

CODE EDITOR

A Project Report

Submitted in partial fulfilment of the
Requirements for the award of the Degree of

BACHELOR OF SCIENCE (COMPUTER SCIENCE)

By

KANAD RAVINDRA GAIKWAD

CS-2021

Under the esteemed guidance of

Ms. MAMATA JADHAV

ASSISTANT PROFESSOR

DEPARTMENT OF COMPUTER SCIENCE

D.G. RUPAREL COLLEGE

(Affiliated to University of Mumbai)

MUMBAI- 400012

YEAR 2019-2020

ACKNOWLEDGEMENT

I am pleased to present **Code Editor** project. I would like to thank our college for providing us with excellent facilities that helped us to complete and present this project. I would also like to thank the staff members and lab assistants for permitting the use computers in the lab as and when required.

I express my deepest gratitude towards my project guide

Ms. Mamata Jadhav for her valuable and timely advice during the various phases in my project. I would also like to thank her for providing with proper facilities, support, having faith in my capabilities and for giving us flexibility in terms of working and reporting schedules.

Finally, I would like to thank everyone who has helped me directly or indirectly in my project.

ABSTRACT

Code Editor which gives an all-in-one platform to edit code from different programming languages such as Python, Java, C++, C and Perl. It also supports scripting languages as HTML, CSS and Tex. The editor is able to isolate programming languages by highlighting syntax of programs. Moreover, editor features in-built tools like auto-completion of variables that are used, incomplete bracket indication through colour-highlight.

Code Editor also provides simple yet powerful features such as a rich text-editing tool, which enables user to add images. Sticky Notes gives user an option to store important notes, which are saved in database. Sticky Notes is a window-less tool that can be dragged around desktop. Code Editor embeds a terminal which gives access over the system. Source Control saves all the files with a date-stamp that allows user to fetch files from previous versions. File Browser has been included to give an overview of project files. PDF viewer has been inculcated to give users a reading-friendly environment. File Encryption and Decryption allows user to securely store files on their machines that would be encrypted by industry standard algorithms and can be decrypted with a passkey generated by the user.

INDEX

❖ Introduction

○ Features

- Text Editor
- Sticky Notes
- File Browser
- PDF Viewer
- Source Control
- Embedded Terminal
- File Security

❖ Gantt Chart

❖ System Analysis

- Existing System
- Proposed System
- Requirement Analysis, Gathering and Planning
- Tools and Technology
- Requirements
- Justification of Platform
- Project Planning and Standards Followed

❖ System Design

- Event Table
- Sequence Diagram
- Class Diagram
- Use Case
- Deployment Diagram
- Database Schema

❖ **Implementation and Use Cases**

- **Sticky Note - Program Description**
- **Application Layout**
- **Code Editor**
- **File Encryption and Decryption**

❖ **Conclusion and Future Work**

- **Conclusion**
- **Future Work**

❖ **References**

Chapter 1

Introduction

Code Editor has been specifically designed for students and professionals that are new to the platforms and concepts of programming. The central idea behind developing the software is to introduce students to a easy-to-use and easy-to-learn, intuitive code editing environment. Code Editor will help students to grasp and accustomed themselves to concepts of programming languages with features such as syntax highlighting, auto completion, use of shortcuts etc.

Students need various reference materials and note-making tools to understand and implement various concepts of programming languages. In order to inculcate this idea, Code Editor is equipped with a rich-text editing tool, a pdf-viewer and sticky notes which gives students an interactive environment where they have access to all the reference material under one roof. Students can get access over whole system using embedded terminal. They can have a overlook of the project files using the File-Browser. Students will also be able to get acquainted with source control, as Code Editor is able to save files using a date-stamp. Keeping security of files a major priority, students can encrypt their files using a passkey which would be handled by Linux i.e. no threat for file tampering or deletion if system is used by another individual.

Features of Code Editor:

- A) Programming.
- B) Text-Editor.
- C) Sticky Notes.
- D) File Browser.
- E) Embedded Terminal.
- F) Source Control.
- G) File Security.
- H) PDF Viewer.

Features

PROGRAMMING

Coding is the central idea of the Code Editor. Students can code in multiple languages under one IDE which helps them to focus on learning aspects of the language. Features such as auto-completion and syntax highlighting would aid students to strengthen their command over the languages. Intuitive shortcuts would make them familiar with use of efficient keyboard usage. Code Editor has an intuitive display of tabs, that may contain codes of different languages.

Toolbar has a 'Run Code' feature which executes the code in current tab and displays output. The lines are numbered on the left side of the screen. On click of the line number, the text-row gets selected which helps to copy, cut and paste code. The Output window is displayed on the bottom of the screen. It would display the output of the code that has been executed or error (if any) in the code.

1. Programming Languages include

1. Java
2. Python
3. C++
4. C
5. C#
6. Perl

2. Scripting Languages include

1. HTML
2. CSS
3. XML

3. Other Languages

1. SQL
2. Tex

TEXT-EDITOR

Text-Editor is a rich text word processor implemented in Python and Qt. Students can use it to open, edit and save HTML-formatted files, format view. Basic formatting, headings, lists and images are supported.

Fonts

The font dropdown list shows the fonts available on the host system, with each font listed by name with a demo of the font.

Font Size can be changed from dropdown list.

Sizes include [7, 8, 9, 10, 11, 12, 13, 14, 18, 24, 36, 48, 64, 72, 96, 144, 288]

- **Styles**

Style handling uses **BOLD**, *Italic*, and underline features. Shortcuts keys for the same are been added.

- **Alignment**

Alignment has Left-Align, Centre-Align, Right-Align and Justified. Shortcuts keys for the same are been added.

- **Opening & Saving files**

Files can be opened and saved using Tool-bar icons or shortcut keys. HTML files can be opened and saved in Text-Editor.

- **File Security**

Files in Text-editor can be saved securely using one passkey.

Sticky Notes

Sticky Notes can be positioned anywhere on the desktop screen. The text can be formatted, and the notes can be resized. It can be easily activated by tapping the Start button. Additional notes can be created by clicking the New Note button, which has a plus ("+") sign on it. Once a note has been opened, the user can start typing. A note can be deleted by clicking the "x" button at the top right of the note. Sticky Notes also supports keyboard shortcuts for different types of formatting.

Database Integration

The contents and position of any individual note is stored in database until its deleted from the desktop.

File Browser

File Browser gives an overview of all the files on the system. It uses tree view hierarchal structure which makes it easy to navigate.

PDF-Viewer

Code Editor uses system's built-in pdf-viewer to display recent and already-opened pdf. It inculcates all the pdf files that are present in the system. Most of references for academic purposes are formatted on pdf, hence PDF-view was incorporated.

Source Control

Source Control is an important aspect in a project. It is very essential to save and being able to access all the previous versions of a file. In case of system failure, incremental feature failure, or any other catastrophic failure, it becomes necessary to use previous files. Also, it helps to keep a track of files if multiple students are working on the projects. The files will be saved with a date-time stamp in a folder named “*Source-Control Code Editor*” in the home directory.

Student needs to provide a “*Project Name*” and “*File Name*” in order to save project in the source-control folder. If the folder with “*Project Name*” exists, the “*File Name*” file will be saved with corresponding date-time stamp. If “*Project Name*” doesn’t exists, it will create a new folder.

Embedded Terminal

Terminal is the most integral part of the Linux OS. Students of various streams have Linux OS in their curriculum and hence a terminal has been embedded in the software. Terminal alongside the File-Browser gives student a interactive Linux environment.

Terminal has root privileges i.e. the tasks could range from renaming a certain file to configuring a server.

File Security

Code Editor gives students an option to secure file with one passkey. Toolbar has an icon of encryption and decryption of files.

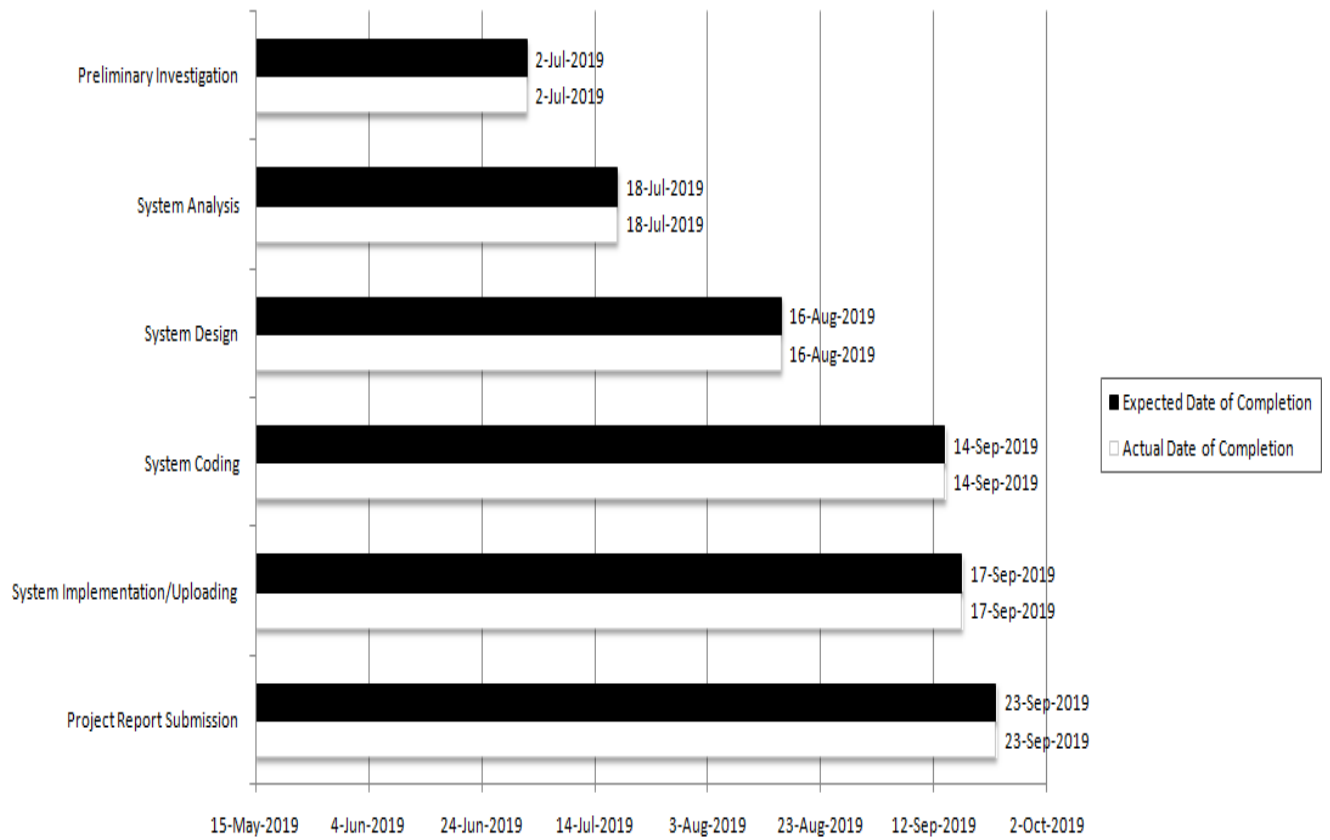
- **Encrypting files**

In order to encrypt files, user needs to give a filename and a passkey. Files are stored with file extension “. *gpg*”.

- **Decrypting files**

Files that are to be decrypted, needs two-key authentication. A *master-key* which is provided by Code Editor is to be entered first and later the passkey of the encrypted file.

Gantt Chart



Chapter 2

System Analysis

2.1 Existing System

Students which are new to the concepts of programming languages are needed to use different IDE's for different programming languages. As those IDE's are developed from different organizations, it becomes difficult to adapt. Different IDE's use different user interfaces, different shortcut keys and sometimes whole different environment. The main aspect of learning the language and fundamental concepts is hindered with the changes. Some of the IDEs require complex configuration and are resource consuming (w.r.t. memory and processing power.)

Students are required to use reference materials from different sources. To incorporate those references, different mediums are required. Most reference books are available in pdf formats which may include diagrams and source-codes. If student wants to store information about particular topic, the pdf needs to be bookmarked. Sharing the necessary information is not intuitive. Instead saving the diagram and contents in Text-Editor and executing the code in Code Editor gives student a hassle free approach.

Project planning and project management are important aspects. Modern IDE's do provide options for source control, but it requires considerable amount of time to learn and use the features. Moreover, in institutions where more than one individual work on same system, file security is a major issue. File may get tampered or even deleted by accidents which may result in permanent loss of data.

Limitations of Existing System

- Different programming languages require different IDE's which are difficult to adapt and sometimes resource consuming.
- No integrated medium for students to develop and store reference material.
- Complicated version control systems
- No integrated way to secure files.
- The tools which are present either required to be paid or require considerable time to learn and use them.

2.2 Proposed System

1. Code Editor

A lightweight code editor which supports syntax highlighting of different languages. Tools such as auto-completion, numbered-lines and bracket colour highlighting makes the IDE user-friendly. Tab based user interface makes execution of different programming languages easy and hassle free under one window.

2. Text Editor/PDF viewer/Sticky Notes

Students can extract the contains of a reference book from pdf and save the images and contents in the Text-Editor. Sticky Notes helps to plan in-advance steps for project, reminders or any last minute notes. PDF offers higher readability experience and also the snippets of code from pdfs can be tested in Code Editor.

3. Embedded Terminal/File Browser

Linux Terminal gives whole access over the system. Integrating it with File Browser would give student an overview of the project as well as the system. It can prove to be a efficient tool to learn entry-level Linux Commands.

4. Source Control/File Security

Source Control feature provides a hassle-free way to save the files. The files are saved in a different folder with the date-time stamp.

File security provides two-way authentication, where a master password is used as well as passkey which is used to encrypt file is also required.

5. Opensource

The software is developed on Linux and uses libraries which are available free of cost. The software abides to all Freedoms of open source software.

2.3 Requirement Analysis, Gathering and Planning

The software was developed with a simple aim, i.e. to develop an IDE which is lightweight, user-friendly, easily adaptable, intuitive, and can run programming languages such as C, C++, Python, Java etc. Also, a platform where reference materials could be accessed easily and could be integrated with the programming aspects of student's curriculum was needed. A series of thorough interactive sessions with students from S.Y. and T.Y. B.Sc. CS prompted the aforementioned challenges while learning concepts of programming languages.

Through the same sessions and guidance of teachers, the features that would solve the challenges were enlisted. Research for implementation of those features led to insightful tools that made the software possible. Most expected features were prioritised, and later the add-on features were inculcated in the software. The software had to be a standard industry grade software and hence some features required intricate development.

2.4 Tools and Technology

Tools

1. IntelliJ PyCharm Community Edition (IDE).
 - Open Source.
 - Industry Standard.
 - Vast Online Community.
 - Support from IntelliJ.

2. QT designer (UI design).
 - UI development tool for QT.
 - Drag and Drop features.
 - Designer View and Source Code View available.

3. Git (Version Control)
 - Distributed Version control system
 - Free to use
 - Extensive file upload limits(both in size and in number)
 - Vast community

Languages

Python

Python is a simple but powerful object-orientated language. Its simplicity makes it easy to learn, but its power means that large and complex applications can be created. Its interpreted nature means that Python programmers are very productive because there is no edit/compile/link/run development cycle.

- Open Source
- High Readability Factor
- Supports OOPS
- Supports GUI
- Extensive 3rd party libraries
- Vast community support

Technology

- **PyQT5**

PyQt5 is a set of Python v2 and v3 bindings for The Qt Company's Qt application framework and runs on all platforms supported by Qt including Windows, OS X, Linux, iOS and Android. The bindings are implemented as a set of Python modules and contain over 1,000 classes.

