Summer Project On

PyRho - Python IDE

By

Omprakash Kambli (2021510025)

Under the guidance of

Internal Supervisor

Prof. Sakina Shaikh

Department of Master Of Computer Application
Sardar Patel Institute of Technology
Autonomous Institute Affiliated to Mumbai University

2022-23

CERTIFICATE OF APPROVAL

This is to certify that the following students

Omprakash Kambli (2021510025)

Has satisfactorily carried out work on the project
entitled

"PyRho - Python IDE"

Towards the fulfilment of project, as laid down
by
Sardar Patel Institute of Technology
during year
2022-23.

Project Guide:
Sakina Shaikh

PROJECT APPROVAL CERTIFICATE

This is to certify that the following students

Omprakash Kambli (2021510025)

Have successfully completed the Project report on

"PyRho - Python IDE",

which is found to be satisfactory and is approved

at

SARDAR PATEL INSTITUTE OF TECHNOLOGY,
ANDHERI (W), MUMBAI

INTERNAL EXAMINER EXTERNAL EXAMINER

HEAD OF DEPARTMENT PRINCIPAL

Contents

Abstract

Pyrho is a Python IDE Built in python for linux operating system. It provides the basic functionalities of an IDE as well as allows us to run python scripts without saving them first.

The User can create new files, open a new python project, make changes to the source code of the IDE itself to suit their needs and add functionalities of their choice per se.

The app will provide the users with a user-friendly interface which is bloat-free as compared to the existing systems as well as smaller in size.

Objectives

The Python based Application "Pyrho - Python IDE" is used

- To provide a user friendly interface to code in python
- To provide a convenient way to run scripts without wasting space on the device
- To provide a better coding experience on linux without the need for complex packages
- To provide an IDE that can be customized by changing it's source code.

List of Figures

List of Tables

1 Introduction

1.1 Problem Definition

To eliminate the need for a complex application to write python scripts. To allow users to run code without the need to save it everytime

1.2 Objectives and Scope

1.2.1 Objectives

The Python based application "PyRho - Python IDE" is

- To provide a better coding experience on linux without the need for complex packages
- To provide an IDE that can be customized by changing it's source code.
- To provide a user friendly interface to code in python
- To provide a convenient way to run scripts without wasting space on the device

1.2.2 Scope

The student can provide his/her details in the profile and view for various open internship and placement offers on the app.

In the application the user must type the code in the code editor GUI on the right hand side of the screen

Our System is being made for reducing the space requirements for a code-editor and IDE so that there is lesser load on the individual system.

1.3 Existing System

The IDEs that are currently present are quite bulky in size and also are complex to use for a beginner. They tend to be resource as well as space intensive and may require complex installations at times.
Some of the disadvantages of existing system are as follows :

- Save and Run Small Codes
  Every time the code needs to be saved unless you are using the Python shell which does not allow you to run multiple lines of code at a time efficiently.
- Large Size
  Most of the times the applications tend to take up a lot of unnecessary space and may require more space later for it's functioning
- Inconvenient to use
  There at times can be a need of complex steps to install and run applications. The same might not be documented properly.

1.4 Proposed System

The User is the Person who wants to code using the IDE who would open the application and type out the code as required. This System will be able to function on any debian based linux distribution without hassle given that it has python3 installed and up to date.

The application will provide the basic functionalities of a code editor, lexer and autocompletion and also be able to function properly on low-spec systems which might not be able to provide resources to store a lot of data or run high-end applications.

You can also test your code without saving it since it allows the use of temp files to run code until you save it. Once the file has been executed the executed python script gets removed from the system and cannot be traced back to the user.

Some of the advantages of our system are as follows :

- User Friendly
  It provides a simple and user friendly interface to the user for writing code for python language
  Allow running code without the need to save it
  The ability to run code on a machine with low specs.

- Privacy Considerate
  The ability to run code and leave no traces on the client machine once it has been executed.
  Absence of any third-party applications or trackers embedded within so your code is truly yours and no one can access it under regular circumstances.

1.5 System Requirements

- Hardware Requirements on Development Machine

Table 1.5.1: Hardware Requirements on Development Machine

Processor	Dual Core Processor or Above
RAM	Minimum 1 GB RAM
Storage	Minimum 10 GB Hard Disk Space for smooth run

- Software Requirements on Development Machine

Table 1.5.2: Software Requirements on Development Machine

OS	Any Debian Based OS
Packaging System	dpkg or apt based

2 Software Requirement Specification (SRS) and Design

2.1 Purpose

The purpose of our project is to develop an UI application that can help user (Programmer) to efficiently write code without the need for a complex interface or a high-end device either
This can save lots of space on a users device and also allow those with a low-end PC to code in python as well as save on precious storage space which the older PCs or the new ones with an SSD these days lack. This app will also cater to the needs of a privacy conscious programmer who wants no one else to access his/her code until they wish to let them.

