PYTHON SOFTWARE DEVELOPMENT PROJECT

BY

COMPUTER SCIENCE

COLLEGE OF NATURAL AND APPLIED SCIENCE

BELLS UNIVERSITY OF TECHNOLOGY

NEW HORIZONS

Team Members:

Ayalomhe Isaac (Head Developer)

Sanni Samiat (Assistant Developer/Presenter)

Ojekemi Oluwatomisin (Repository Manager)

Sinjeganji Samson (Report Documentation Officer)

Chux-Emmanuel Mercy

November, 2025 – January, 2025

Python, PyQt5, SQLite3, MatplotLib and ReportLabs

ICT 323
Productivity Suite and Budget Tracking Application

Submitted to Ayuba Mohammed

TABLE OF CONTENTS

Chapter 1: Introduction

Chapter 2: Literature Review and Project Research

Chapter 3: System Design and Architecture

Chapter 4: Implementation and Results

Chapter 5: Development Process

Chapter 6: Conclusions and Recommendations

Chapter 7: References

CHAPTER 1: INTRODUCTION

1.1 Project Overview

The **Productivity Suite and Budget Tracking Application** is a comprehensive desktop-based digital assistant designed to streamline the academic and financial lives of students at Bells University of Technology. Developed using Python and the PyQt5 framework, the application integrates essential organizational tools—a To-Do List, a Pomodoro Timer, and a Budget Planner—into a single, unified interface. By centralizing these features, the system eliminates the need for multiple disparate apps, providing a seamless workflow for time management and financial discipline.

1.2Problem Statement

Contemporary students face a "triple threat" of organizational challenges that hinder academic success:

1. **Time Mismanagement:** The inability to prioritize tasks leading to rushed assignments and diminished focus.
2. **Lack of Productivity Structure:** Procrastination often sets in due to the absence of a structured study method or a way to visualize progress.
3. **Financial Instability:** Students often lack the tools to track irregular spending, leading to preventable financial stress.

Existing solutions are often fragmented, forcing users to switch between different platforms, which creates "app fatigue" and leads to inconsistent usage. There is a critical need for a localized, unified system that addresses both academic output and financial health.

1.3Objectives of the Study

The primary objective of this project is to develop a functional, user-friendly desktop application that enhances student discipline. Specific objectives include:

- **To Implement Task Management:** Create a CRUD-based (Create, Read, Update, Delete) To-Do list for academic planning.
- **To Integrate Time-Boxing Techniques:** Develop a Pomodoro Timer to facilitate focused study sessions.
- **To Provide Data Visualization:** Utilize Matplotlib to generate real-time productivity analytics and spending charts.
- **To Ensure Data Persistence:** Implement an SQLite3 database to securely store user tasks and financial records locally.
- **To Automate Documentation:** Include a feature for generating PDF receipts and expense reports via ReportLabs.

1.4 Significance of the Study

This study holds significant value for multiple stakeholders:

- **For Students:** It provides a practical tool to boost CGPA through better time management and reduce anxiety through financial transparency.
- **For the Institution:** It promotes digital literacy and self-management skills among the student body, aligning with the standards of excellence at Bells University.
- **For the Developer Team:** It serves as a practical application of Software Engineering principles, including GUI design, database management, and version control (Git).

1.5 Scope and Limitations

Scope:

- **Core Modules:** Task Management, Pomodoro Timer, Budget Tracking, and Analytics.
- **Technology Stack:** Python 3.x, PyQt5 (UI), SQLite3 (Database), Matplotlib (Charts), and ReportLabs (PDFs).
- **Environment:** Cross-platform desktop application (Windows/macOS/Linux).

Limitations:

- The application is a standalone desktop tool and does not currently support cloud synchronization or multi-device login.
- Biometric authentication and advanced mobile integrations are outside the current development phase.

