PROJECT DESCRIPTION

In this assignment, we were asked to make a Latex to Markdown parser from scratch. The parser takes a latex document as input and generates equivalent markdown output.

The parser needs to support latex features such as Section, Italics, Bold, Horizontal Line, Paragraph, Code blocks, Hyperlink, Images, Ordered and Unordered Lists and Tables.

The parser should create an Abstract Syntax Tree and then traverse the Abstract Syntax Tree to generate equivalent markdown output.

2. SYSTEM STUDY

2.1 TOOLS AND TECHNOLOGY USED

C++ : C++ is a powerful, general-purpose programming language. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented, and functional programming.

Flex: Flex is a tool for generating lexical analysers or scanners, which are used to tokenize input text. It performs pattern matching in the input text and then generates token accordingly. It is commonly used in conjunction with Bison (or Yacc), a tool for generating parsers, to create compilers or interpreters.

Bison: Bison is a tool used for generating parsers in programming languages. Bison generates the parser code, which processes the tokens produced by the lexer and builds a syntactic structure (AST) from them. We define grammar using Bison for processing the input.

G++ Compiler: It is the GNU Compiler Collection (GCC) compiler for C++. It is a widely used compiler for C++ programs on various platforms. g++ is known for its robust performance, extensive features, and support for various C++ standards.

Gtest: Google Test (gtest) is a C++ testing framework developed by Google. It provides a robust set of tools for writing and running unit tests for C++ code. It can be used to test various parts of you code using assertations.

Doxygen: Doxygen is a documentation generator for various programming languages, including C++, C, Java, Python, and others. It is widely used to create comprehensive and organized documentation directly from comments in the source code.

2.2 SOFTWARE AND HARDWARE REQUIREMENTS

- C++11
- Flex 2.6 or above
- Bison 3.0 or above
- g++ compiler 11.4.0 or above
- Make 4.3 or above
- GTest 1.8.0 or above
- Operating system: Linux- Ubuntu 22.04.3, or Windows 7 to 11,
- Laptop / Desktop – Any Standard Laptop / Desktop

3. WORKING & CODE MODULES

The entire project has been divided into separate modules for handling different parts of the entire parsing process. Here we will go through all these modules and try to understand the working of the Latex to Markdown parser.

1. Lexer.l

This file is built using Flex (Fast Lexical Analyzer Generator) and performs lexical analysis on input text. Here we specify various pattern matching rules using regular expressions and also the corresponding token for that expression. The main task of lexer is to identify the token value pairs and then forward them to the parser for further processing.

2. Parser.y

This file is built using Bison and it maps the tokens forwarded by the lexer using appropriate grammar rules for Markdown language. It then generates a parser in C/C++ language. Along with the grammar rules, we have also specified the Abstract Syntax Tree Nodes that are created based on various rules so that our AST is created which can be traversed to get the markdown output.

3. Ast.h

This file defines the structure and the functions that will be utilized to create nodes in our AST, traversing the AST, printing the AST and other operations on the AST.

4. Ast.cpp

This file actually implements the functions that are defined in ast.h file.

5. Main.cpp

This is the entry point of the program. Here we are parsing by reading an input LaTeX file and generating an AST. The AST is then traversed to produce a corresponding Markdown file. In case any file opening error or parsing error occurs then it is also handled.

6. Makefile

This file automates the build process.

7. Run.sh

This file automates the execution of programs.

Here the lexer first generates tokens through pattern matching and then the parser uses these tokens along with the grammar rules to generate an AST. Once an AST is created then we just call the traverseAST() function to generated equivalent markdown output.