2.2 Definition

To build a Python IDE that is user-friendly and able to function on low-spec devices.

2.3 Overall Description

2.3.1 Product Functions

The product function includes:

1. Create File: Users can create a new file to start writing code.
2. Open File: This will open an existing file
3. Save File: This will save the current file
4. Execute File: Runs the code currently written in the editor section of the application
5. Beautify Code: Makes the code more legible and tries to resolve indentations wherever possible

2.3.2 User Characteristics

There is a single type of user:

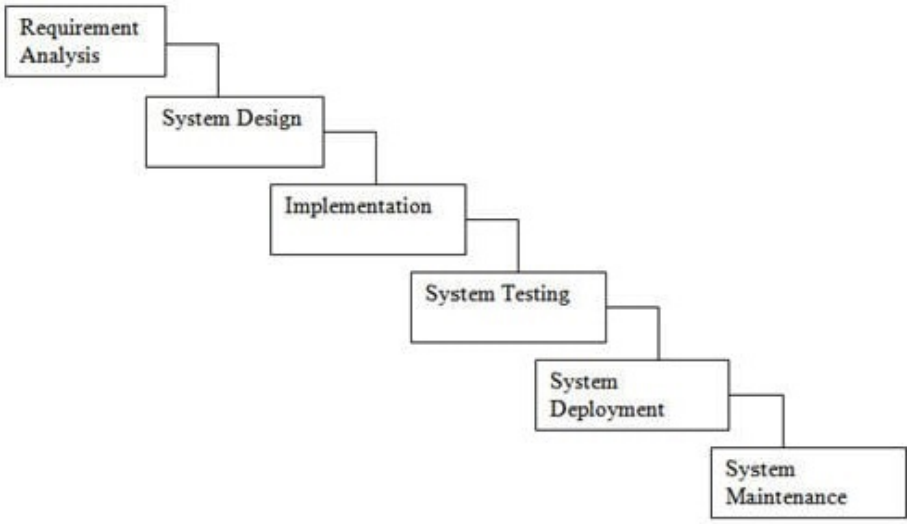
- The user of the application can write scripts, save them and execute them as per need.

3 Project Analysis and Design

3.1 Methodologies Adapted

In Waterfall model, very less customer interaction is involved during the development of the product. Once the product is ready then only it can be demonstrated to the end users.

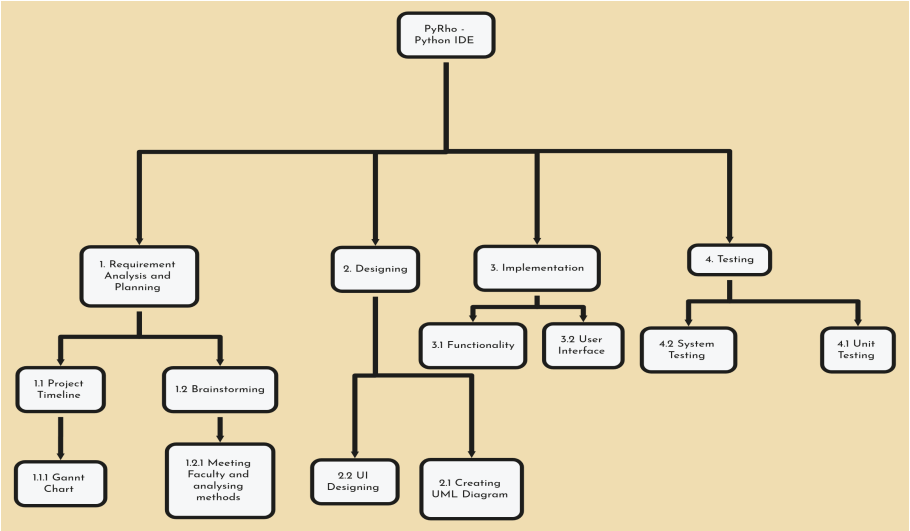
Once the product is developed and if any failure occurs then the cost of such issues is very high, because we need to update everything from document till the logic.