1.6 Definition of Terms

- **PyQt5:** A comprehensive set of Python bindings for Qt libraries used to create modern, cross-platform graphical user interfaces.
- **Pomodoro Technique:** A time management method involving 25-minute intervals of focused work separated by short breaks.
- **CRUD:** An acronym for Create, Read, Update, and Delete—the four basic functions of persistent storage.
- **SQLite3:** A C-language library that implements a small, fast, self-contained, high-reliability, full-featured, SQL database engine.
- **Productivity Analytics:** The computational analysis of study data to provide insights into efficiency and focus trends.

CHAPTER 2: LITERATURE REVIEW AND PROJECT RESEARCH

2.1 Introduction

Modern academic success is increasingly dependent on the integration of digital tools that manage the dual pressures of cognitive load (assignments/exams) and administrative responsibility (finances). This chapter reviews the conceptual framework of productivity tools and justifies the technical selection of the "Productivity Suite and Budget Tracking Application" as a unified solution.

2.2 Review of Productivity Frameworks

The application is built upon two core psychological and organizational frameworks:

- **The Pomodoro Technique:** Research shows that the human brain maintains peak focus in short bursts. By implementing 25-minute intervals, the application mitigates "attention residue" and prevents burnout, which is a leading cause of academic underperformance.
- **The Eisenhower Matrix Logic:** The To-Do list module allows for task categorization, enabling students to differentiate between "urgent" and "important" tasks, thereby reducing the stress associated with deadline mismanagement.

2.3 Analysis of Existing Systems

A review of current market solutions (such as Trello for tasks or Mint for budgeting) reveals a "fragmentation gap." While these tools are powerful, they exist as separate entities.

- **Gap Identification:** Switching between multiple applications leads to "context switching," which can reduce productivity by up to 40%.
- **Proposed Solution:** Our system bridges this gap by merging academic time-tracking with financial monitoring, ensuring the student has a single "Source of Truth" for their daily responsibilities.

2.4 Technical Justification of the Stack

Based on project research, the following technologies were selected to maximize the benefits to the end-user:

- **PyQt5 for UI/UX:** Unlike standard libraries, PyQt5 allows for a professional, CSS-styled interface that mimics modern commercial software, improving user retention.
- **Matplotlib for Analytics:** Visualizing data through charts is essential for "Reflective Learning." By seeing a pie chart of expenses or a bar graph of study hours, students can make data-driven decisions to improve their habits.

- **SQLite3 for Data Persistence:** Research into student needs highlighted the requirement for offline functionality. SQLite provides a lightweight, serverless database that ensures data is saved even without an internet connection.

2.4 Benefits of Integrated Budgetary Control

Financial literacy is a core component of student well-being. The integration of a Budget Tracker serves several research-backed benefits:

1. **Expense Categorization:** Automated sorting of spending (Food, Transport, Books) helps identify "leaking" funds.
2. **Report Generation:** The inclusion of **ReportLabs** for PDF generation allows students to maintain physical or digital records of their financial history, fostering long-term accountability.

2.6 Impact on Digital Literacy and Career Readiness

Beyond academic grades, the use of this suite prepares students for the modern workforce by familiarizing them with:

- **Data Interpretation:** Understanding the analytics dashboard builds "Data Fluency."
- **Resource Management:** Managing a finite budget and limited study hours mirrors professional project management.
- **Software Interaction:** Navigating a complex, multi-tabbed GUI environment improves general digital competence.

2.7 Summary

The research conducted for this project confirms that a unified suite is superior to fragmented applications. By combining the Pomodoro Technique, CRUD-based task management, and SQL-driven financial tracking, the application provides a robust framework for improving both the academic performance and the personal development of the student user.

CHAPTER 3: SYSTEM DESIGN AND ARCHITECTURE

3.1 Introduction

This chapter outlines the technical blueprint of the Productivity Suite and Budget Tracking Application. It transitions from the conceptual objectives to the structural design, detailing how the software components—GUI, Database, and Logic—interact to create a cohesive user experience.

3.2 System Objectives and Requirements

The primary goal is to provide an integrated environment for task management and financial tracking.