PyQt5 brings together the Qt C++ cross-platform application framework and the cross-platform interpreted language Python. Qt is more than a GUI toolkit. It includes abstractions of network sockets, threads, Unicode, regular expressions, SQL databases, SVG, OpenGL, XML, a fully functional web browser, a help system, a multimedia framework, as well as a rich collection of GUI widgets. Qt classes employ a signal/slot mechanism for communicating between objects that is type safe but loosely coupled making it easy to create re-usable software components.

PyQt5 combines all the advantages of Qt and Python. A programmer has all the power of Qt, but is able to exploit it with the simplicity of Python.

- **PySide2**

PySide2 is the official Python module from the Qt for Python project, which provides access to the complete Qt 5.12+ framework. The Qt for Python project is developed in the open, with all facilities you'd expect from any modern OSS project such as all code in a git repository and an open design process

- **QsciScintilla**

QScintilla is a text editing component for the Qt application framework written in the C++ programming language. It is a wrapper for the Scintilla text editing component created by Neil Hodgson also written in the C++ programming language. The Qt application framework is a set of objects, also referred to as widgets, that help making GUI (Graphical-User-Interface) and other types of applications easier and is cross-platform.

Some QScintilla features:

- Built-in syntax highlighting for more than 30 programming languages.
- Text styling: underlining, highlighting.
- Autocompletion functionality.
- UTF-8 support

- **GnuPrivacyGuard**

GnuPG is a command line tool without any graphical user interface. It is an universal crypto engine which can be used directly from a command line prompt, from shell scripts, or from other programs. Therefore, GnuPG is often used as the actual crypto backend of other applications.

Database and its Design

- **Choice of Database**

SQLite

SQLite is the most used database engine in the world. SQLite is built into all mobile phones and most computers and comes bundled inside countless other applications that people use every day.

Features:

- Transactions are atomic, consistent, isolated, and durable (ACID) even after system crashes and power failures.
- Zero-configuration - no setup or administration needed.
- Full-featured SQL implementation with advanced capabilities like partial indexes, indexes on expressions, JSON, common table expressions, and window functions. (Omitted features)
- A complete database is stored in a single cross-platform disk file. Great for use as an application file format.
- Simple, easy to use API.
- Self-contained: no external dependencies.
- Cross-platform: Android, iOS, Linux, Mac and Windows are supported out of the box. Easy to port to other systems.

- **Object Relational Mapper**

SQLAlchemy

SQLAlchemy is the Python SQL toolkit and Object Relational Mapper that gives application developers the full power and flexibility of SQL. It provides a full suite of well-known enterprise-level persistence patterns, designed for efficient and high-performing database access, adapted into a simple and Pythonic domain language.

2.5 Requirements

- **Software Requirements**

Linux Ubuntu 18.04.3 LTS and above

Python Version 3.6 and above

PyCharm 2019.2.2 (Community Edition) and above

PyQT5 Version 5.6 and above

Pyside2 Version 5.13 and above

QScintilla Version 2.11.2 and above

SQLAlchemy 1.3.6 and above

Sqlite3

- **Hardware Requirements**

A device which is able to run Linux Ubuntu 18.04.3 LTS

2.6 Justification of Platform

Project Application Type: **Desktop application.**

OS used: **Linux**

Distribution: **Ubuntu**

Version: **18.04.3 LTS**

- **Desktop IDE's**

- Desktop applications are almost always faster than any other type of applications.
- Desktop IDEs have the edge when it comes to the speed of opening files, editing, and loading the editor, as well.
- The sheer array of tool diversity in desktop IDEs is something else that can't be matched by browser- based alternatives.
- With desktop IDEs you can use any local tool on your codebase, but this isn't possible for browser-based IDEs.
- The browser environment simply imposes too many restrictions on what an online IDE can do, and those limitations are going to be felt by developers.

- **LINUX**

- Open source
- Compatibility
- Low cost
- Security
- Fast and easy installation
- Stability
- Performance
- Flexibility
- Multitasking

2.7 Project Planning and Standards Followed

- Various stages of projects were developed in incremental approach, following Incremental Model of SDLC
- For python, the PEP-8 standard style guide was followed.
- The software was implemented with an eye for maintainable and healthy code.
- Software was designed keeping best recommended practices wherever possible, without exceeding any deadlines or overusing any system resources.

Chapter 3

System Design

List of Tables and Figures

- ❖ STICKY NOTES – PROGRAM SNIPPET
- ❖ EVENT TABLE
- ❖ SEQUENCE DIAGRAM
- ❖ CLASS DIAGRAMS
- ❖ USE CASE DIAGRAMS
- ❖ DEPLOYMENT DIAGRAM
- ❖ DATABASE TABLE

Program Snippet

➤ ORM Class for Sticky Notes

```
Base = declarative_base()

class Note(Base):
    __tablename__ = 'note'
    id = Column(Integer, primary_key=True)
    text = Column(String(1000), nullable=False)
    x = Column(Integer, nullable=False, default=0)
    y = Column(Integer, nullable=False, default=0)

engine = create_engine('sqlite:///notes.db')
# Initialize the database if it is not already.
if not engine.dialect.has_table(engine, "note"):
    Base.metadata.create_all(engine)

# Create a session to handle updates.
Session = sessionmaker(bind=engine)
session = Session()

_ACTIVE_NOTES = {}
```

➤ Note Window

```
class MainWindow(QMainWindow, Ui_MainWindow):
    def __init__(self, *args, obj=None, **kwargs):
        super(MainWindow, self).__init__(*args, **kwargs)
        self.setupUi(self)
        self.setWindowFlags(self.windowFlags() | Qt.FramelessWindowHint | Qt.WindowStaysOnTopHint)
        self.show()

        # Load/save note data, store this notes db reference.
        if obj:
            self.obj = obj
            self.load()
        else:
            self.obj = Note()
            self.save()

        self.closeButton.pressed.connect(self.delete_window)
        self.moreButton.pressed.connect(create_new_note)
        self.textEdit.textChanged.connect(self.save)

        # Flags to store dragged-dropped
        self._drag_active = False

    def load(self):
        self.move(self.obj.x, self.obj.y)
        self.textEdit.setHtml(self.obj.text)
        _ACTIVE_NOTES[self.obj.id] = self

    def save(self):
        self.obj.x = self.x()
        self.obj.y = self.y()
        self.obj.text = self.textEdit.toHtml()
```

```

def mousePressEvent(self, e):
    self.previous_pos = e.globalPos()

def mouseMoveEvent(self, e):
    delta = e.globalPos() - self.previous_pos
    self.move(self.x() + delta.x(), self.y()+delta.y())
    self.previous_pos = e.globalPos()

    self._drag_active = True

def mouseReleaseEvent(self, e):
    if self._drag_active:
        self.save()
        self._drag_active = False

def delete_window(self):
    result = QMessageBox.question(self, "Confirm delete", "Are you sure you want to delete this note?")
    if result == QMessageBox.Yes:
        session.delete(self.obj)
        session.commit()
        self.close()

if __name__ == '__main__':
    app = QApplication(sys.argv)
    app.setApplicationName("Sticky Note")
    app.setStyle("Fusion")

    # Custom brown palette.
    palette = QPalette()
    palette.setColor(QPalette.Window, QColor(188,170,164))
    palette.setColor(QPalette.WindowText, QColor(121,85,72))
    palette.setColor(QPalette.ButtonText, QColor(121,85,72))
    palette.setColor(QPalette.Text, QColor(121,85,72))
    palette.setColor(QPalette.Base, QColor(188,170,164))
    palette.setColor(QPalette.AlternateBase, QColor(188,170,164))
    app.setPalette(palette)

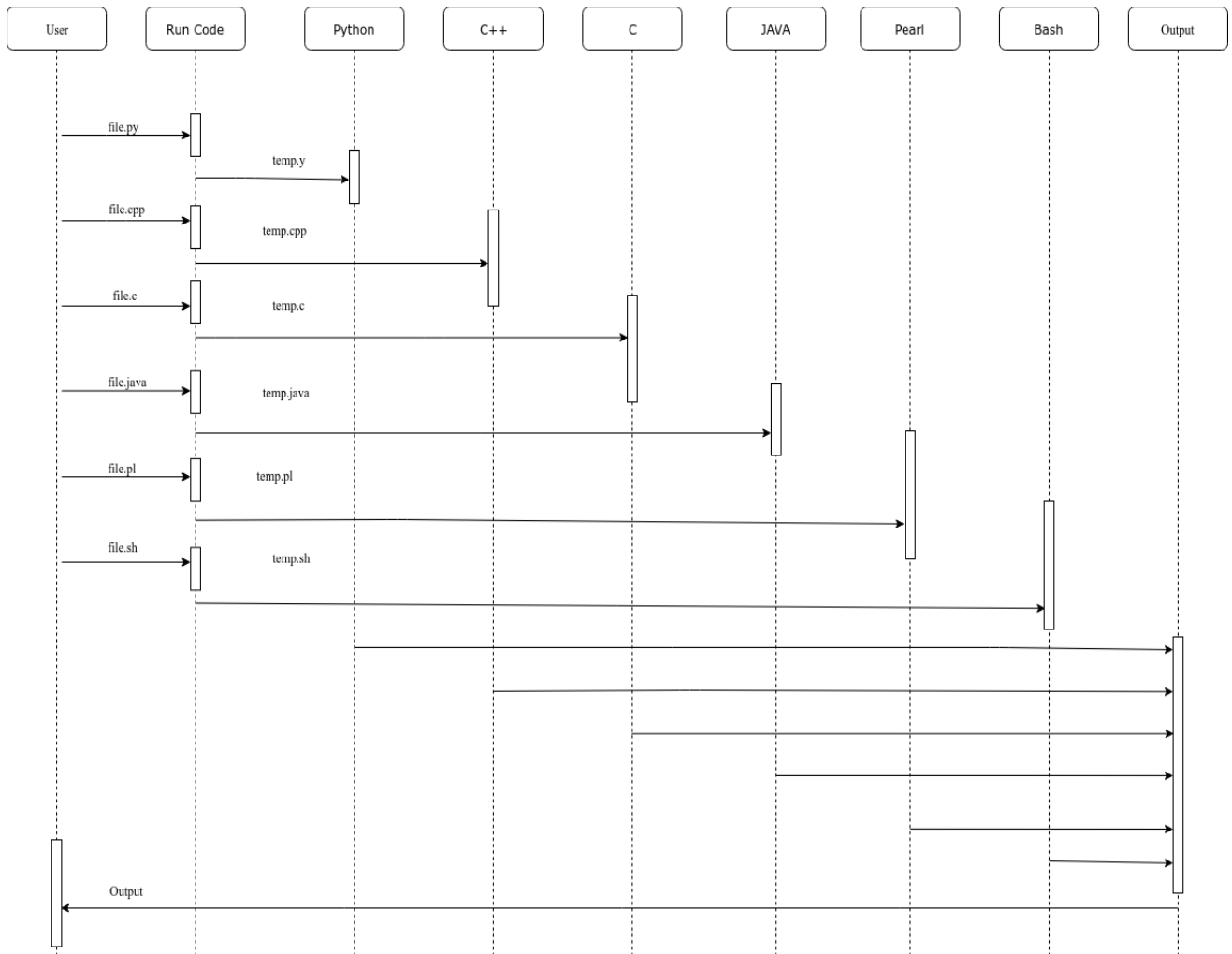
    existing_notes = session.query(Note).all()
    if len(existing_notes) == 0:
        MainWindow()
    else:
        for note in existing_notes:
            MainWindow(obj=note)
    app.exec_()

```

• Event Table

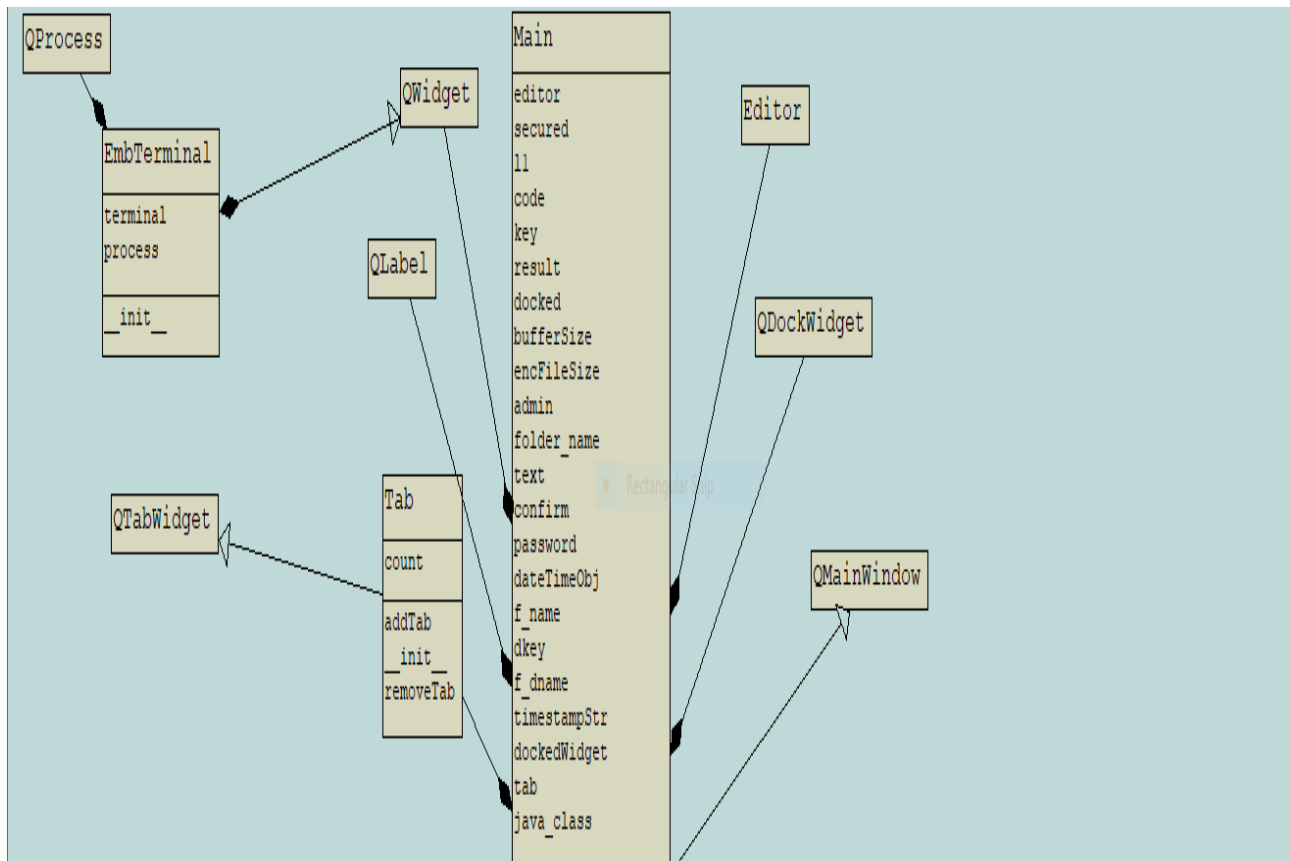
Event	Trigger	Source	Activity	Response	Destination
Text Editor	Text Editor-window	User	Create, edit, save textual content.	Save created file	User
Sticky Notes	Notes-windowless frame	User	Create, save and delete notes	Data storage in database	User
PDF viewer	Document Viewer-window	User	Open pdf's	Recent pdfs are displayed.	User
Embedded Terminal	Urxvt terminal-docked window	User	System control from Embedded Terminal	Terminal opens within the directory	User
File Browser	File Browser-window	User	Overview of files	Files in hierarchical view are displayed	User
Encrypt File	Prompt File details	User	Secure Files with passkey	File created with. gpg	User
Decrypt File	Prompt Authentication	User	View Encrypted files with passkey	Original file retained	User
Source Control	Prompt File details	User	Save Files with Date-time Stamp	Files saved in project folder	User
Run Code	Output -docked window	User	Execute the program for desired output	Output or Error displayed .	User