3.1.1: Diagrammatic Representation of Waterfall Model

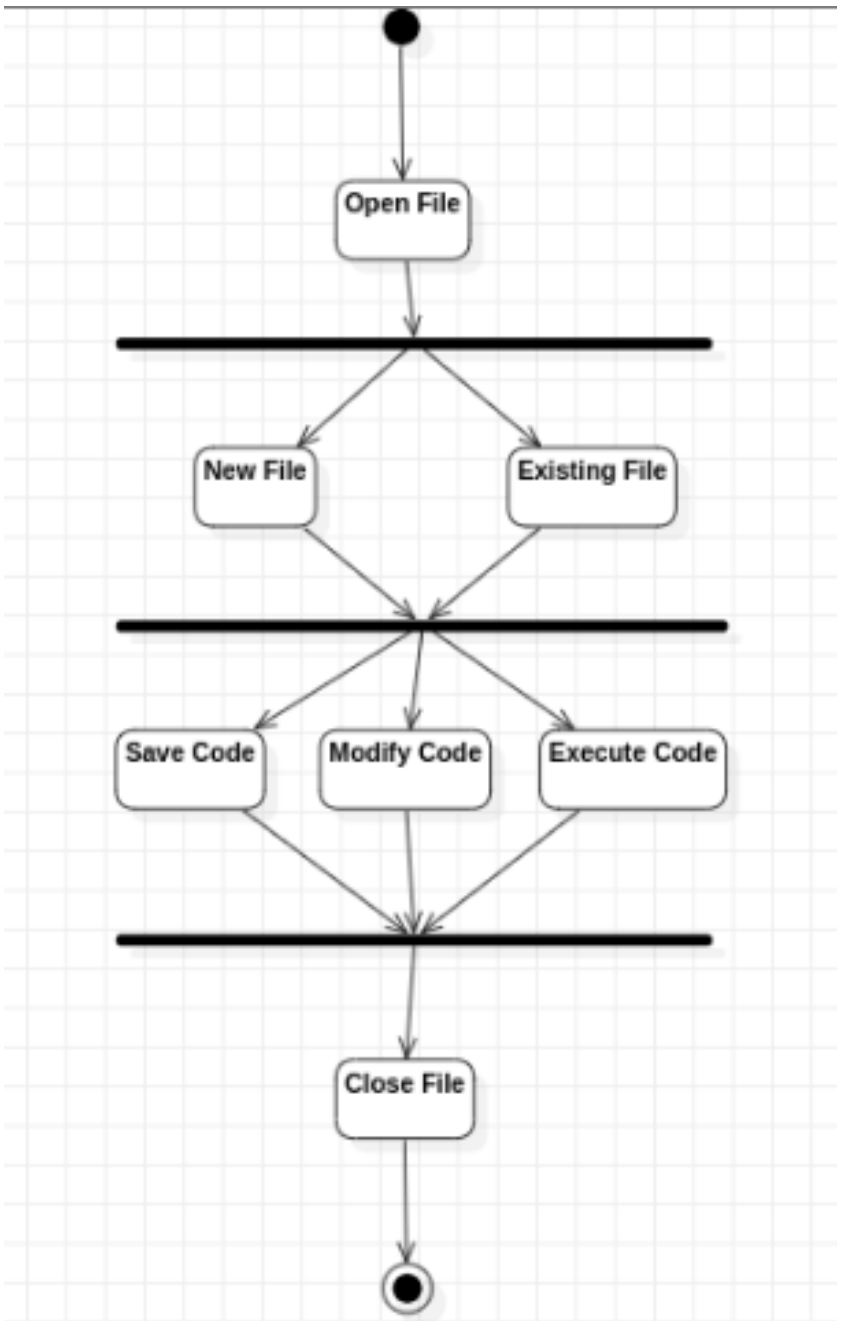
3.2 Modules

3.2.1 Work Breakdown Structure



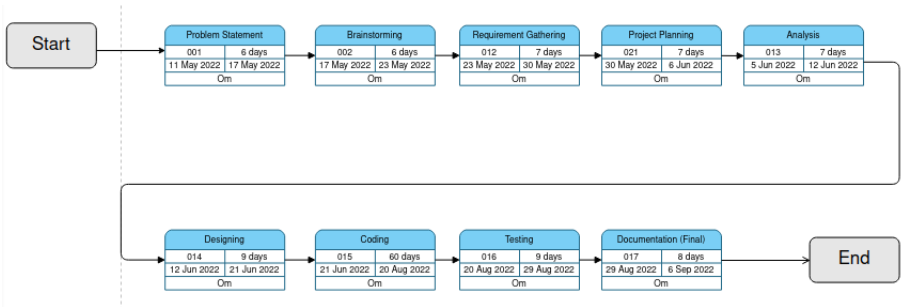
3.2.1: Work Breakdown Structure

3.2.2 Activity diagram