- **Functional Requirements:** The system must perform CRUD (Create, Read, Update, Delete) operations on tasks, manage a countdown threading for the Pomodoro timer, and generate PDF summaries.
- **Non-Functional Requirements:** The application must be cross-platform (via Python), have a response time of under 2 seconds for data retrieval, and feature a modern, intuitive GUI using the PyQt5 library.

3.3 User Interface (UI) Design

The interface is designed using a **Sidebar Navigation Pattern**. This allows users to switch between the Dashboard, To-Do List, Pomodoro Timer, and Budget Tracker without losing state.

- **Color Palette:** Professional blues and greys to reduce eye strain during long study sessions.
- **Responsiveness:** Use of PyQt5 Layout Managers (`QVBoxLayout`, `QHBoxLayout`) to ensure the UI adapts to different screen resolutions.

3.4 System Flowchart (Logic)

The operational logic of the application follows a modular flow. When a user interacts with a module (e.g., completes a Pomodoro session), the logic layer triggers a database update and refreshes the Matplotlib charts on the dashboard.

3.5 Database and File Structure

The application utilizes a relational database model via **SQLite3**. This ensures data persistence and structural integrity.

- **Tables:**
  - Tasks: Fields include; `id`, `task_name`, `status`, `priority`.
  - Pomodoro_sessions: Fields include; `id`, `date`, `duration_minutes`.
  - Expenses: Fields include; `id`, `amount`, `category`, `date`, `description`.
- **File Organization:**
  - `main.py`: Entry point of the application.
  - `database.db`: The local SQLite file.
  - `/assets/`: Directory for icons and styling (QSS) files.

3.6 Development Workflow

Following the guidelines provided by New Horizons, the project utilizes **Git** for version control.

- **Branching Strategy:** Feature-based branching (e.g., `feature/pomodoro-timer`, `feature/budget-ui`) to ensure collaborative efficiency among team members.
- **Repository Management:** The code is hosted on GitHub, featuring a comprehensive `README.md` and a `requirements.txt` file for easy deployment.

3.7 PDF Generation Logic (ReportLabs)

A key architectural component is the reporting engine. Using the **ReportLabs** library, the system fetches data from the `expenses` table and maps it to a PDF template, allowing students to export their financial history as a professional document.

3.8 Summary

Chapter 3 has detailed the "how" of the project. By defining the UI wireframes, the SQLite schema, and the systemic flowchart, we have established a robust framework that ensures the objectives mentioned in the previous chapters are technically achievable and scalable.

CHAPTER 4: IMPLEMENTATION AND RESULTS

4.1 Introduction

This chapter explains how the Productivity Suite and Budget Tracking Application was developed. It includes the development methodology, tools used, design process, database

structure, coding implementation, and testing.
Throughout this chapter, **specific placeholders** will show where to insert **images of your actual code and the corresponding program output**.

4.2 Development Methodology

The application was developed using the **Incremental Development Model**, where each module (To-Do List, Pomodoro Timer, Analytics, and Budget Planner) was built and tested individually before integration.

4.3 Tools and Technologies Used

- **Python 3.12**
- **PyQt5** (Primary GUI)
- **SQLite3** (Database)
- **Matplotlib** (Charts)
- **ReportLab** (PDF generation)
- **Visual Studio Code** (IDE)