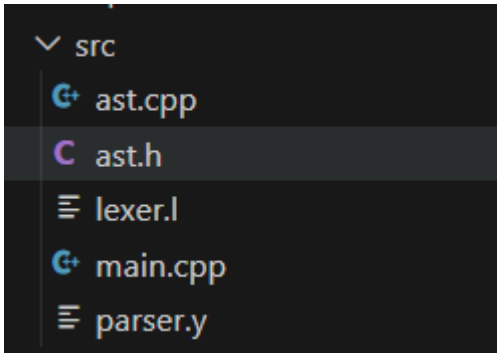

Fig 3.1 - Important files for the project

```
Parsing has been successful. AST structure:
Node: START, Content:
 - Node: DOCUMENTCLASS, Content: article
 - Node: PACKAGE, Content: graphicx
 - Node: PACKAGE, Content: hyperref
 - Node: TITLE, Content: COP701 Sample Test cases
 - Node: DATE, Content: August 2024
 - Node: BEGINDOC, Content:
 - Node: NEWLINE, Content:
 - Node: SECTION, Content: Introduction to IIT Delhi
 - Node: NEWLINE, Content:
 - Node: SUBSECTION, Content: Overview
 - Node: NEWLINE, Content:
 - Node: SUBSUBSECTION, Content: History
 - Node: NEWLINE, Content:
 - Node: NEWLINE, Content:
 - Node: ITALICS, Content: Indian Institute of Technology Delhi (IIT Delhi) is one of the premier engineering institutes in India.
 - Node: NEWLINE, Content:
 - Node: BOLD, Content: Founded in 1961, IIT Delhi has grown into a world-class institution known for its cutting-edge research and academic excellence.
 - Node: NEWLINE, Content:
 - Node: NEWLINE, Content:
 - Node: HRULE, Content:
 - Node: NEWLINE, Content:
 - Node: NEWLINE, Content:
 - Node: TEXT, Content: IIT Delhi is located in Hauz Khas, New Delhi. It offers undergraduate, postgraduate, and doctoral programs in various fields of engineering
ology.
 - Node: NEWLINE, Content:
 - Node: NEWLINE, Content:
 - Node: TEXT, Content: The institute is renowned for its rigorous academic programs and distinguished faculty.
 - Node: NEWLINE, Content:
 - Node: NEWLINE, Content:
 - Node: VERBATIM, Content:
def iit_delhi_info():
    print("Welcome to IIT Delhi!")
    print("Explore the world-class research and academic programs.")
```

Fig 3.2 - AST Structure

4. TESTING

The GoogleTest framework has been used for unit testing the components in this project. Gtest generates a consolidated test report and test cases which are similar can be clubbed into various test suites.
This project takes input and generates output directly into an output file. Taking this into consideration I have provided the latex input and then compared the expected markdown with the markdown code generated in the output file.

Here is a snapshot of the actual working of the gtest.