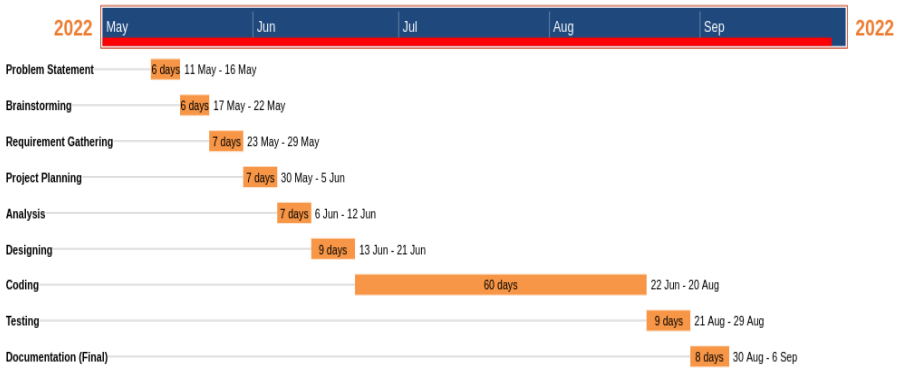
3.2.2: Activity Diagram

3.2.3 PERT Chart



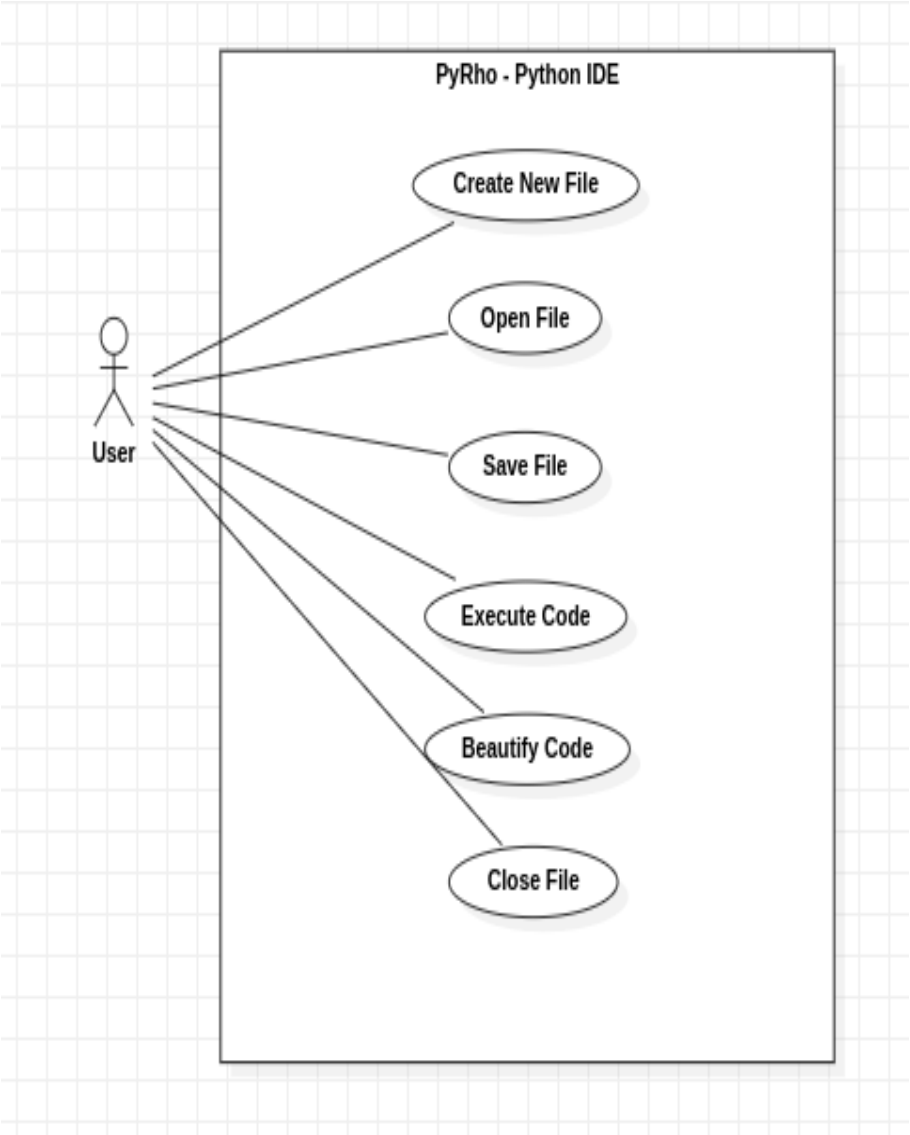
3.2.3: PERT Chart

3.2.4 Gantt Chart



3.2.4: Gantt Chart

3.2.5 Use-Case



3.2.5: Use-Case Diagram

Use Cases:

