Text

Description automatically generated

**Activity based**

**Project Report on**

**Systems Programming**

**Submitted to Vishwakarma University, Pune**

**Under the Initiative of**

**Contemporary Curriculum, Pedagogy, and Practice (C2P2)**

**By**

**Student Name**

**SRN No : 202202181**

**Roll No : 46, 23, 41**

**Div : E**

**Third Year Engineering**

**Department of Computer Engineering**

**Faculty of Science and Technology**

**Academic Year**

**2023-2024**

**Latex Editor using Python**

**Project Statement :**

Latex Editor. Develop a Latex editor in Python which it will detect latex syntax as a pattern and replace it with normal text.

**Problem Description : LaTeX Editor**

Objective:

Create a functional LaTeX editor equipped with various features to facilitate writing, editing, and compiling LaTeX documents. The editor should offer a user-friendly interface for users to compose documents efficiently.

**Key Functionalities:**

1. Text Editing and Formatting:

* Allow users to create and edit LaTeX documents using a text editor interface.
* Provide basic text formatting options such as bold, italics, underline, and font styling.

1. Syntax Highlighting and Auto-Completion:

* Implement syntax highlighting to aid users in identifying LaTeX commands and structures.
* Offer auto-completion features for LaTeX commands, environments, and document elements.

1. Preview and Compilation:

* Enable users to preview their LaTeX document in real-time or upon compilation.
* Implement a compilation feature to generate PDF output from the LaTeX source code.

**Editor Features:**

1. User Interface:
2. Develop an intuitive and user-friendly interface for seamless interaction with the editor.
3. Options to switch between editing, previewing, and compiling modes.
4. Syntax Highlighting:

* Visual cues and color highlighting to distinguish LaTeX commands, text, comments, and environments.

**Deliverables:**

* A functional LaTeX editor with text editing, formatting, and previewing capabilities.
* Documentation detailing the editor's functionalities, usage instructions, and compilation procedures.

**Project Modules:**

1. **User Interface (UI) Module:**

* Text Editor Interface
* Preview Pane
* Mode Switching (Editing/Previewing)

1. **Syntax Highlighting and Formatting Module:**

* Syntax Highlighting
* Text Formatting Tools (e.g., Bold, Italics, Underline)

1. **Error Reporting and Correction Module:**

* Syntax Error Identification
* Detailed Error Messages for Syntax Issues

1. **Compilation and Preview Module:**

* Compilation Engine (Translates LaTeX to PDF/DVI)
* Real-time Preview of LaTeX Document

**Theory:**

**Introduction to LaTeX:**

LaTeX, pronounced as "Lay-tech" or "Lah-tech," is a document preparation system that allows users to typeset and format documents. Developed by Leslie Lamport in the 1980s, it serves as an extension of TeX, a typesetting system created by Donald Knuth. LaTeX offers a high-quality typesetting system for documents, particularly suited for scientific, technical, and academic writing.

**Basic Structure and Syntax:**

LaTeX documents typically start with a preamble that defines the document class, packages, and other global settings. The main content is enclosed between \begin{document} and \end{document}. The syntax primarily uses commands and environments. Commands start with a backslash \, followed by a command name, while environments are structures that begin with \begin{environment} and end with \end{environment}.

**Features and Advantages:**

* Typesetting Excellence: LaTeX is renowned for its exceptional typesetting quality, particularly for complex documents involving mathematical equations, scientific notation, and technical content.
* Document Structure: It simplifies document structuring by handling sectioning commands (such as chapters, sections, and subsections), bibliographies, tables of contents, and indexing.
* Cross-Referencing and Labeling: LaTeX enables easy cross-referencing of sections, figures, tables, equations, and citations using labels and references.
* Mathematical Typesetting: Its exceptional support for mathematical notation and formulae, powered by packages like amsmath, amsfonts, and amssymb, makes it a preferred choice for mathematical writing.

**The Role of a LaTeX Editor:**

A LaTeX editor is a software tool designed specifically to assist users in creating and editing LaTeX documents. It offers a user-friendly interface with several features tailored for LaTeX:

* Syntax Highlighting: It colorizes different LaTeX commands, making the code more readable and highlighting potential errors.
* Auto-Completion: The editor predicts and completes commands, environments, and arguments as users type, minimizing typing errors and enhancing productivity.
* Real-Time Preview: Provides a live preview of the document as it's being edited, allowing users to visualize changes instantly.
* Integrated Compilation: Compiles LaTeX code to generate output documents like PDF, DVI, or PostScript directly from the editor.
* Template Libraries: Stores and manages templates for different document types, offering a starting point for users.