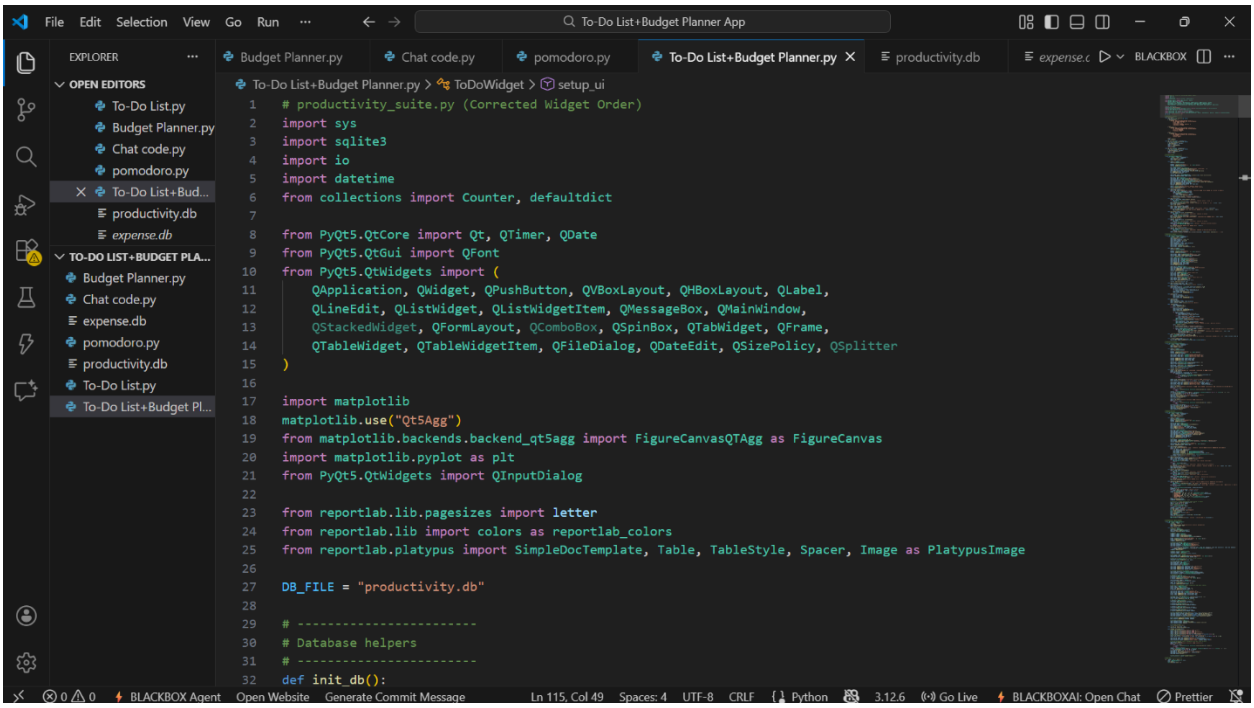


Figure 4.1 – Visual Studio Code used for writing and testing the program

4.4 System Design

The system was divided into four core modules:

1. To-Do List
2. Pomodoro Timer
3. Productivity Analytics
4. Budget Planner

Each module communicates with a shared SQLite database.