1. Create New File
2. Open File
3. Save File
4. Execute Code
5. Beautify Code
6. Close File

Table 4.2.1: Use Case Table - Create New File

Use Case ID	1
Use Case Name	Create New File
Actor	User
Pre-Condition	Application must be open
Post-Condition	None
Flow of events	File menu and click on New or Ctrl N

Table 4.2.2: Use Case Table - Open File

Use Case ID	2
Use Case Name	Open File
Actor	User
Pre-Condition	File Must Exist
Post-Condition	File must be in a pythonic format
Flow of events	File menu and click on Open or Ctrl O and then browse and select the desired file.

Table 4.2.3: Use Case Table - Save File

Use Case ID	3
Use Case Name	Save File
Actor	User
Pre-Condition	File must be open in editor
Post-Condition	User must provide a valid name for the file

Table 4.2.4: Use Case Table - Execute Code

Use Case ID	4
Use Case Name	Execute Code
Actor	User
Pre-Condition	Code editor must be open
Post-Condition	The code written must be error-free to execute properly

Table 4.2.5: Use Case Table - Beautify Code

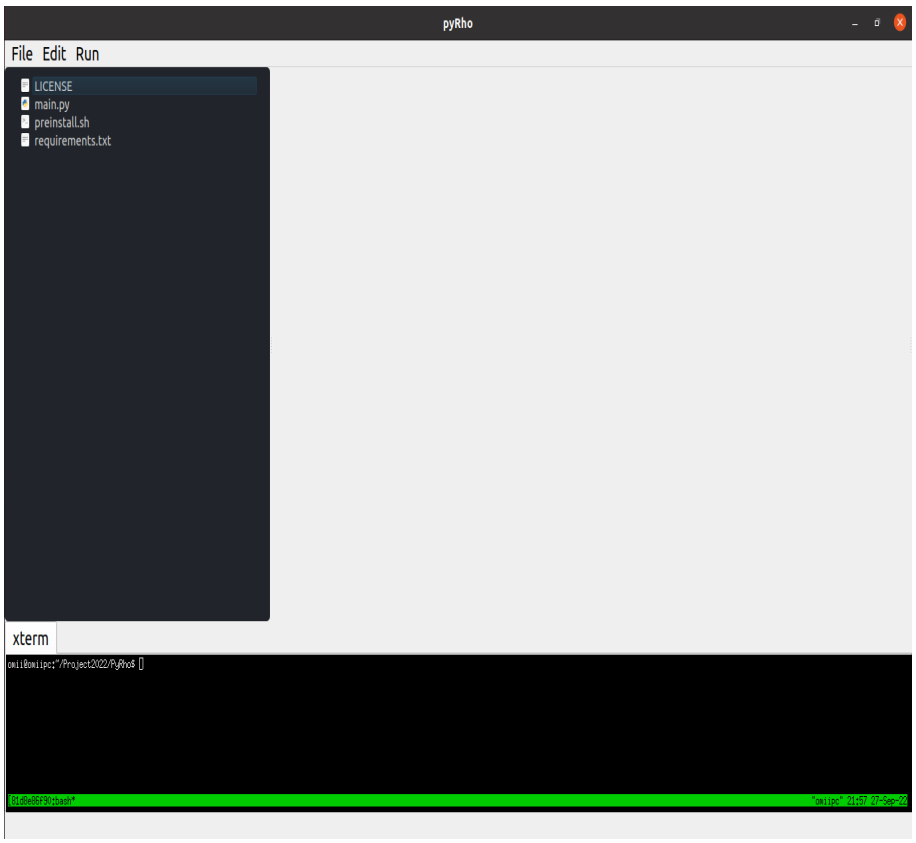
Use Case ID	5
Use Case Name	Beautify Code
Actor	User
Pre-Condition	Code Editor must be open and contain some text
Post-Condition	None