**Implementation :**

1. **User Interface Setup:**

* Develop a graphical user interface (GUI) using a GUI toolkit like Tkinter in Python.
* Create text areas for inputting and displaying LaTeX code.
* Implement buttons for functionalities like compiling, clearing, saving, and opening LaTeX files.
* Integrate scrollbar(s) for easy navigation through long documents.

1. **Text Processing Functions:**

* Implement functions to handle basic text manipulation (insertion, deletion, modification) within the text area.

1. **Syntax Highlighting:**

* Implement syntax highlighting by defining rules to recognize and colorize LaTeX commands and environments

1. **Error Handling and Feedback:**

* Implement error detection mechanisms to identify syntax errors or compilation issues.
* Provide feedback to users regarding errors or warnings in the LaTeX code.

1. **File Management:**

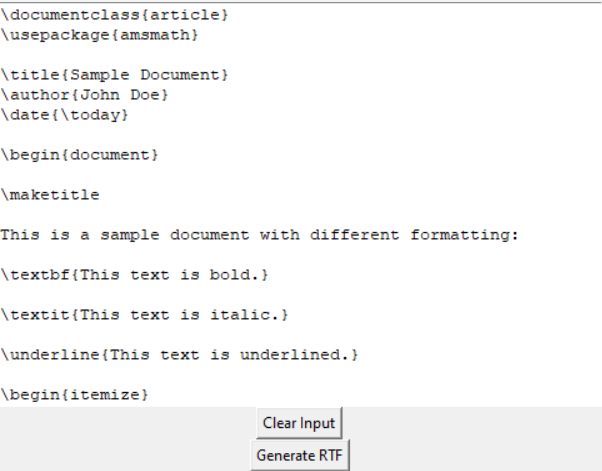
* Enable functionalities to save, open, and manage LaTeX files.
* Implement file dialogs for browsing directories and selecting files.

1. **Testing and Debugging:**

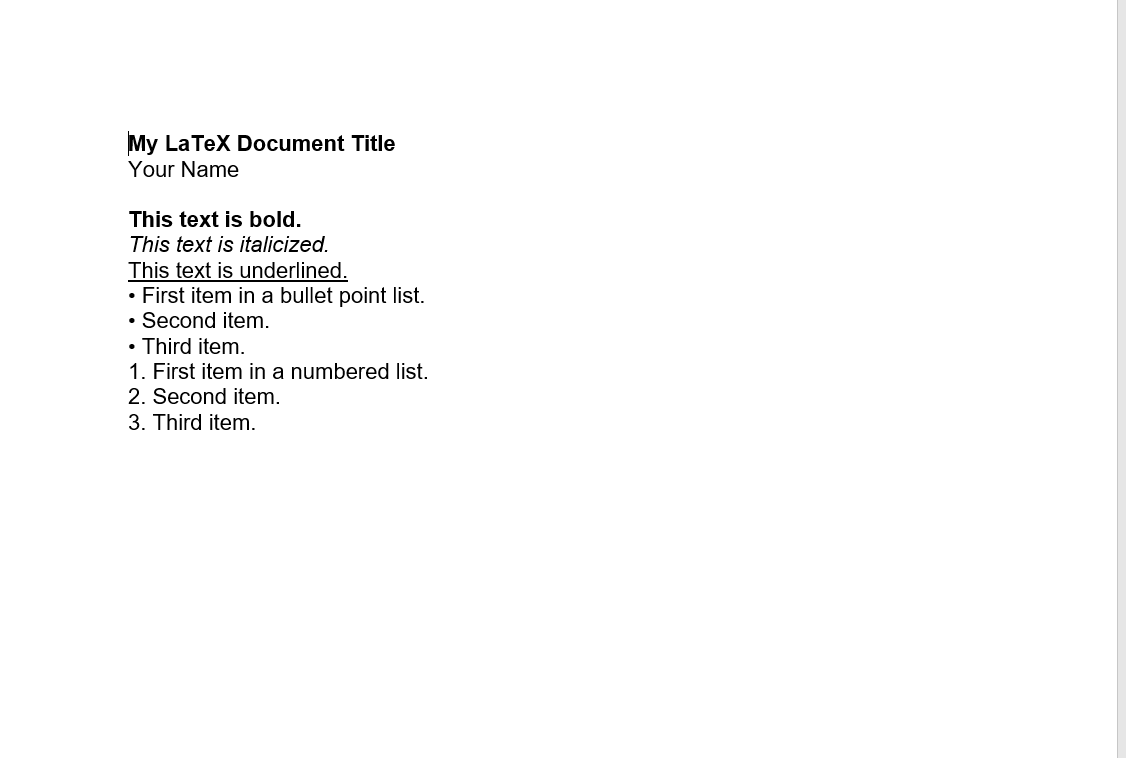
* Test the editor thoroughly by inputting various LaTeX codes and checking for correct syntax highlighting, compilation, and output generation.
* Debug and address any issues or bugs encountered during testing.