Sequence Diagram

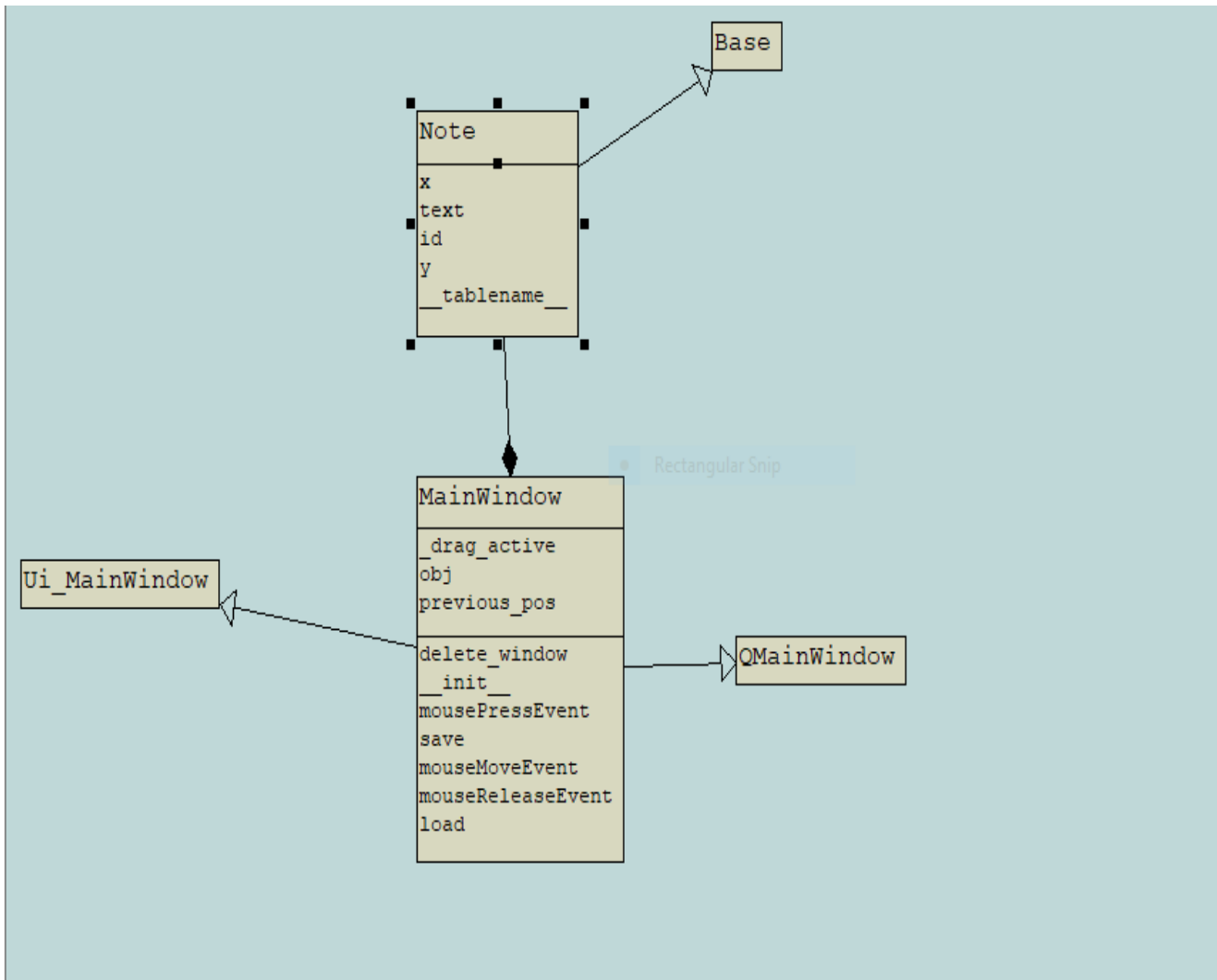


• Class Diagram

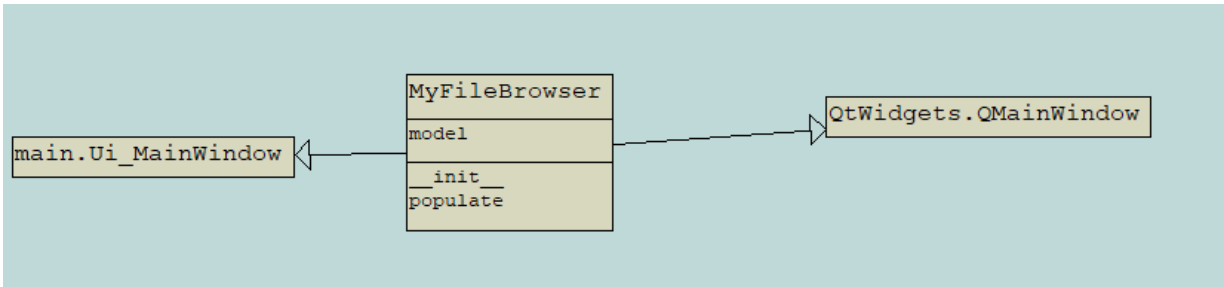
Code Editor



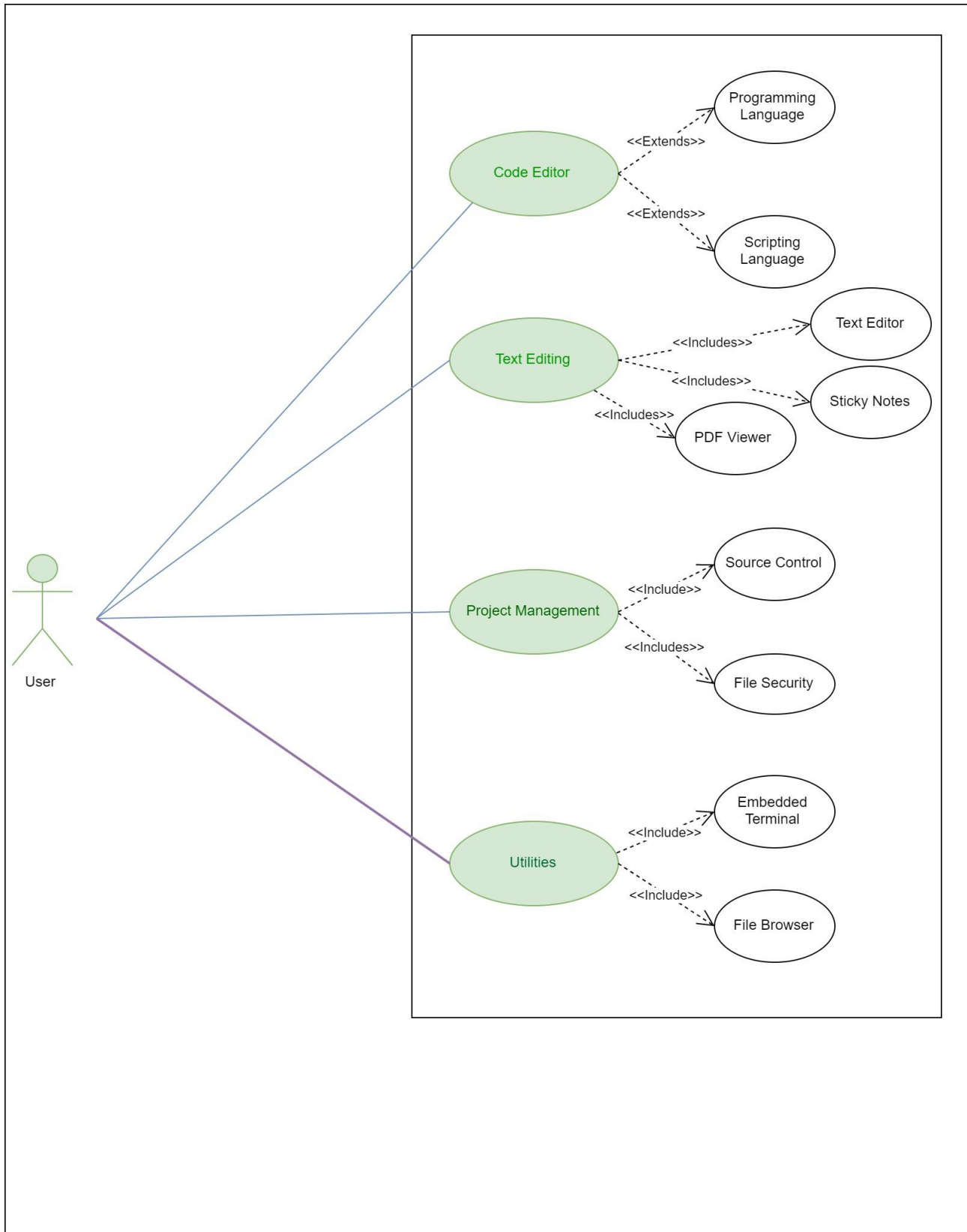
Sticky Notes



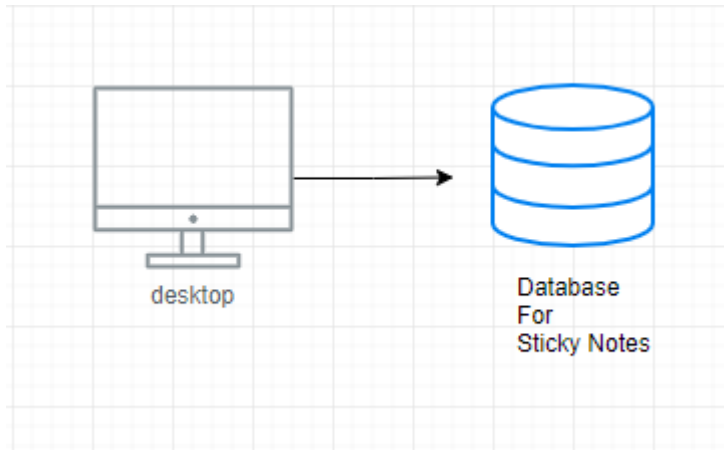
File Browser



• Use Case



- **Deployment Diagram**



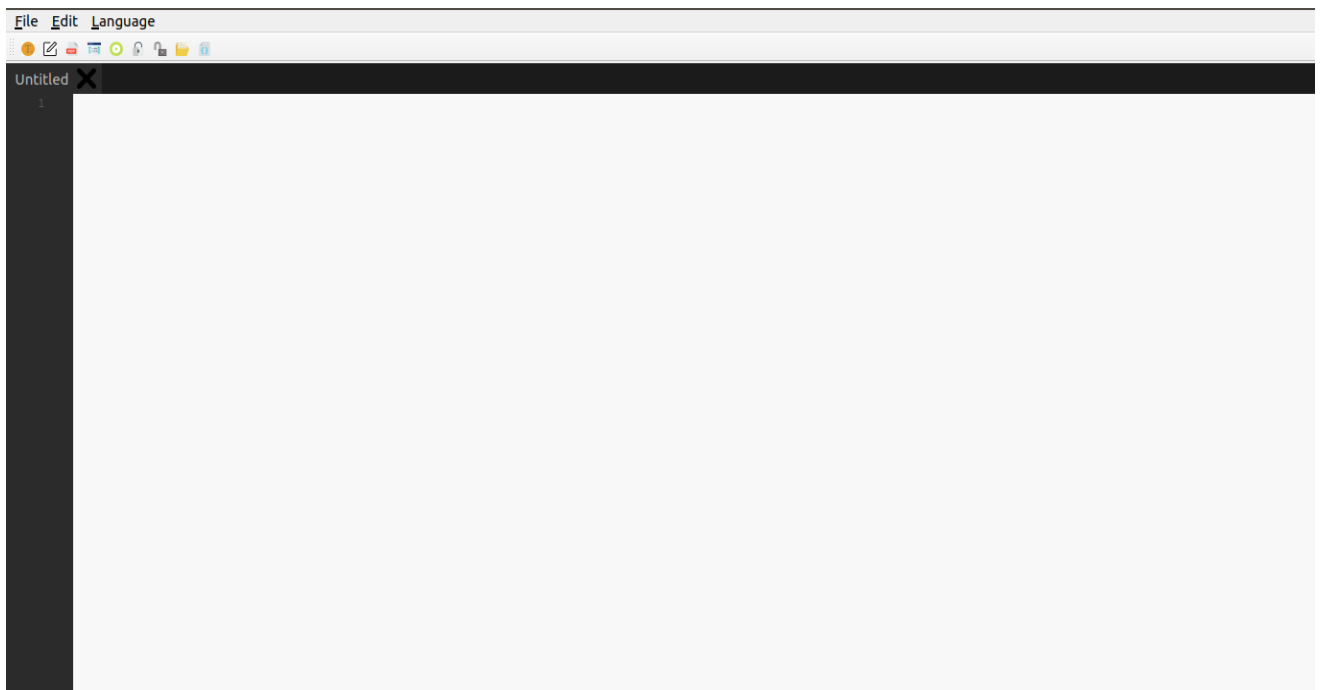
- **Database Table**

note	
id	INTEGER
text	VARCHAR(1000)
x	INTEGER
y	INTEGER

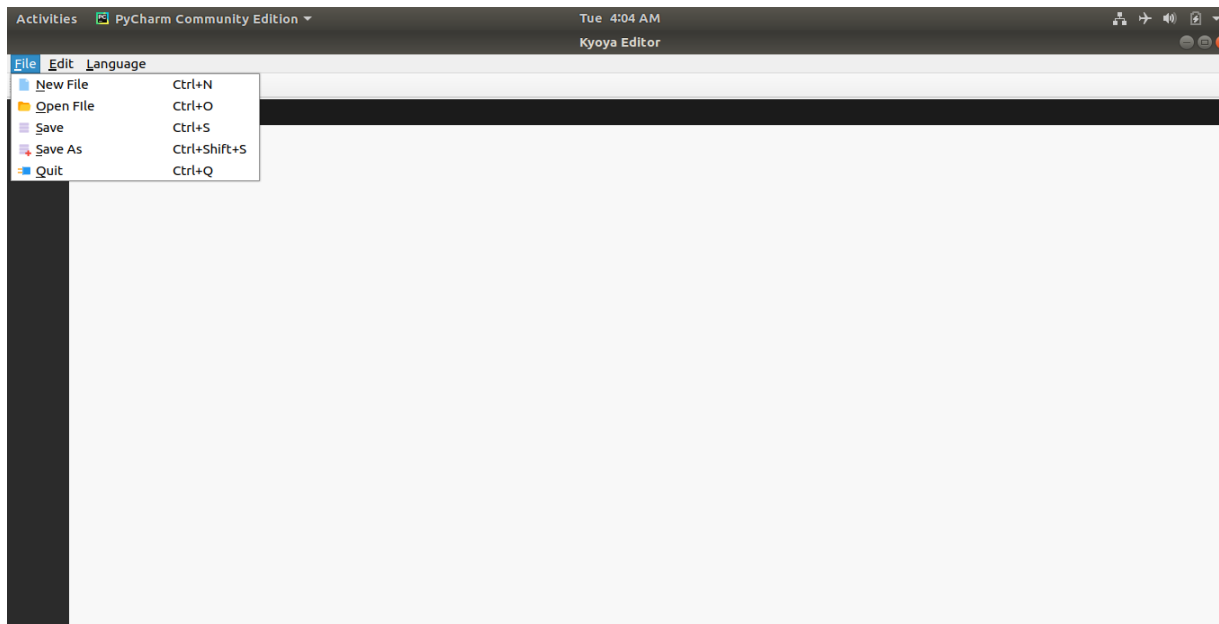
Chapter 4

Implementation and Use Cases

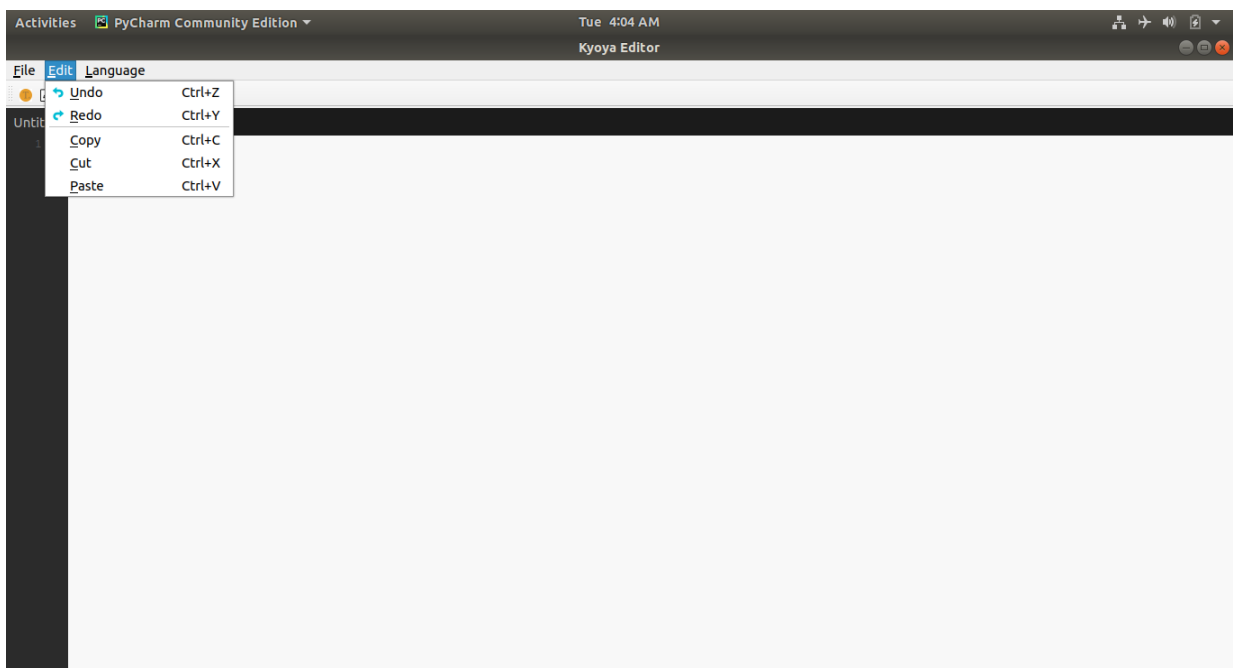
- Application Layout