Table 4.2.6: Use Case Table - Close File

Use Case ID	6
Use Case Name	Close File
Actor	User
Pre-Condition	File must be open
Post-Condition	None

4 Project Implementation and Testing

4.1 Initial UI

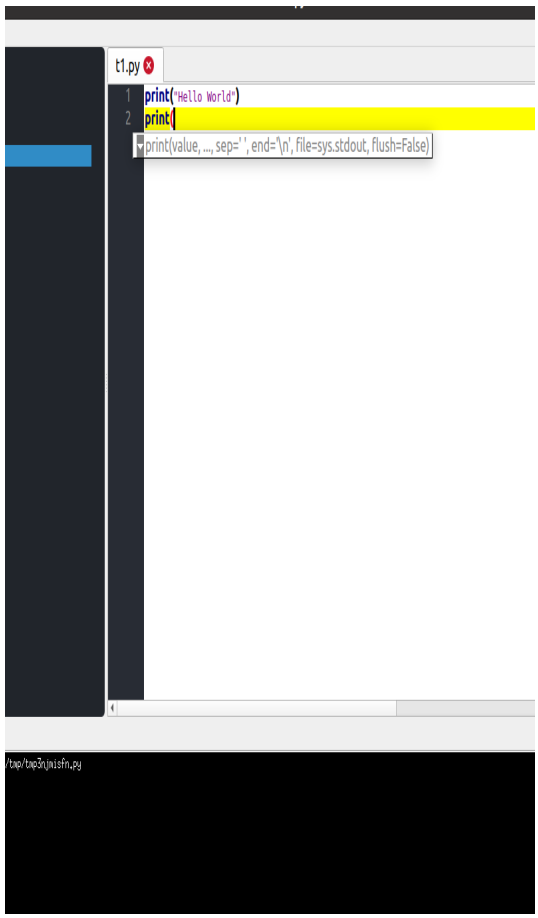


4.1.1: Initial UI

4.2 Code Execution

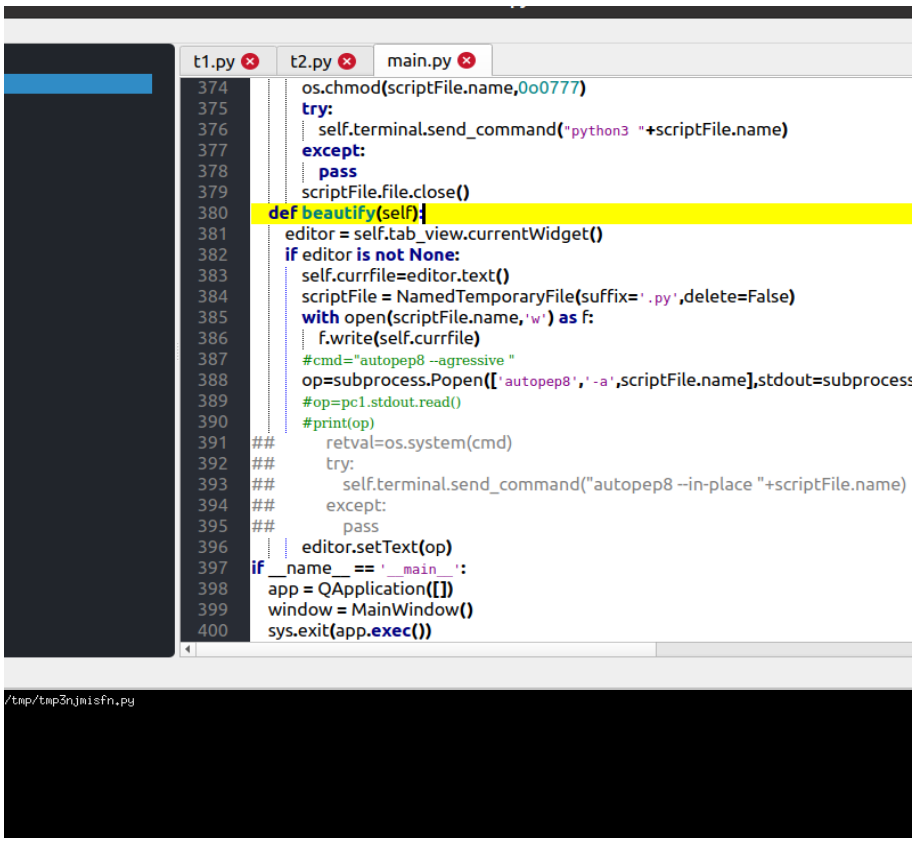
4.2.1: Code Execution

4.3 Autocompletion



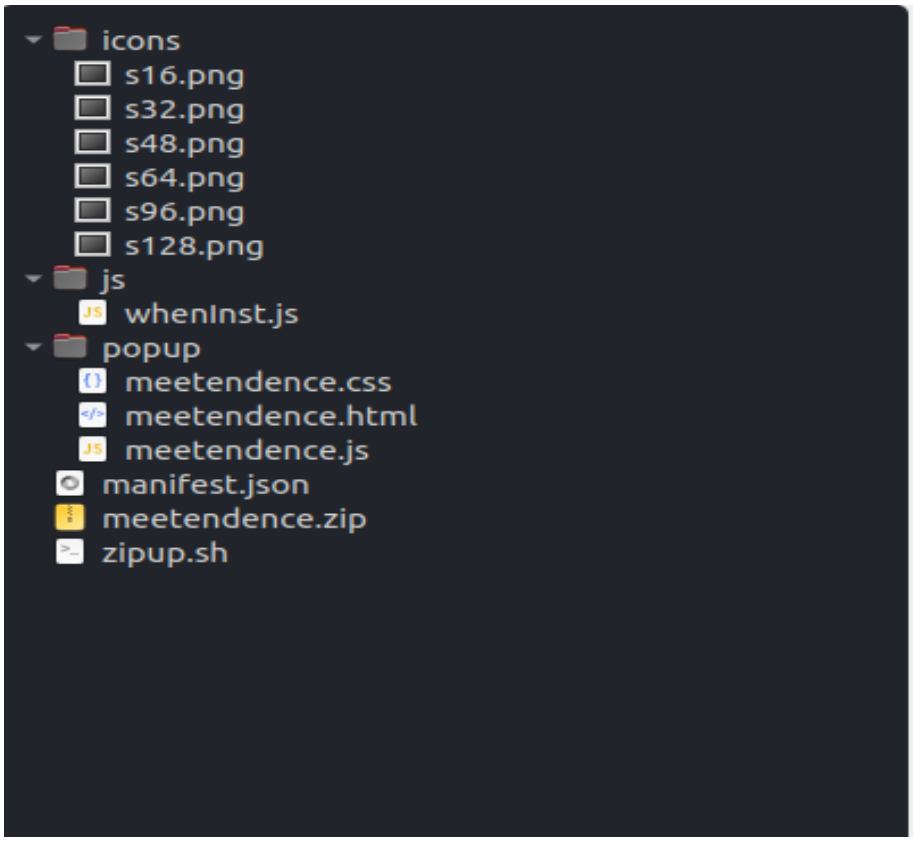
4.3.1: Autocompletion

4.4 Line Numbering and current line highlighted



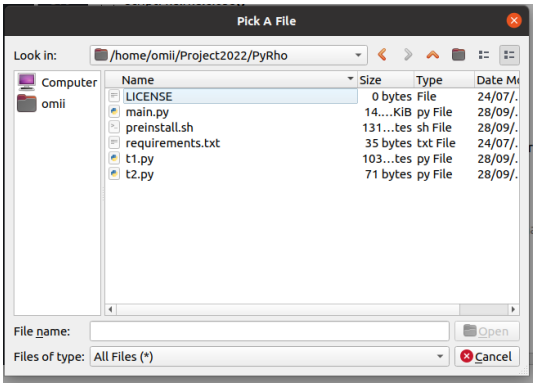
4.4.1: Line Numbering and current line highlighted

4.5 TreeView for File Browser



4.5.1: TreeView for File Browser

4.6 File Picker Dialog



4.6.1: File Picker Dialog

4.7 Code 1

main.py

```python

def save_as(self):
    editor = self.tab_view.currentWidget()
    if editor is None:
        return
    ispyth=False
    file_path = QFileDialog.getSaveFileName(self, "Save As", os.getcwd(),"*.py")[0]
    pyexs=[".py",".pyx",".pyc",".pyw",".pyd",".pyi",".xpy",".pyp",".pyz"]
    for i in pyexs:
        if(file_path.endswith(i)==1):
            ispyth=True
        else:
            pass

    if file_path == '':
        self.statusBar().showMessage("Cancelled", 2000)
        return
    if(not ispyth):
        file_path+=".py"
    path = Path(file_path)
    path.write_text(editor.text())
    self.tab_view.setTabText(self.tab_view.currentIndex(), path.name)
    self.statusBar().showMessage(f"Saved {path.name}", 2000)
    self.current_file = path
```