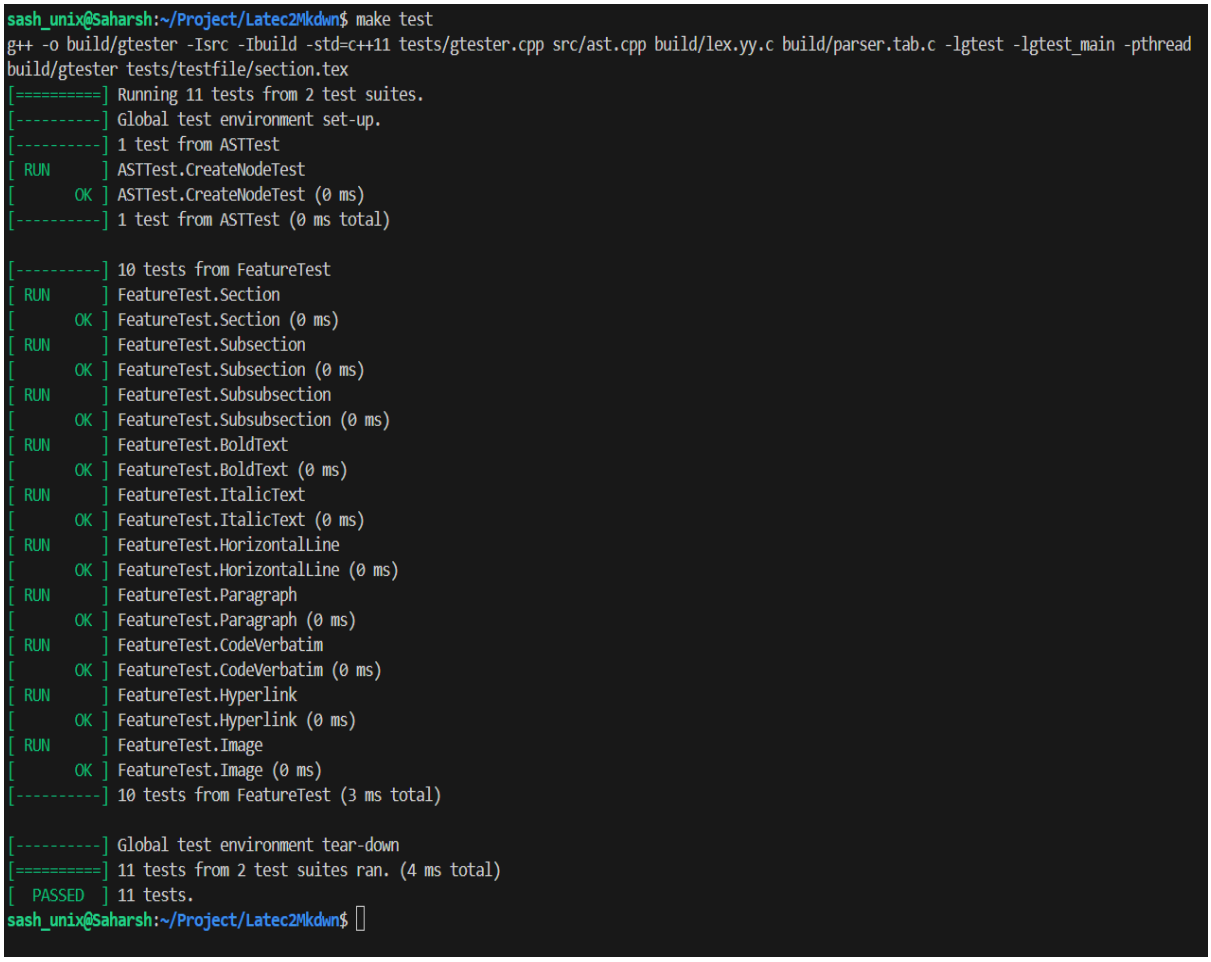

Fig 4.1 – Gtests on various features

5. RESULTS

The latex to markdown parser was successfully built from scratch using Flex-Bison. The parser also builds an AST and uses the AST to generate equivalent markdown output.

input.tex

```latex
\documentclass{article}
\usepackage{graphicx}
\usepackage{hyperref}
\title{COP701 Sample Test cases}
\date{August 2024}
\begin{document}

\section{Introduction to IIT Delhi}
\subsection{Overview}
\subsubsection{History}

\textit{Indian Institute of Technology Delhi (IIT Delhi) is one of the premier engineering institutes in India.}
\textbf{Founded in 1961, IIT Delhi has grown into a world-class institution known for its cutting-edge research and academic excellence.}

\hrule

IIT Delhi is located in Hauz Khas, New Delhi. It offers undergraduate, postgraduate, and doctoral programs in various fields of engineering and
The institute is renowned for its rigorous academic programs and distinguished faculty.

\begin{verbatim}
def iit_delhi_info():
    print("Welcome to IIT Delhi!")
    print("Explore the world-class research and academic programs.")
\end{verbatim}

For more information, visit the official IIT Delhi website: \href{https://www.iitd.ac.in}{IIT Delhi Official Website}

\includegraphics[width=0.5\textwidth]{images/technology.jpg}

\begin{itemize}
    \item Established: 1961
    \item Location: Hauz Khas, New Delhi
    \item Programs: B.Tech, M.Tech, Ph.D., and more
\end{itemize}

\end{document}
```

Fig 5.1 – Input latex file

output.md > ...

```markdown
# Introduction to IIT Delhi
## Overview
### History

*Indian Institute of Technology Delhi (IIT Delhi) is one of the premier engineering institutes in India.*
**Founded in 1961, IIT Delhi has grown into a world-class institution known for its cutting-edge research and academic excellence.**


---


IIT Delhi is located in Hauz Khas, New Delhi. It offers undergraduate, postgraduate, and doctoral programs in various fields of engineering and technology.

The institute is renowned for its rigorous academic programs and distinguished faculty.