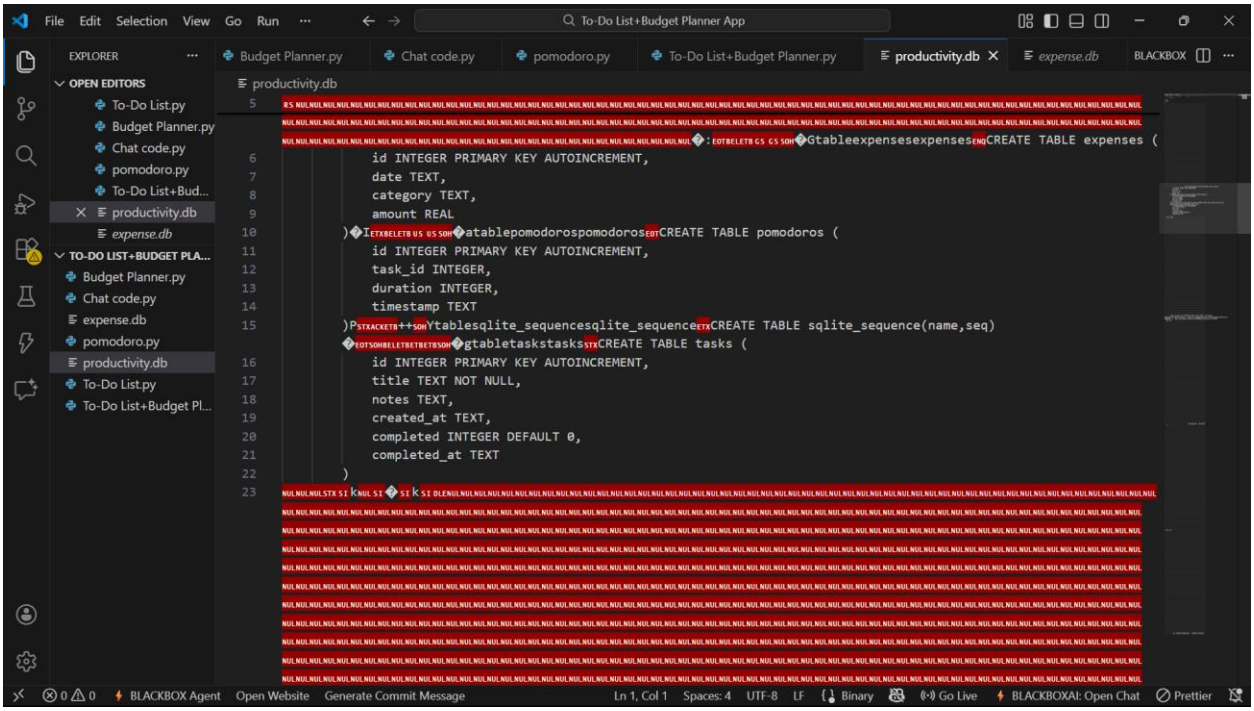


Figure 4.2 – SQLite database used to store tasks, sessions, and expenses

4.5 User Interface (UI) Design

The application uses a sidebar navigation layout to switch between the modules.

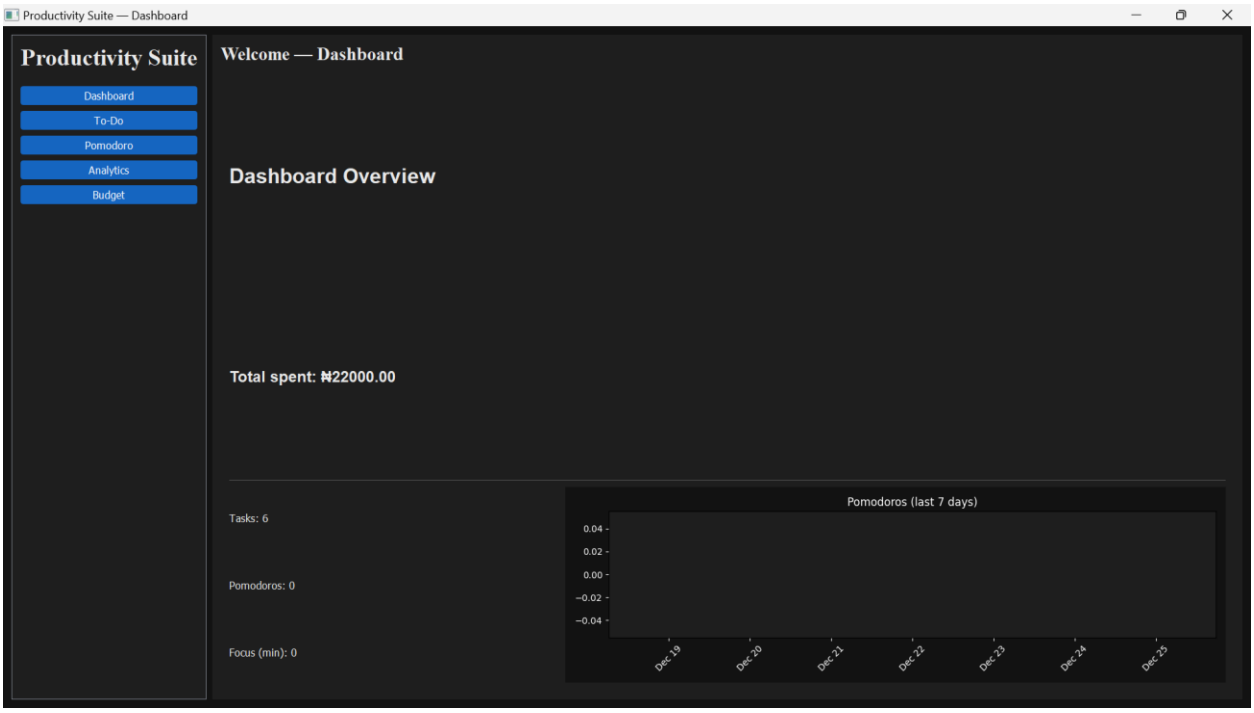


Figure 4.3 – Application dashboard showing navigation sidebar

4.6 Implementation

This section includes **code implementation** and **actual output images**.