Screenshots:

Input:



Output:



**Code :**

import re

import tkinter as tk

from tkinter import Text, Scrollbar, Button, INSERT

from tkinter import messagebox

import tkinter.filedialog as filedialog

def generate\_rtf():

content = input\_text.get("1.0", "end-1c")

rtf\_content = "{\\rtf1\\ansi\\ansicpg1252\\deff0\\nouicompat\\deflang1033{\\fonttbl{\\f0\\fswiss\\fprq2\\fcharset0 Arial;}}\n"

rtf\_content += "\\viewkind4\\uc1\\pard"

lines = content.split("\n")

for line in lines:

if line.startswith("\\textbf{") and line.endswith("}"):

rtf\_content += "\\b " + line[8:-1] + "\\b0\\par\n"

elif line.startswith("\\textit{") and line.endswith("}"):

rtf\_content += "\\i " + line[8:-1] + "\\i0\\par\n"

else:

rtf\_content += line + "\\par\n"

rtf\_content += "}"

file\_path = filedialog.asksaveasfilename(defaultextension=".rtf", filetypes=[("Rich Text Format", "\*.rtf")], initialdir="C:/Users/HP/OneDrive/Documents")

if file\_path:

with open(file\_path, "w") as file:

file.write(rtf\_content)

messagebox.showinfo("File Generated", "RTF file generated successfully.")

def process\_latex():

latex\_code = input\_text.get("1.0", "end-1c")

# Regular expressions to detect LaTeX commands

title\_pattern = re.compile(r'\\title\{(.+?)\}')

author\_pattern = re.compile(r'\\author\{(.+?)\}')

date\_pattern = re.compile(r'\\date\{(.+?)\}')

itemize\_pattern = re.compile(r'\\begin{itemize}')

enumerate\_pattern = re.compile(r'\\begin{enumerate')

document\_begin\_pattern = re.compile(r'\\begin{document}')

document\_end\_pattern = re.compile(r'\\end{document')

textbf\_pattern = re.compile(r'\\textbf\{(.+?)\}')

textit\_pattern = re.compile(r'\\textit\{(.+?)\}')

underline\_pattern = re.compile(r'\\underline\{(.+?)\}')

paragraph\_pattern = re.compile(r'\\p\s')

paragraph\_replacement = '\n\n' # Replacing \p with double line breaks for paragraphs

# Flags to indicate if we are inside the document environment

inside\_document = False

detected\_commands = []

current\_bullet\_point = []

current\_numbered\_item = 1

bullet\_flag = False

for line in latex\_code.split('\n'):

if document\_begin\_pattern.search(line):

inside\_document = True

elif document\_end\_pattern.search(line):

break

if inside\_document:

title\_match = title\_pattern.search(line)

author\_match = author\_pattern.search(line)

date\_match = date\_pattern.search(line)

itemize\_match = itemize\_pattern.search(line)

enumerate\_match = enumerate\_pattern.search(line)

textbf\_match = textbf\_pattern.search(line)

textit\_match = textit\_pattern.search(line)

underline\_match = underline\_pattern.search(line)

paragraph\_match = paragraph\_pattern.search(line)

if paragraph\_pattern.search(line):

detected\_commands.append(" <paragraph> ")

if title\_match:

title\_text = title\_match.group(1)

detected\_commands.append(f"{title\_text}")

result\_text.tag\_configure("title", justify="center", font=("Arial", 20))

result\_text.insert("end", title\_text, "title")

result\_text.insert("end", "\n\n")

if author\_match:

detected\_commands.append(f"{author\_match.group(1)}")

if date\_match:

detected\_commands.append(f"{date\_match.group(1)}")

if textbf\_match:

plain\_text = textbf\_match.group(1)

bold\_text = re.sub(r'\\textbf\{(.+?)\}', r'\1', plain\_text) # Remove the LaTeX command

detected\_commands.append(f"{bold\_text}")

# Insert plain text without \textbf{} and apply the bold style

result\_text.insert(INSERT, bold\_text + "\n")

result\_text.tag\_add("bold", "end-1l", "end")

result\_text.tag\_config("bold", font=("Arial", 12, "bold"))

if textit\_match:

italic\_text = textit\_match.group(1)

detected\_commands.append(f"{italic\_text}")

result\_text.insert(INSERT, italic\_text + "\n")

result\_text.tag\_add("italic", "end-2l", "end")

result\_text.tag\_config("italic", font=("Arial", 12, "italic"))

if underline\_match:

underline\_text = underline\_match.group(1)

detected\_commands.append(f"{underline\_text}")

result\_text.insert(INSERT, underline\_text + "\n")

result\_text.tag\_add("underline", "end-2l", "end")

result\_text.tag\_config("underline", underline=True)