```
def iit_delhi_info():
    print("Welcome to IIT Delhi!")
    print("Explore the world-class research and academic programs.")
```

For more information, visit the official IIT Delhi website: [IIT Delhi Official Website](https://www.iitd.ac.in)

![alt text](images/technology.jpg)

    - Established: 1961
    - Location: Hauz Khas, New Delhi
    - Programs: B.Tech, M.Tech, Ph.D., and more
```

Fig 5.2 – Output Markdown file

The entire project was also maintained as a GitHub repository with regular updates and commits.

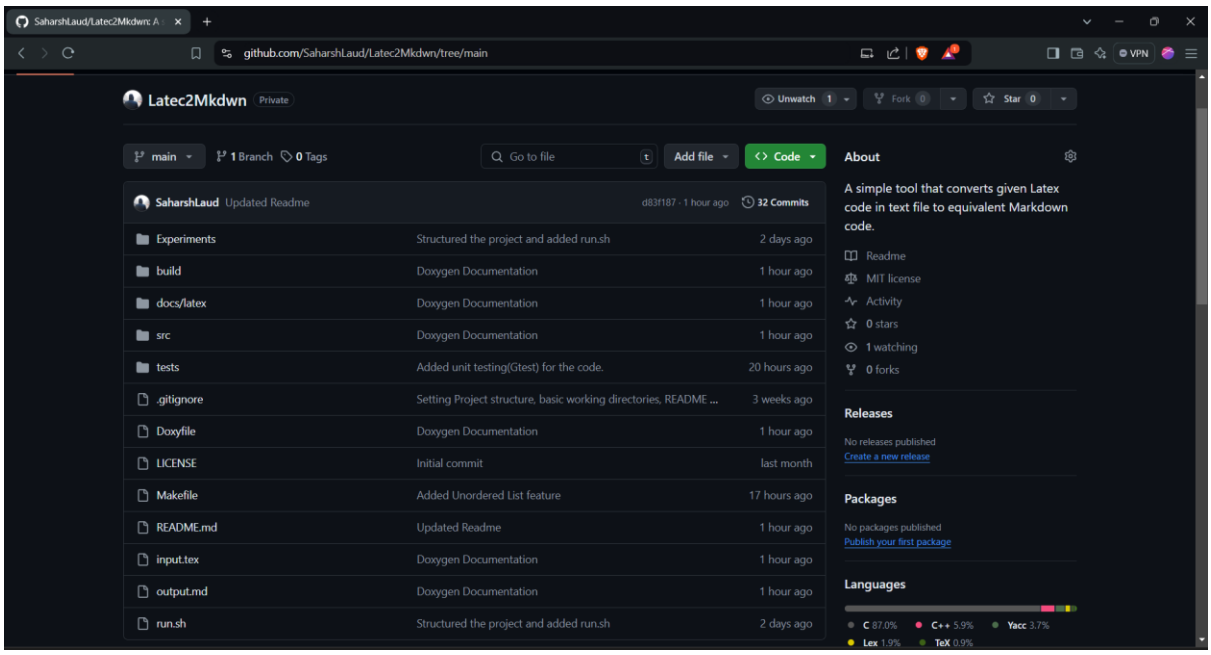

Fig 5.3 – GitHub repository for the project