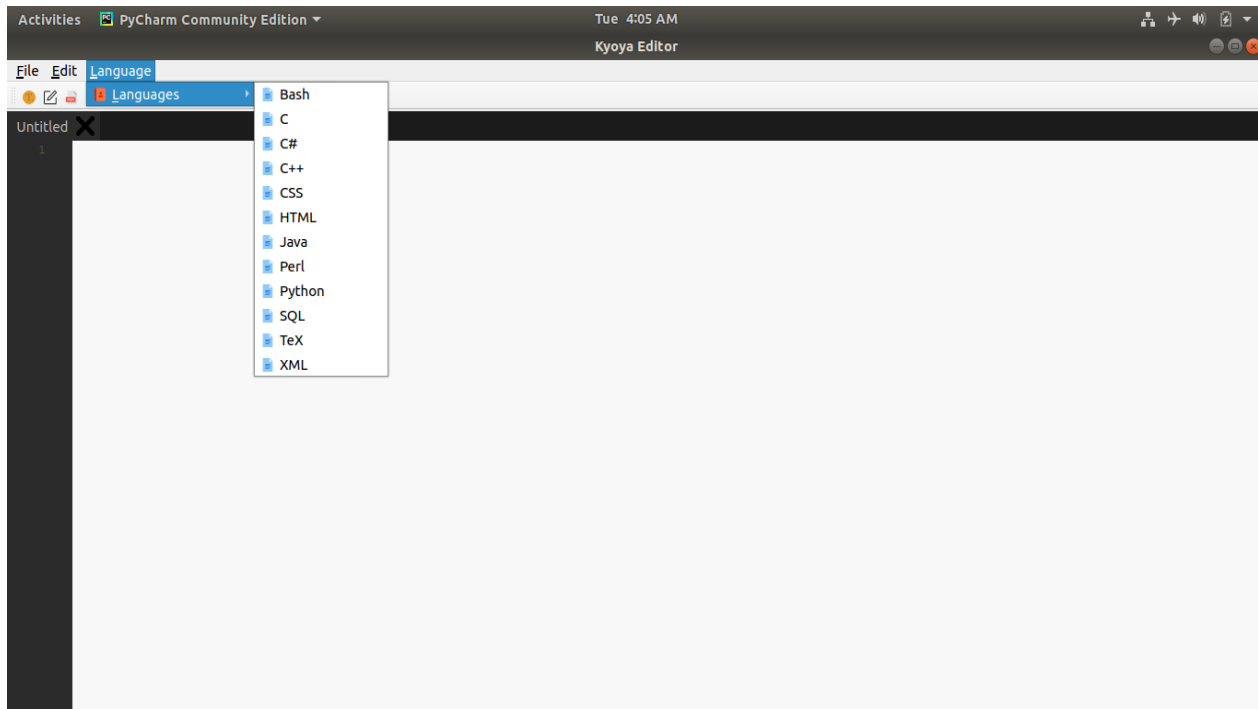
- File Menu



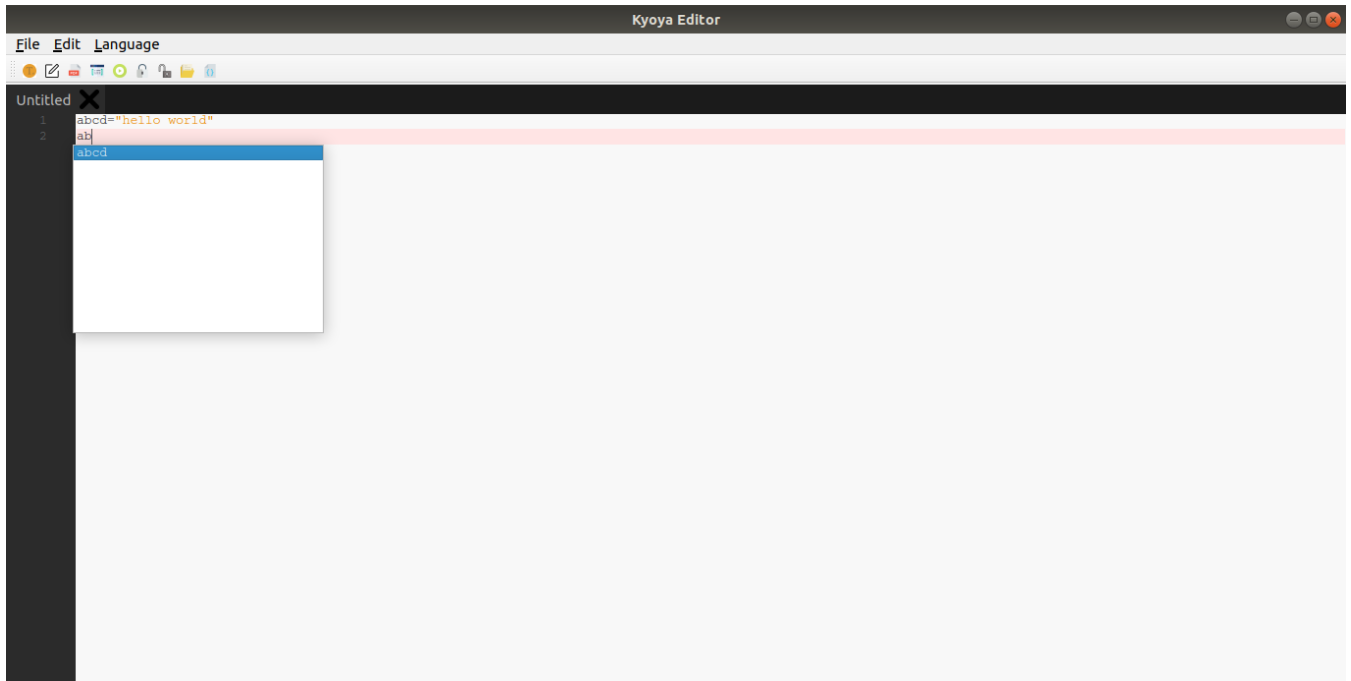
- Edit Menu



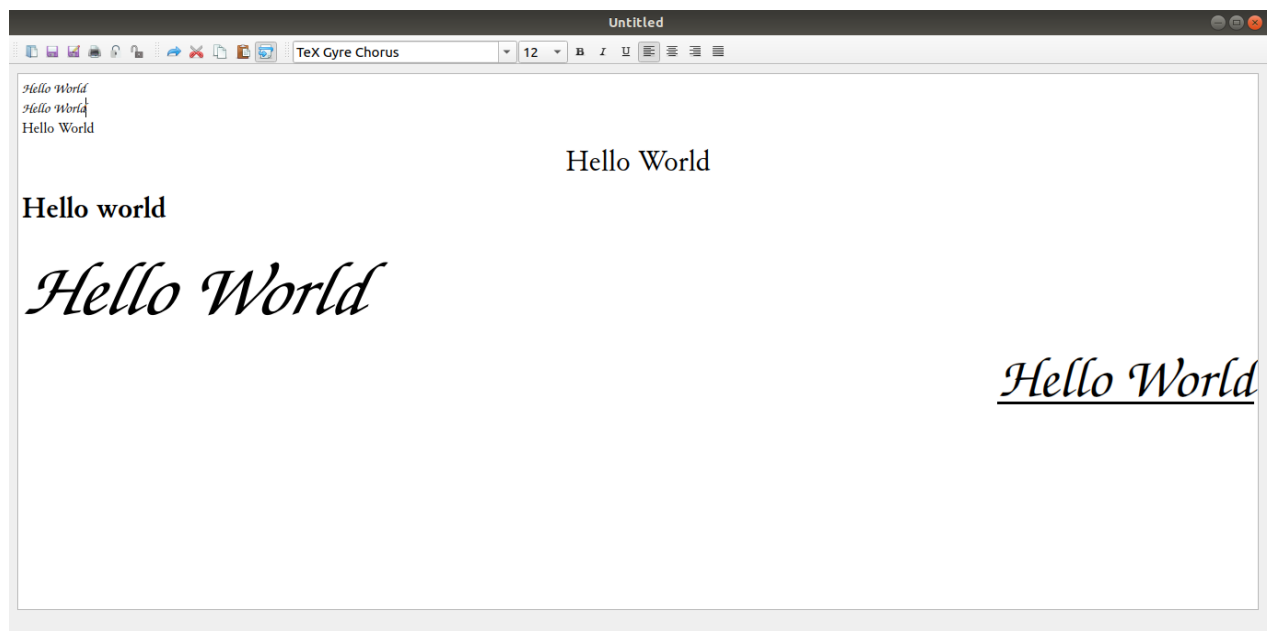
- Languages



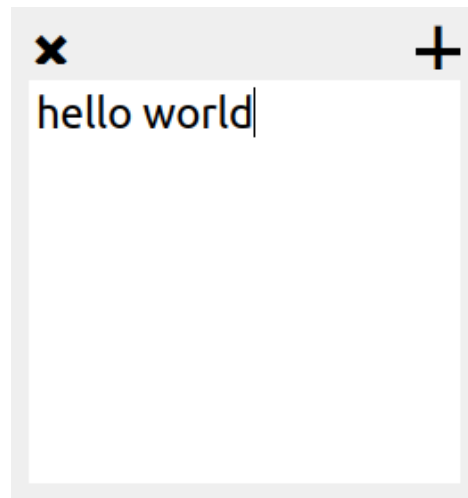
- Auto-Completion



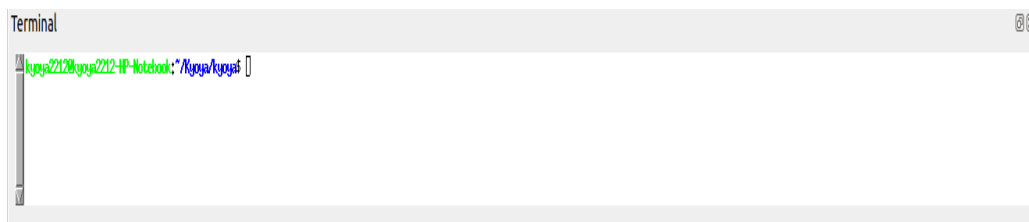
- Text Editor



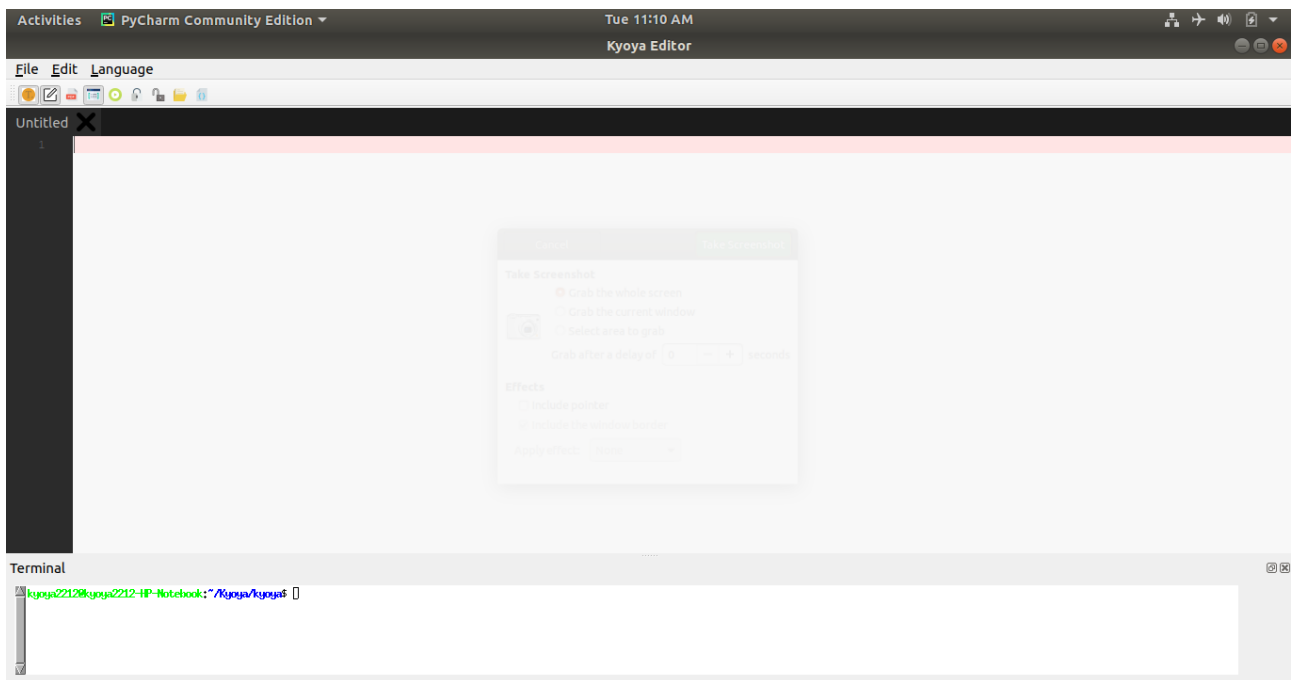
- Sticky Notes



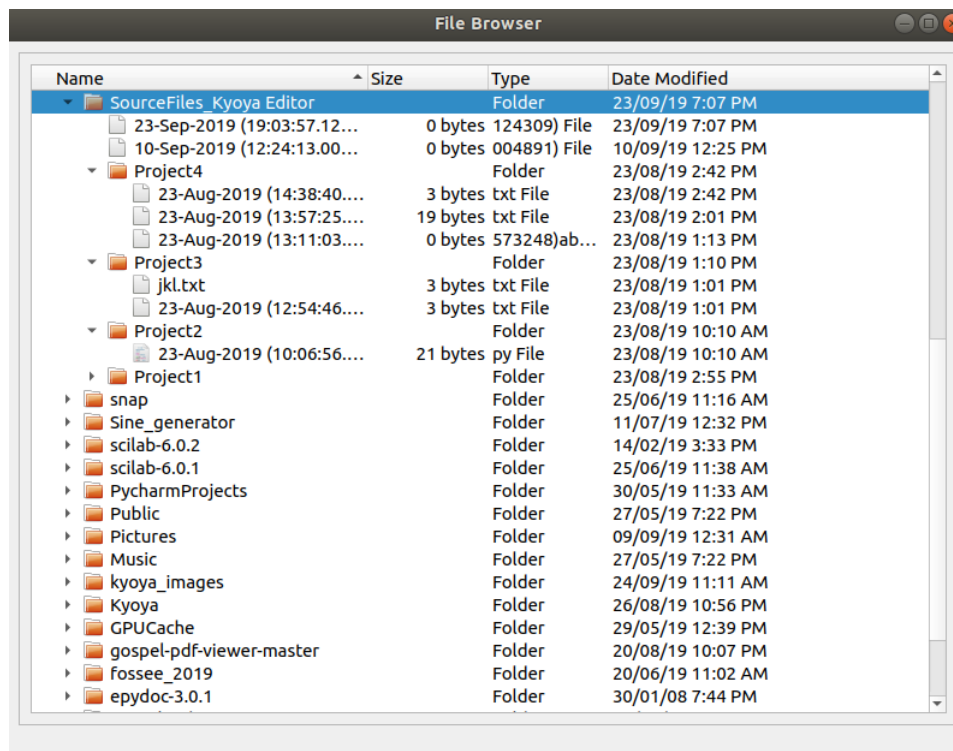
- Terminal



- Embedded Terminal

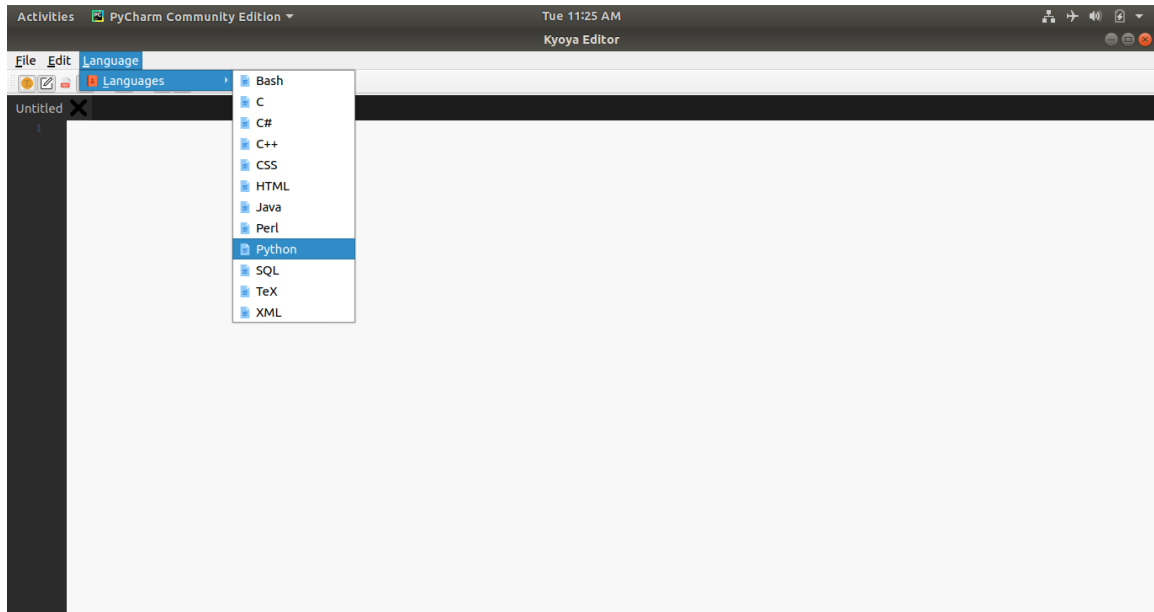


- File Browser

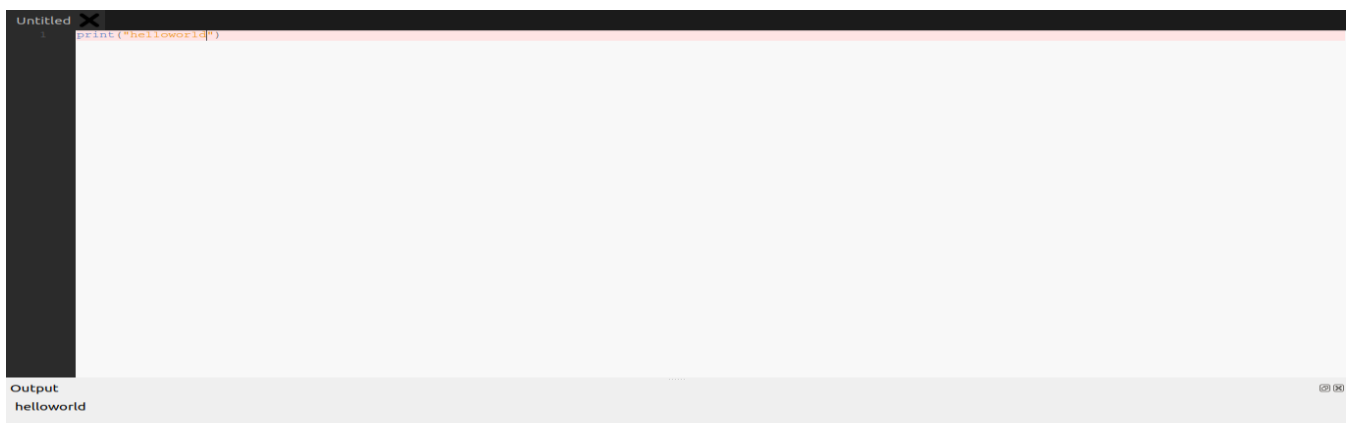


Code Editor Testing