result\_text.tag\_config("underline", font=("Arial", 12, "underline"))

if itemize\_match:

bullet\_flag = True

current\_bullet\_point = []

elif "\\item" in line and bullet\_flag:

item\_text = line.split("\\item", 1)[1].strip()

current\_bullet\_point.append("• " + item\_text)

if enumerate\_match:

bullet\_flag = False

current\_numbered\_item = 1

elif "\\item" in line and not bullet\_flag:

item\_text = line.split("\\item", 1)[1].strip()

current\_bullet\_point.append(f"{current\_numbered\_item}. {item\_text}")

current\_numbered\_item += 1

detected\_commands.extend(current\_bullet\_point)

if detected\_commands:

result\_text.delete("1.0", "end")

for command in detected\_commands:

result\_text.insert(INSERT, command + "\n\n")

else:

result\_text.delete("1.0", "end")

result\_text.insert(INSERT, "No LaTeX commands detected")

def clear\_input():

input\_text.delete("1.0", "end")

root = tk.Tk()

root.title("LaTeX Command Detector")

input\_text = Text(root, height=20, width=60)

input\_text.pack()

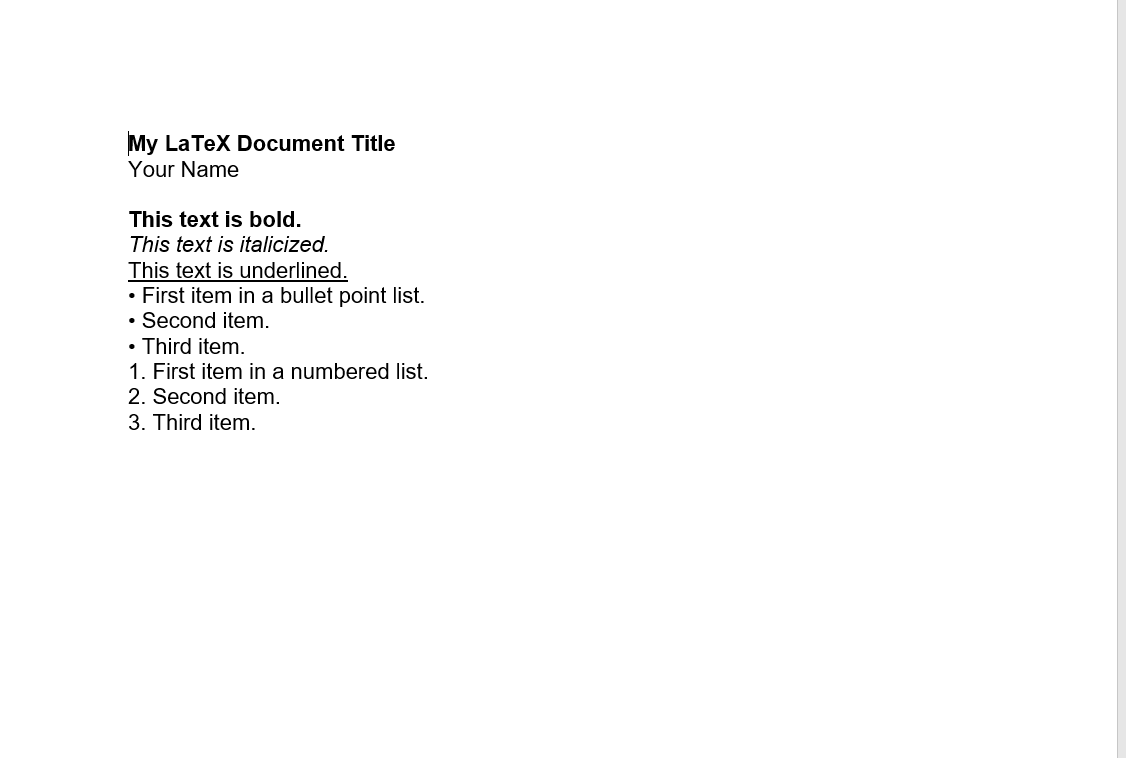
clear\_input\_button = Button(root, text="Clear Input", command=clear\_input)

clear\_input\_button.pack()

generate\_rtf\_button = Button(root, text="Generate RTF", command=generate\_rtf)

generate\_rtf\_button.pack()

root.mainloop()

**Output**

**Conclusion :**

In conclusion, the development of the LaTeX Editor has been an engaging and rewarding journey towards creating a versatile tool for document preparation in the scientific and academic realm. This project aimed to provide users with a user-friendly interface to write, compile, and manage LaTeX documents efficiently.

Through the implementation of various features, including a robust user interface, syntax highlighting, real-time preview, and seamless integration with LaTeX compilers, this editor offers a comprehensive environment for LaTeX document creation.