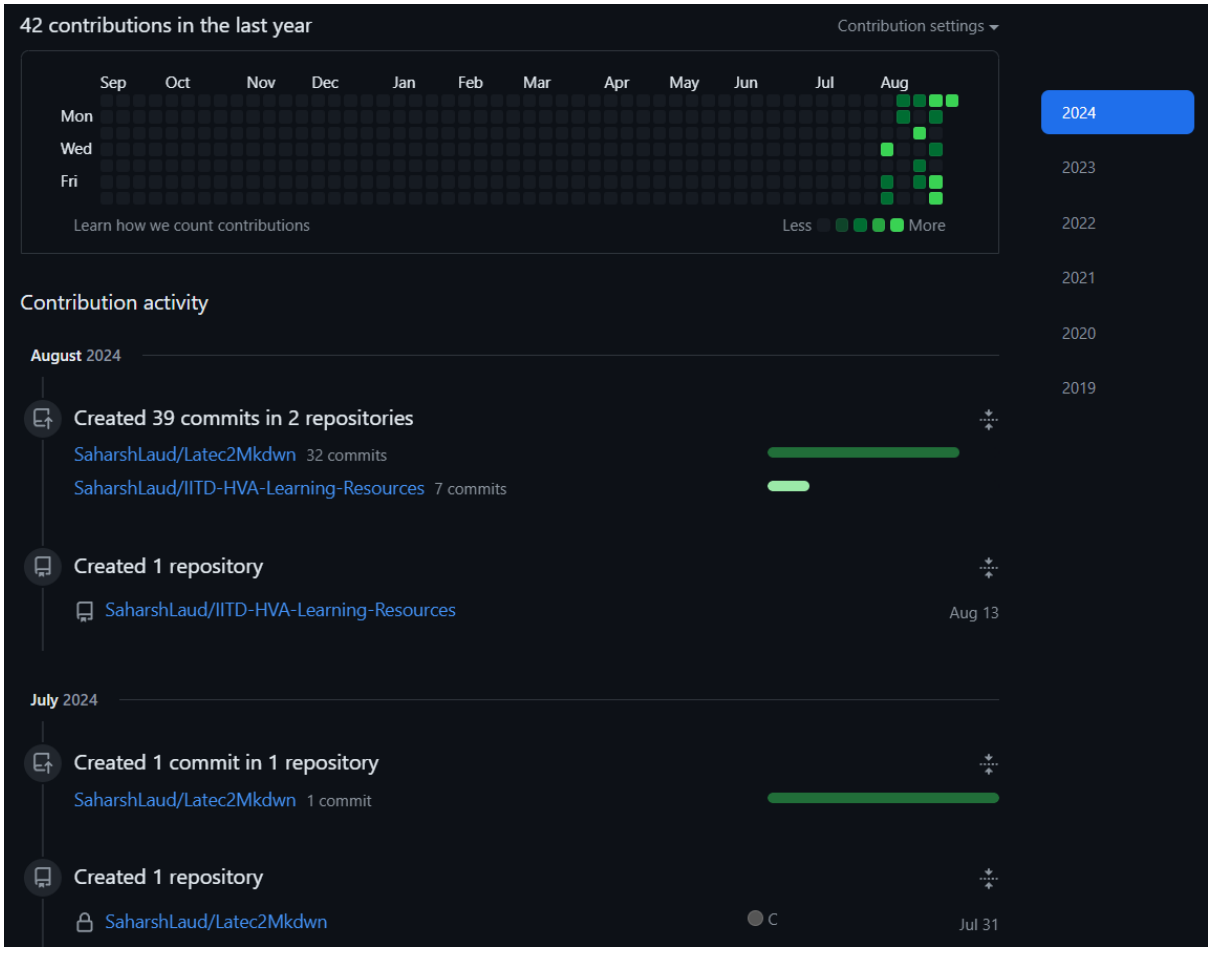

Fig 5.4 – Commit History

6. LIMITATIONS

Although the latex to markdown parser was built successfully it still has some limitations:

- The parser does not support ordered lists and tables.
- The AST created is linear in structure.
- More efficient handling of tabs, spaces and text is needed.

7. REFERENCES

[1] Links referred:

- https://lloydrochester.com/post/flex-bison/json-parse-ast/
- https://github.com/gfrey/reentrant_flex_bison_parser/tree/master
- https://github.com/swakath/markdown2html

[2] Book referred:

- Flex & Bison By John Levine