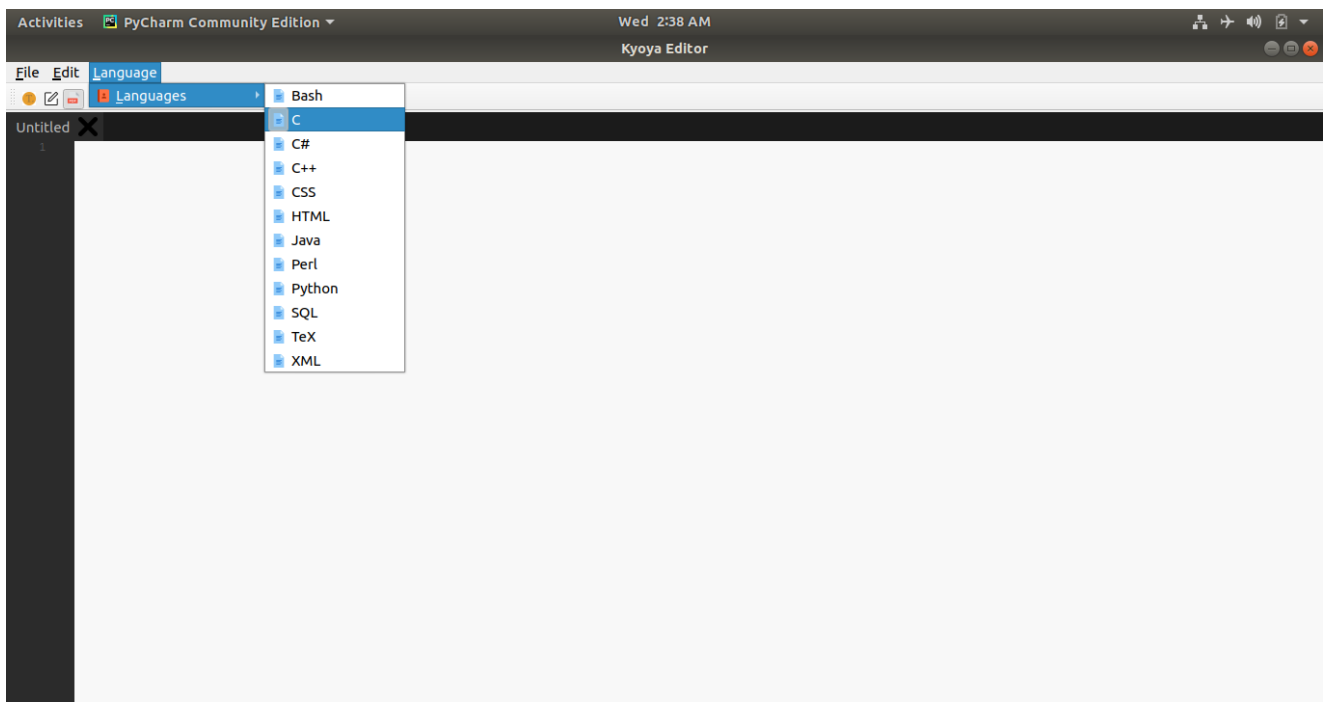
- Python



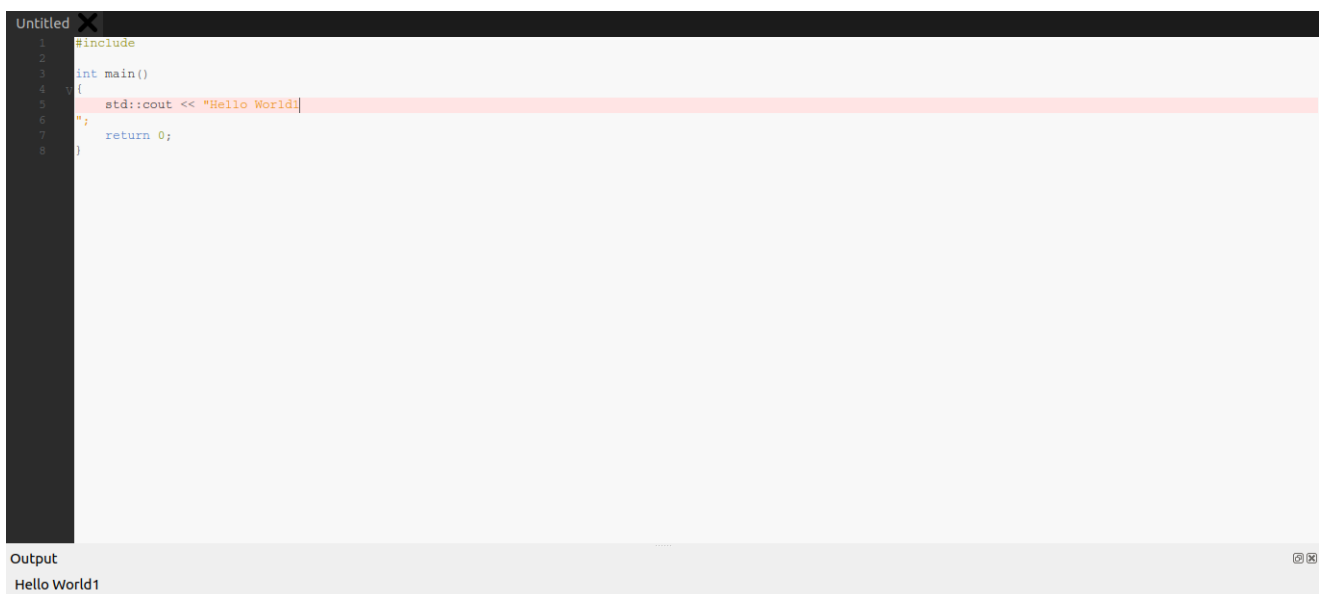
Output



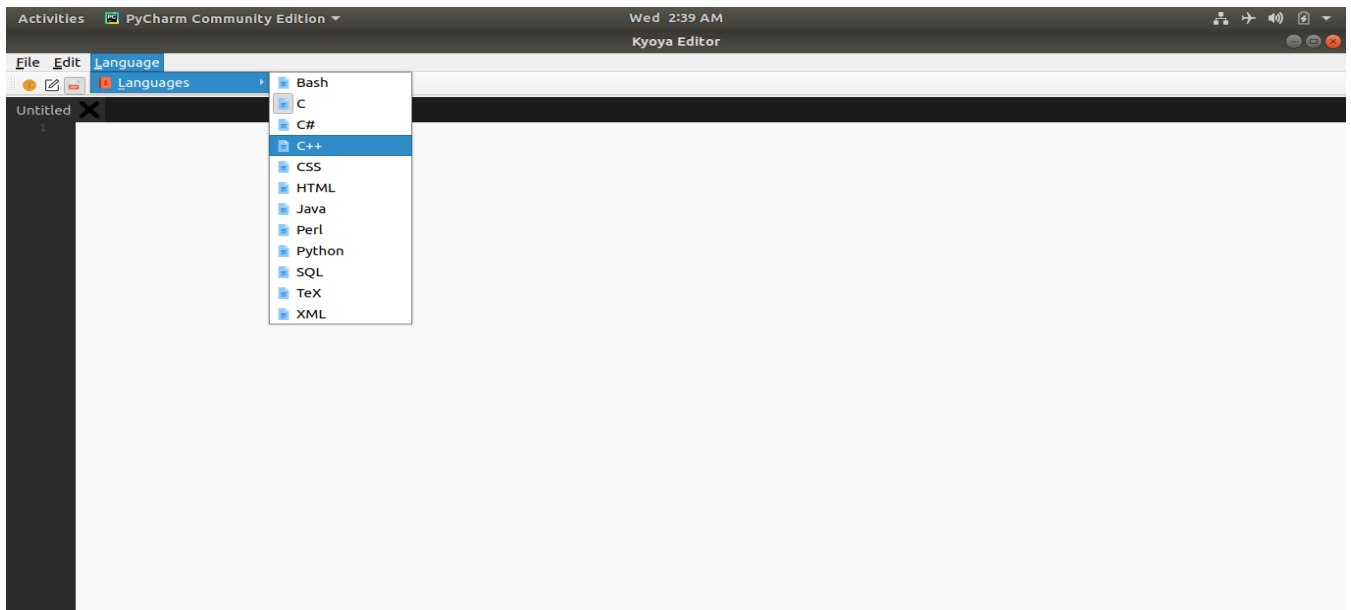
• C



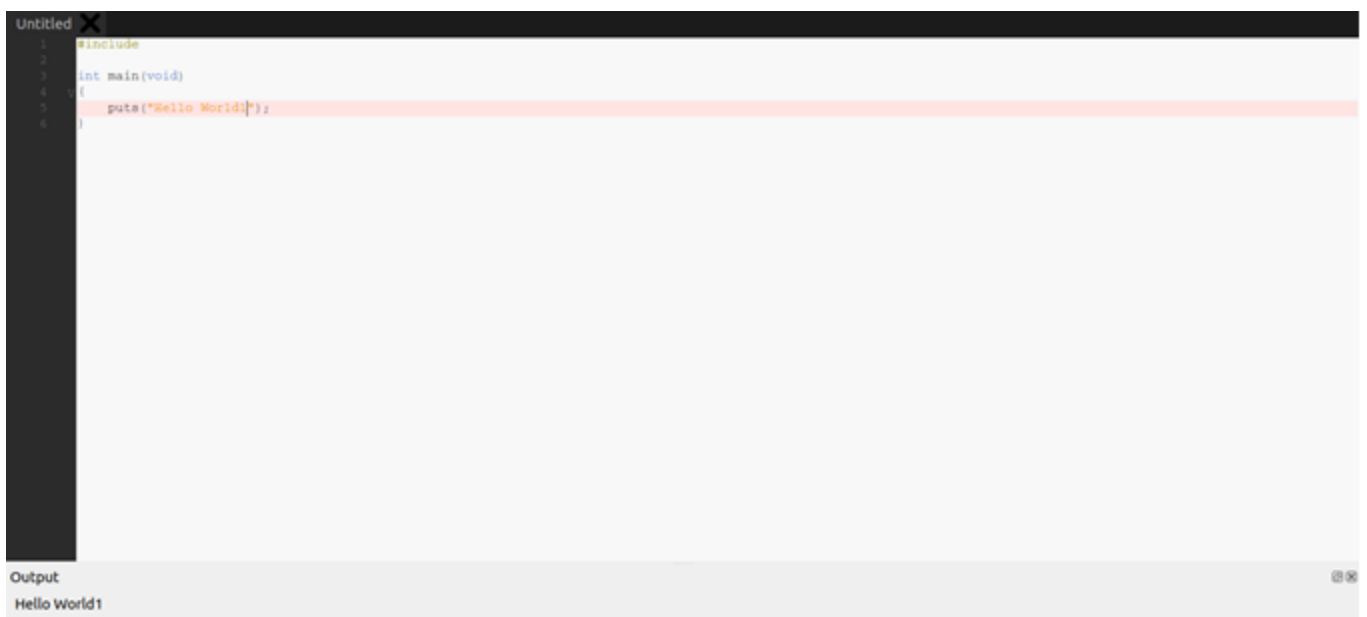
Output



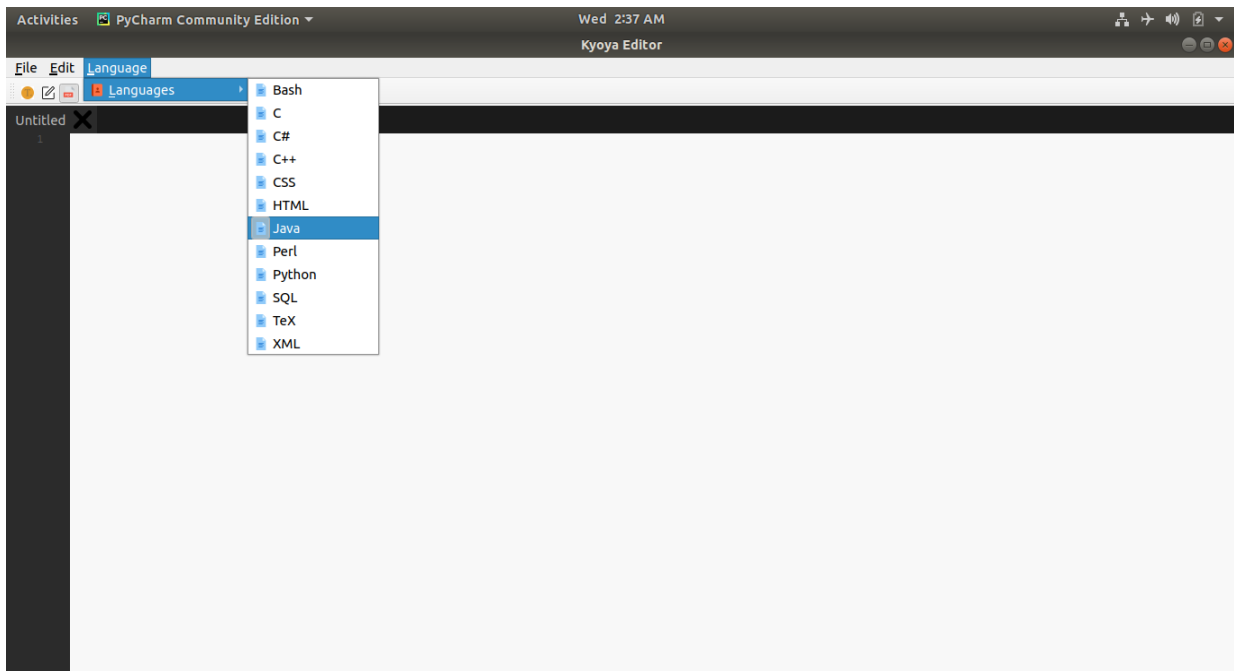
• C++



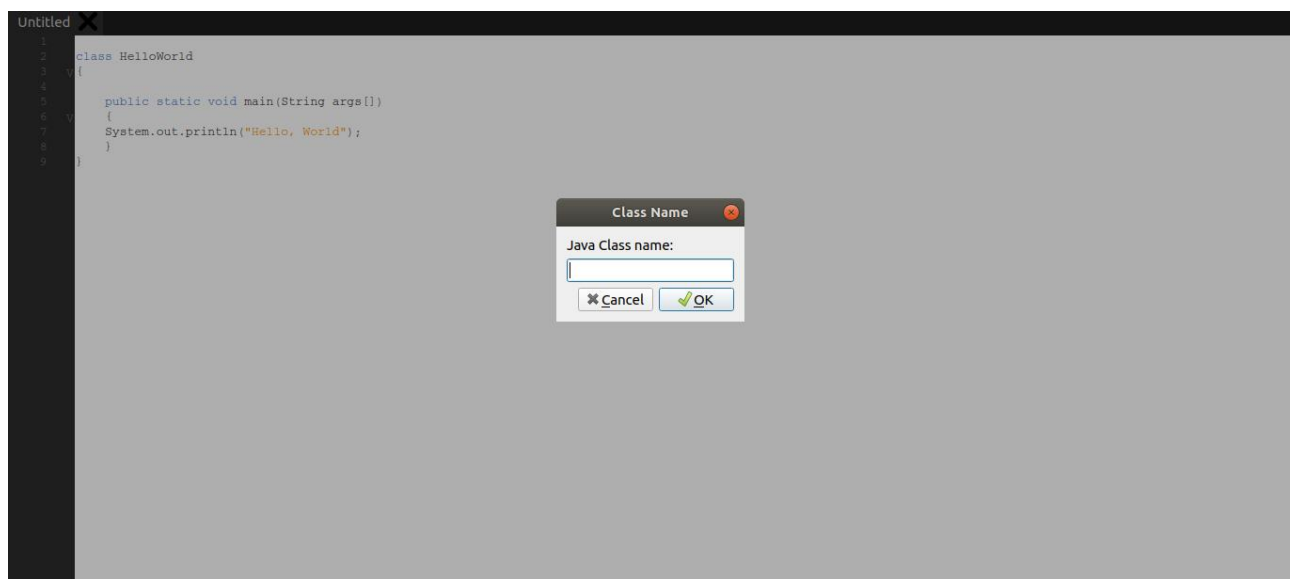
Output



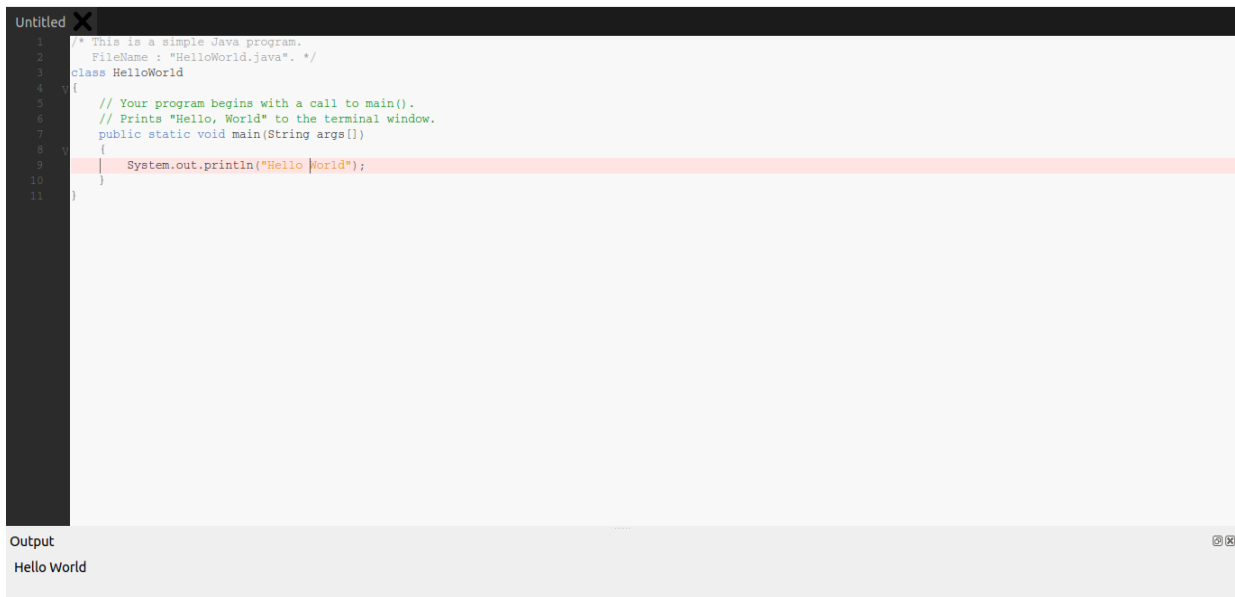
• Java



Class name prompt



Output

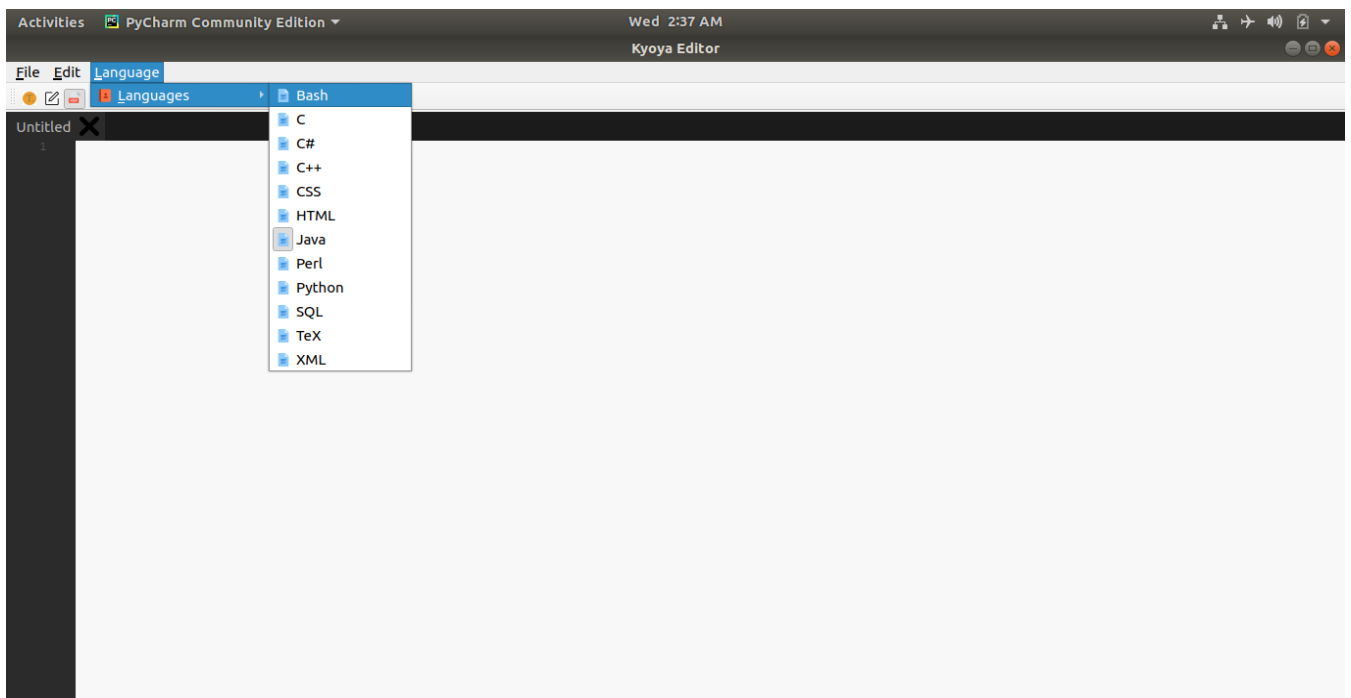


The image shows a screenshot of a code editor window titled "Untitled" with a close button. The editor contains a Java program. The code is as follows:

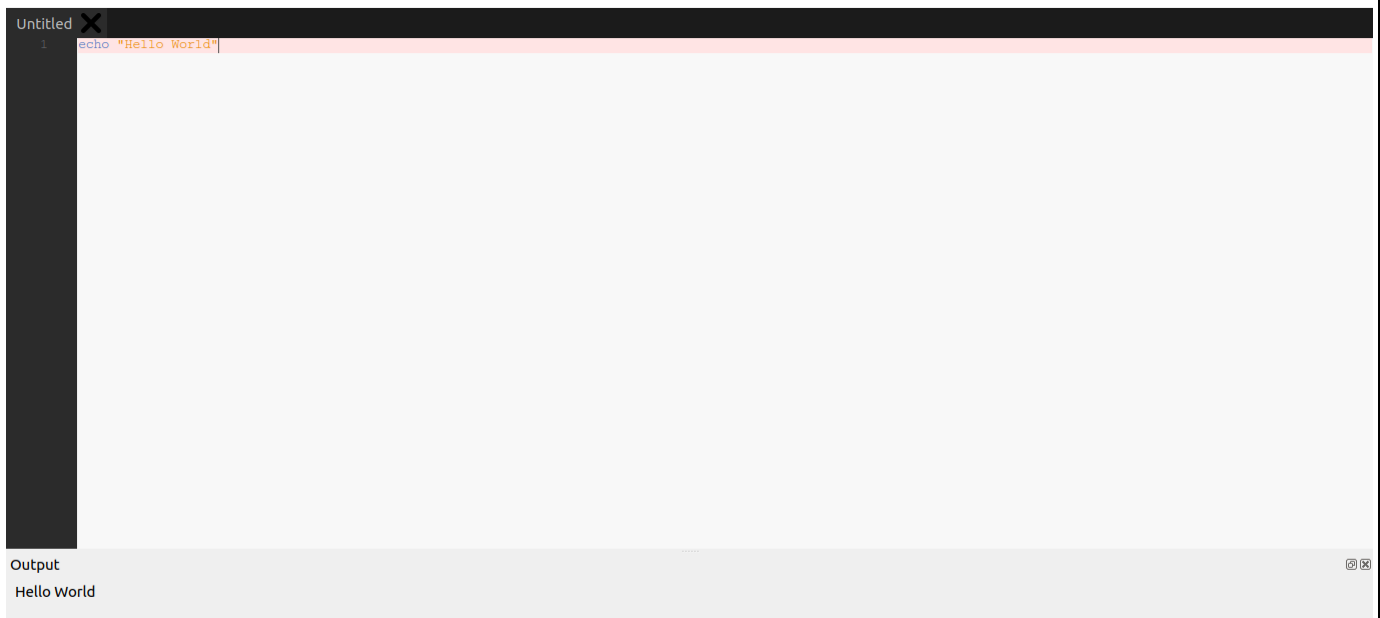
```
1  /* This is a simple Java program.
2     FileName : "HelloWorld.java". */
3  class HelloWorld
4  {
5      // Your program begins with a call to main().
6      // Prints "Hello, World" to the terminal window.
7      public static void main(String args[])
8      {
9          System.out.println("Hello World");
10     }
11 }
```

The line `System.out.println("Hello World");` is highlighted in red. Below the code editor, there is an "Output" section that displays the text "Hello World".