4.8 Code 2

main.py

```python
def set_up_menu(self):
    menu_bar = self.menuBar()
    file_menu = menu_bar.addMenu("File")
    new_file = file_menu.addAction("New")
    new_file.setShortcut("Ctrl+N")
    new_file.triggered.connect(self.new_file)
    open_file = file_menu.addAction("Open File")
    open_file.setShortcut("Ctrl+O")
    open_file.triggered.connect(self.open_file)
    open_folder = file_menu.addAction("Open Folder")
    open_folder.setShortcut("Ctrl+K")
    open_folder.triggered.connect(self.open_folder)
    file_menu.addSeparator()
    save_file = file_menu.addAction("Save")
    save_file.setShortcut("Ctrl+S")
    save_file.triggered.connect(self.save_file)
    save_as = file_menu.addAction("Save As")
    save_as.setShortcut("Ctrl+Shift+S")
    save_as.triggered.connect(self.save_as)
    edit_menu = menu_bar.addMenu("Edit")
    copy_action = edit_menu.addAction("Copy")
    copy_action.setShortcut("Ctrl+C")
    copy_action.triggered.connect(self.copy)
    run_action= menu_bar.addAction("Run")
    run_action.setShortcut("F5")
    run_action.triggered.connect(self.run_file_fromeditor)
    btfy_btn= menu_bar.addAction("Beautify")
```

4.9 Code 3

main.py

```
def copy(self):
    editor = self.tab_view.currentWidget()
    if editor is not None:
        editor.copy()
def run_file_fromeditor(self):
    editor = self.tab_view.currentWidget()
    if editor is not None:
        self.currfile=editor.text()
        scriptFile = NamedTemporaryFile(suffix='.py',delete=False)
        with open(scriptFile.name,'w') as f:
            f.write(self.currfile)
        os.chmod(scriptFile.name,0o0777)
        try:
            self.terminal.send_command("python3 "+scriptFile.name)
        except:
            pass
        scriptFile.file.close()
```

5 Test Cases

Table 5.1: Test Case - Code Execution

Test Case ID	Test Case Name	Test Data	Expected Output	Actual Output	Result
1	User writes the code of for loop	Enters the correct parameters	Proper Output Displayed	Correct Output	Pass
2	User writes the code of for loop	Enters the wrong parameters	Error Displayed	Exception Occured	Pass
3	User writes the code of for loop	Uses Improper Indentation	Indentation Mismatch Displayed	Error Displayed	Pass
4	User writes the code of for loop	Enters incomplete statement	Syntax Error Displayed	Error Displayed	Pass

Table 5.2: Test Case - Save and Close

Test Case ID	Test Case Name	Test Data	Expected Output	Actual Output	Result
1	User creates the new file	Enters no extension	File is saved with extension	File saved with extension .py	Pass
2	User creates the new file	Enters .py as extension	File is saved as .py	File saved with extension .py	Pass

6 Limitations

- It needs a linux system to use.
- It does not have a feature to connect with any third-party applications or extensions.
- The app could have had support for more languages
- It does not have a feature to autocomplete code extensively like the ones by JetBrains which use AI based methods

7 Future Enhancements

- Code Snippets by Shortcuts
- Feature of fetching problem statements from Platforms like CodeChef, Codeforces.
- Support for more languages in addition to python.
- Inclusion of data-visualization libraries during installation.

8 User Manual

Part 1 – Open File
Upon opening the application, user will be able to view a screen with a treeview of the files on left and an empty pane for the editor on the right. As soon as you click on create file or open a file, the editor would be displayed.
You can open multiple files and switch between tabs or close them as required.

Part 2 – Open Folder
User can open a folder to act as a workspace and view other files present in the folder within the editor.

Part 3 – Save File
User can save the files for later use.

Part 4 – Copy
All text present in the editor is properly copied onto the clipboard for the user to paste with ease.

Part 5 – Run
User can run code written within the editor regardless of it being saved or not saved.

Part 6 – Beautify Code
User can reindent the code and make changes to it as per PEP8 standards for better view of the code.

9 Bibliography

9.1 Web References

[1.] http://www.python.org/
[2.] https://qscintilla.com/
[3.] https://www.youtube.com/c/Freecodecamp
[4.] https://stackoverflow.com/
[5.] https://www.draw.io/
[6.] https://www.geeksforgeeks.org/unified-modeling-language-uml-introduction/

9.2 Other References

[1.] https://www.oreilly.com/library/view/programming-in-python/9780137155149/