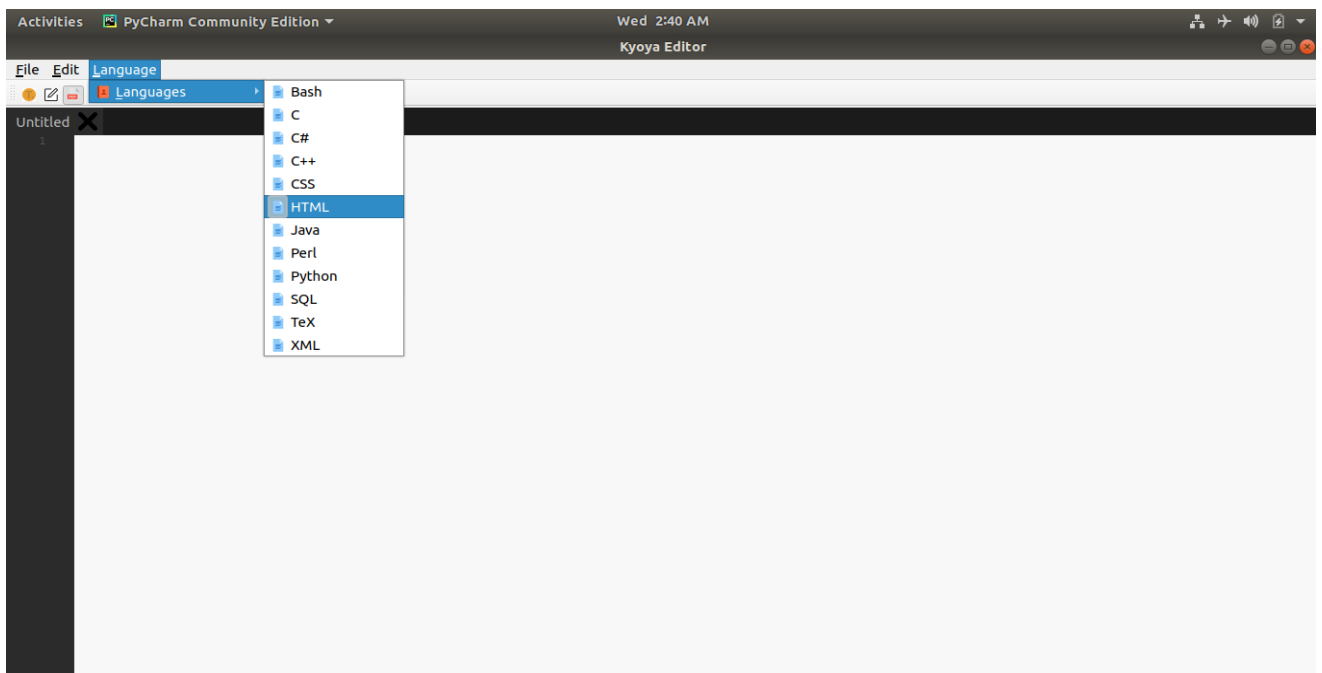
• Bash



Output



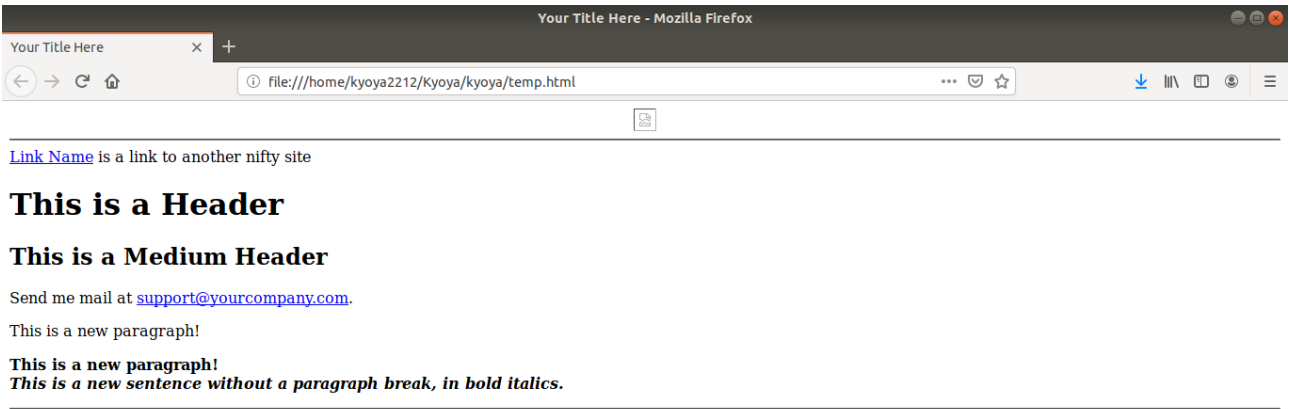
• HTML



Output

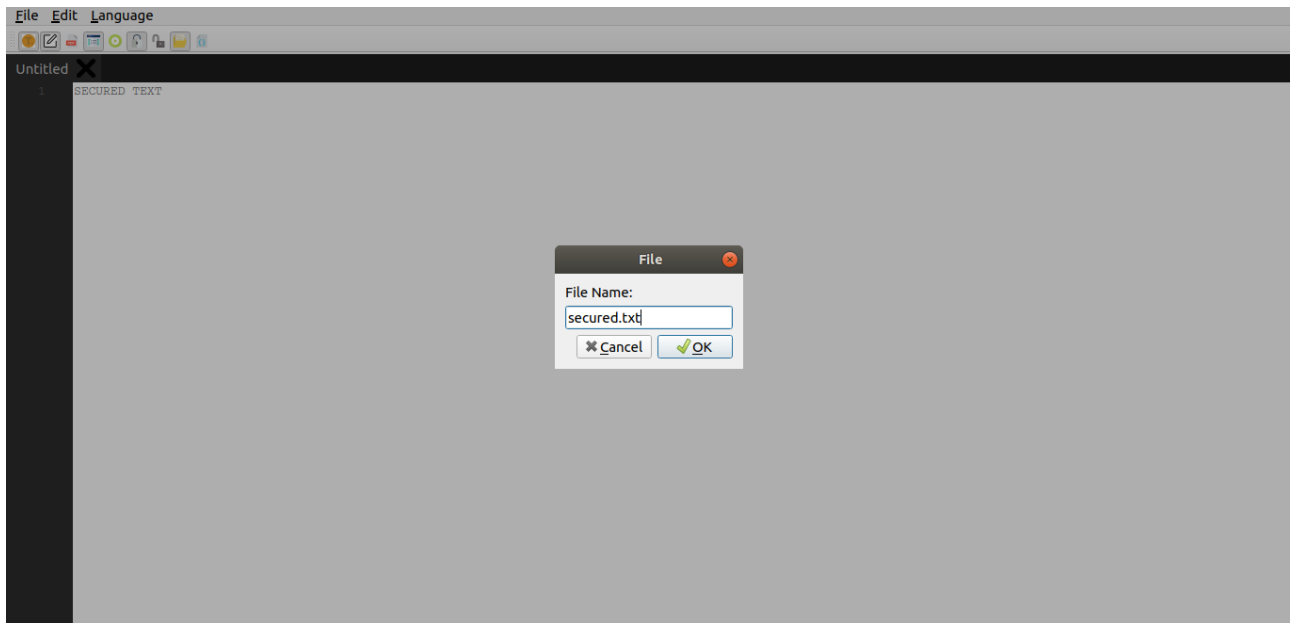


Browser-Output:

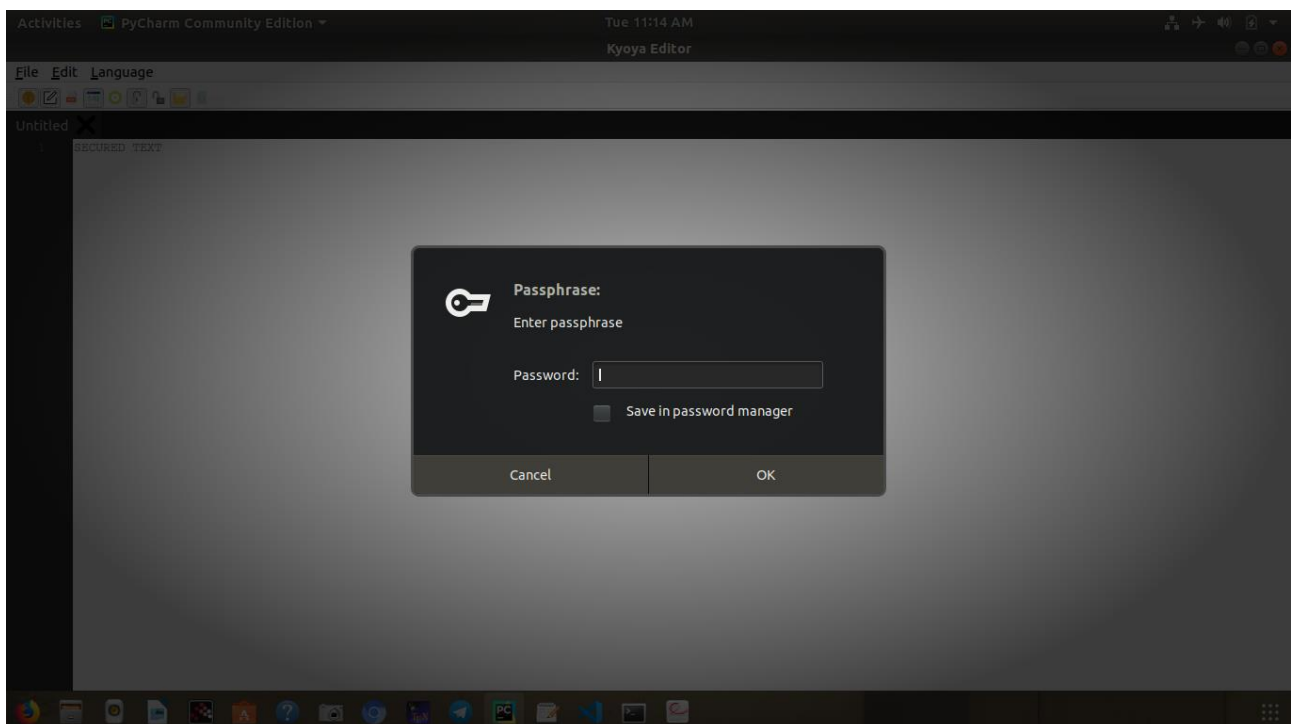


File Encryption

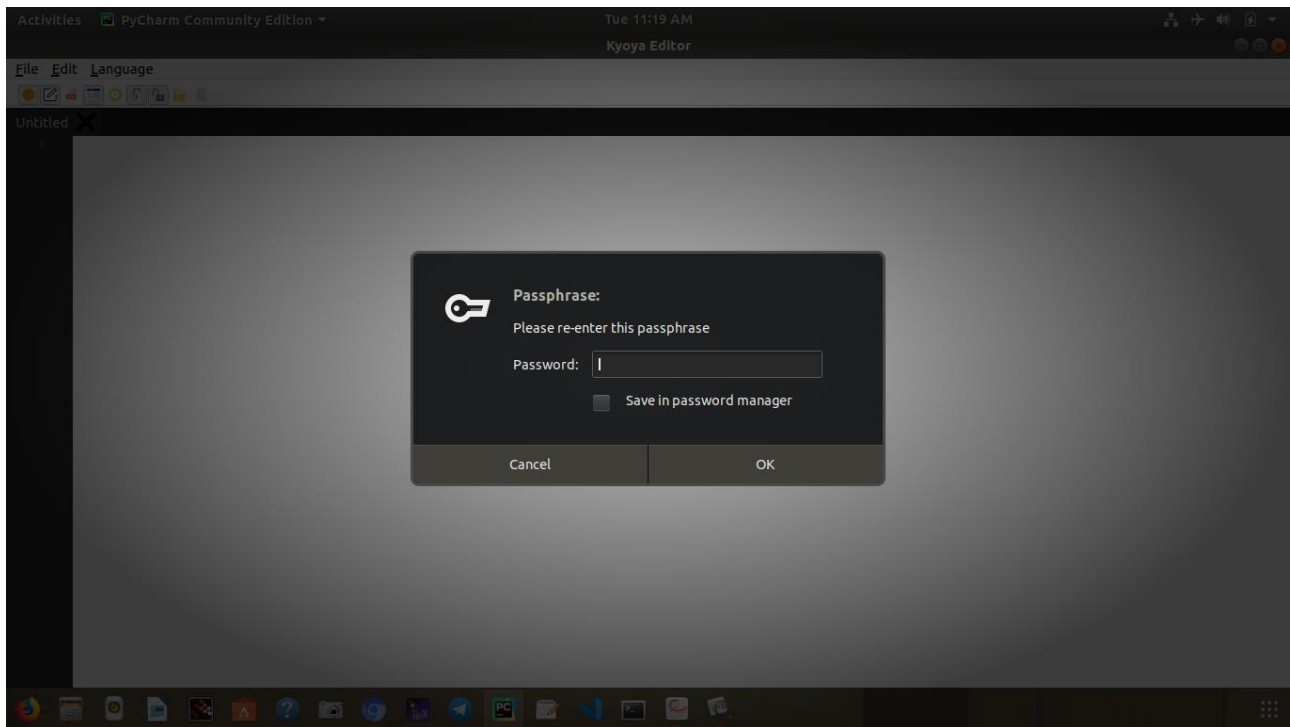
- Filename



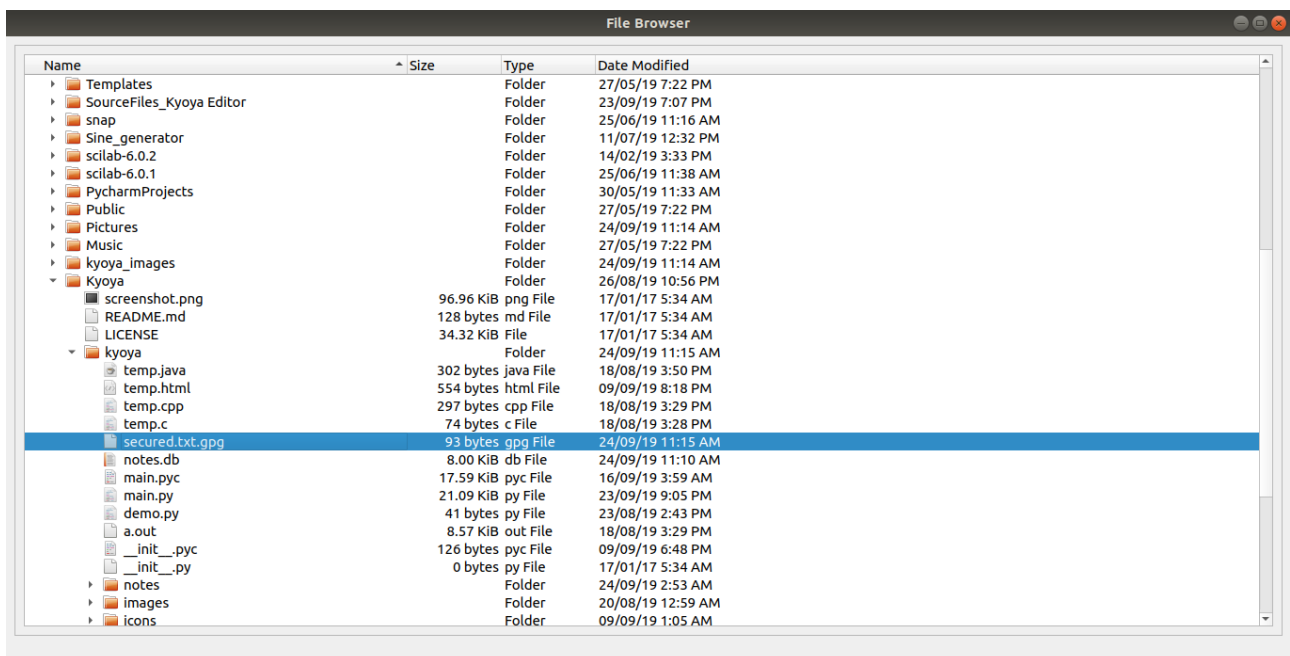
- Passkey



- Confirm Passkey

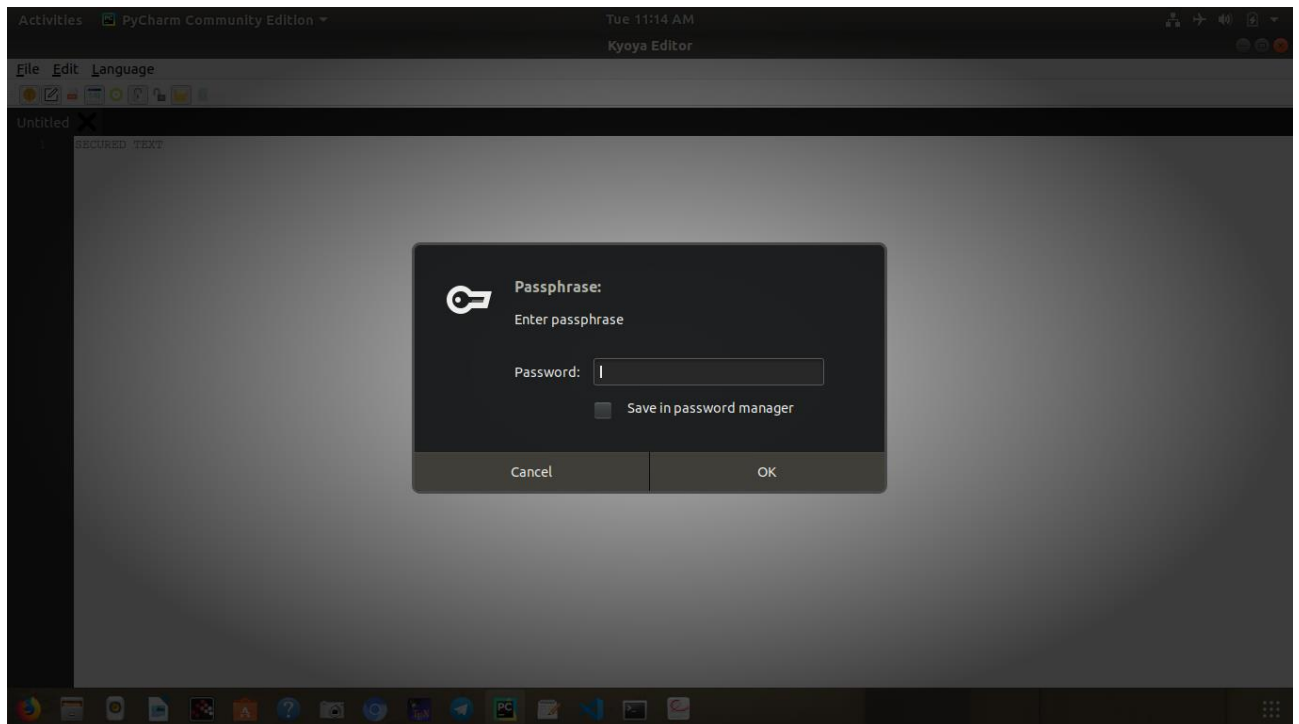
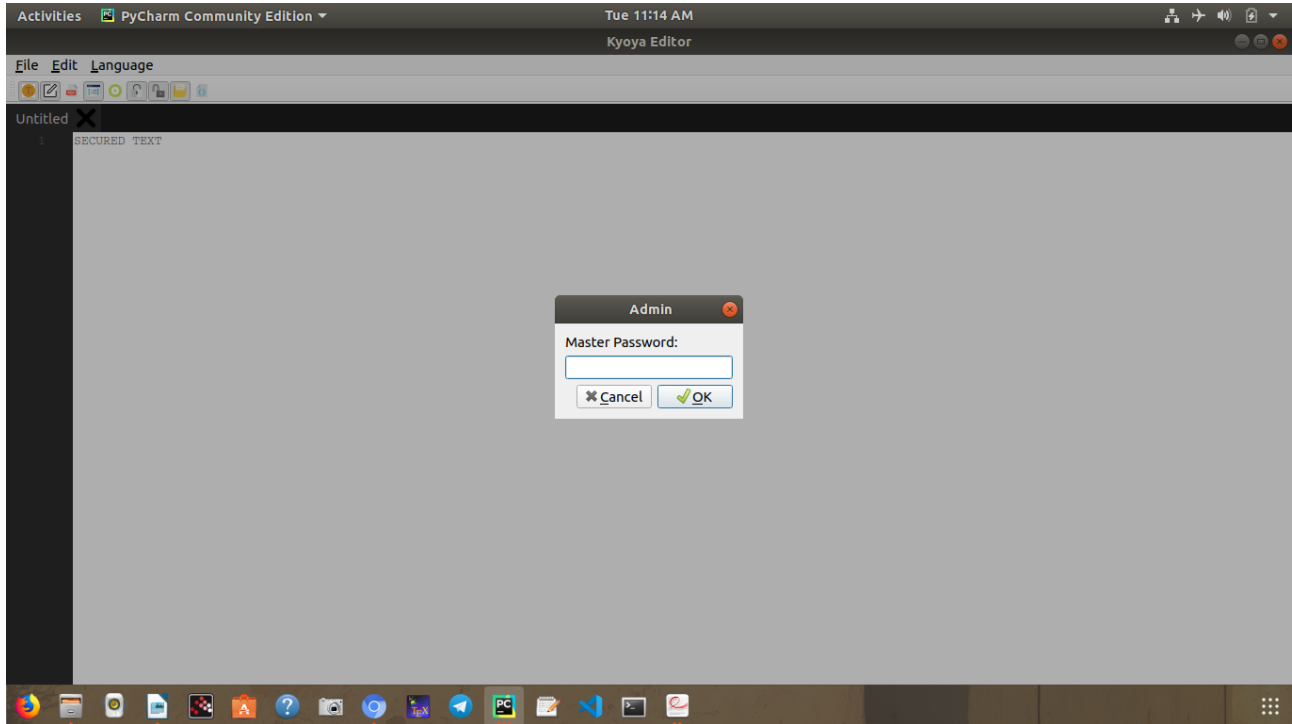


- File Secured

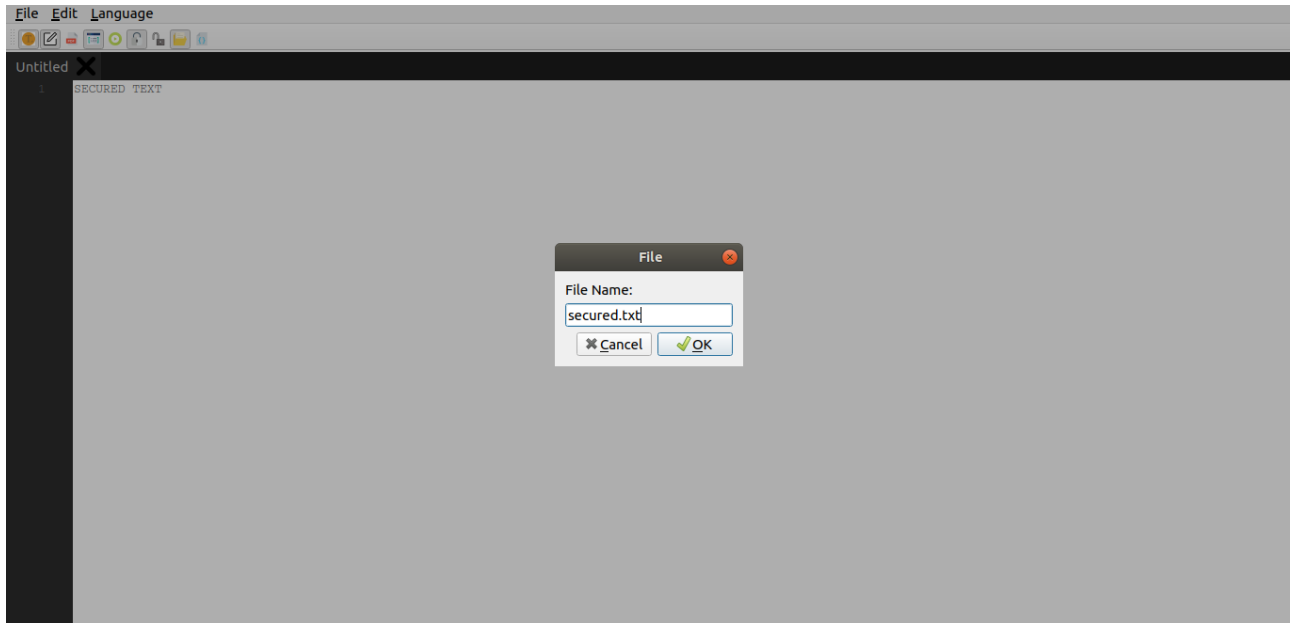


File Decryption

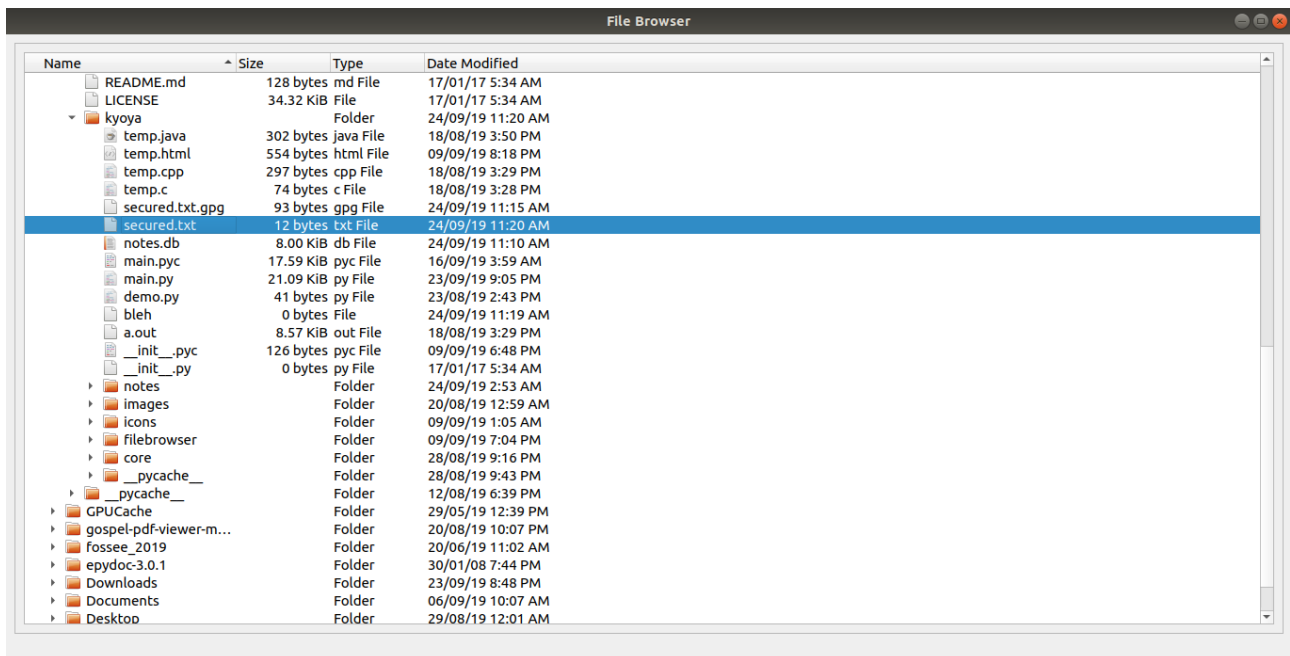
- Two Step Authentication



- Filename



- Decrypted Files



Chapter 5

Conclusion and Future Work

Conclusion

Code Editor provides Free, Opensource, lightweight, intuitive learning environment which focuses on simple yet feature-rich user interface, specially designed for students which are new to programming.

Features such as text-editor, pdf-viewer and sticky notes provide an interactive approach towards reading and notes-making.

Students are can learn entry level project management with source control.

File Browser and terminal gives an integrated approach towards Linux, which can be used for learning as well as system and project maintenance.

File security implementation enables to use Code Editor in intuitions where multiple individuals work on the same system.

➤ **Future Work**

- Dark Mode.
- Debuggers for various languages.
- More Languages support.
- Interconvertibility between languages.
- Keyword Mapping to specific functions in languages.
- Shortcuts to add basic functions such as loops, new classes, methods.

Chapter 6

References

❖ PyQt5

- <https://www.riverbankcomputing.com/static/Docs/PyQt5/introduction.html#pyqt5-components>

❖ Learning Python, 5th Edition: Mark Lutz

❖ Python GUI Programming Cookbook

❖ Zetcode

- <http://zetcode.com/gui/pyqt5/>

❖ PySide2

- <https://doc.qt.io/qtforpython/PySide2/QtGui/index.html>

❖ QsciScintilla

- <https://qscintilla.com/>

❖ SQLite

- <https://www.sqlite.org/index.html>

❖ SQLAlchemy

- <https://www.sqlalchemy.org/>

❖ GitHub

- <https://github.com/kyoyag2212/KyoyaEditor>

❖ Stack Overflow

- <https://stackoverflow.com/pyqt5>