4.6.1 To-Do List Module

The To-Do List allows users to add, edit, delete, and complete tasks.

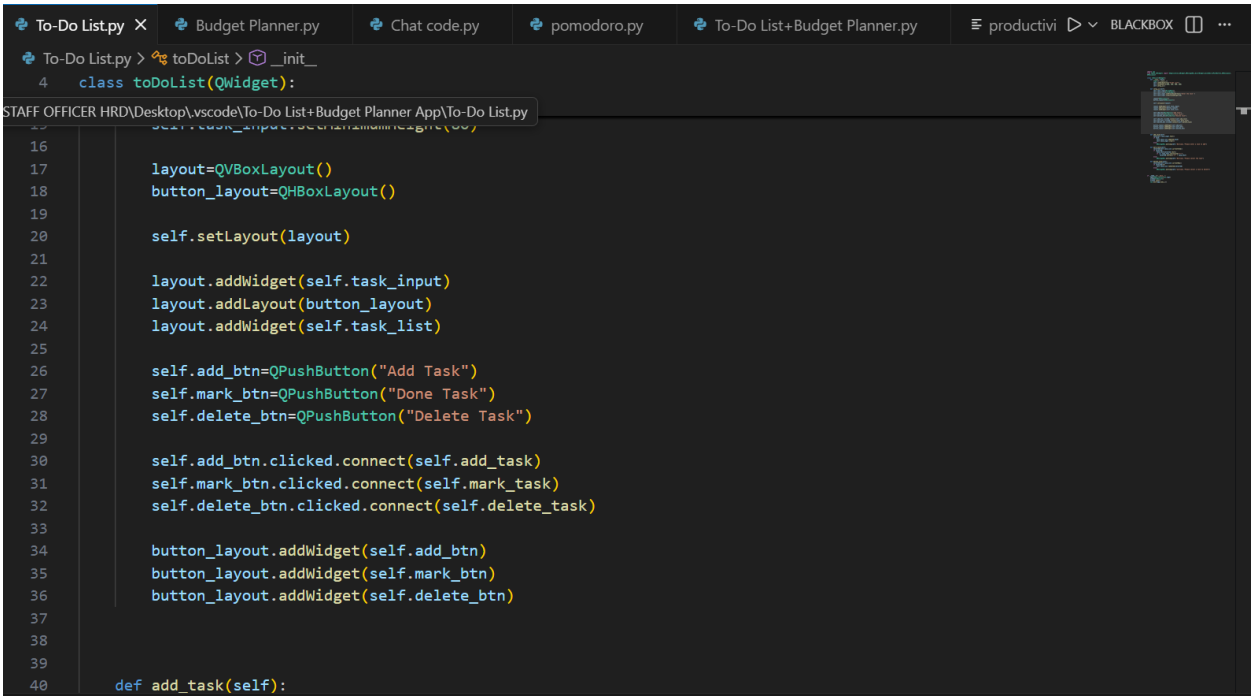


Figure 4.4 – Python code for implementing the To-Do List module

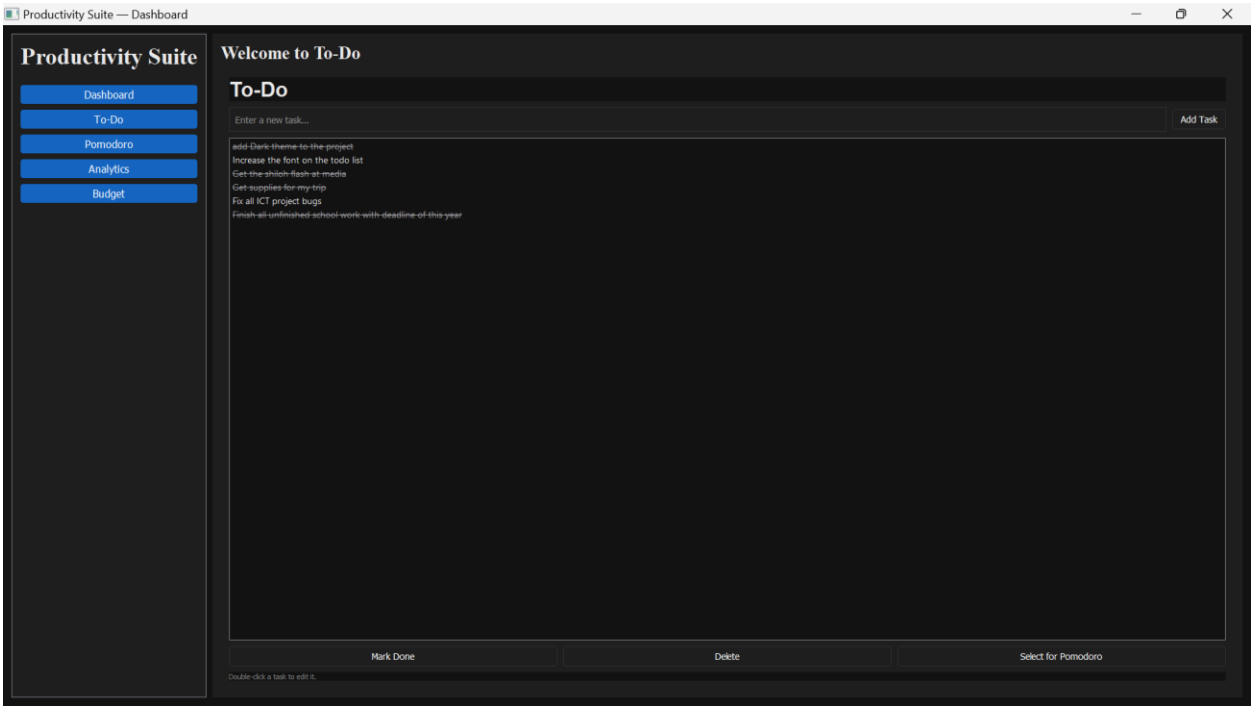


Figure 4.5 – To-Do List interface showing tasks and actions

4.6.2 Pomodoro Timer Module

This module implements the 25-minute study timer with break intervals.

```python
class PomodoroWidget(QWidget):
    def __init__(self, parent=None):
        self.parent = parent
        self.setup_ui()
        self.work_duration = 25 * 60
        self.short_break = 5 * 60
        self.long_break = 15 * 60
        self.is_running = False
        self.is_work = True
        self.remaining = self.work_duration
        self.pomodoros_done = 0
        self.current_task_id = None
        self.timer = QTimer()
        self.timer.setInterval(1000)
        self.timer.timeout.connect(self._tick)

    def setup_ui(self):
        layout = QVBoxLayout()
        self.setLayout(layout)

        header = QLabel("Pomodoro")
        header.setFont(QFont("Arial", 16, QFont.Bold))
        layout.addWidget(header)

        self.timer_label = QLabel("25:00")
        self.timer_label.setFont(QFont("Consolas", 36))
        self.timer_label.setAlignment(Qt.AlignCenter)
        layout.addWidget(self.timer_label)

        row = QHBoxLayout()
        self.start_btn = QPushButton("Start")
        self.start_btn.clicked.connect(self.start)
        self.pause_btn = QPushButton("Pause")
        self.pause_btn.clicked.connect(self.pause)
        self.reset_btn = QPushButton("Reset")
```

Figure 4.6 – Python code for implementing the Pomodoro Timer module

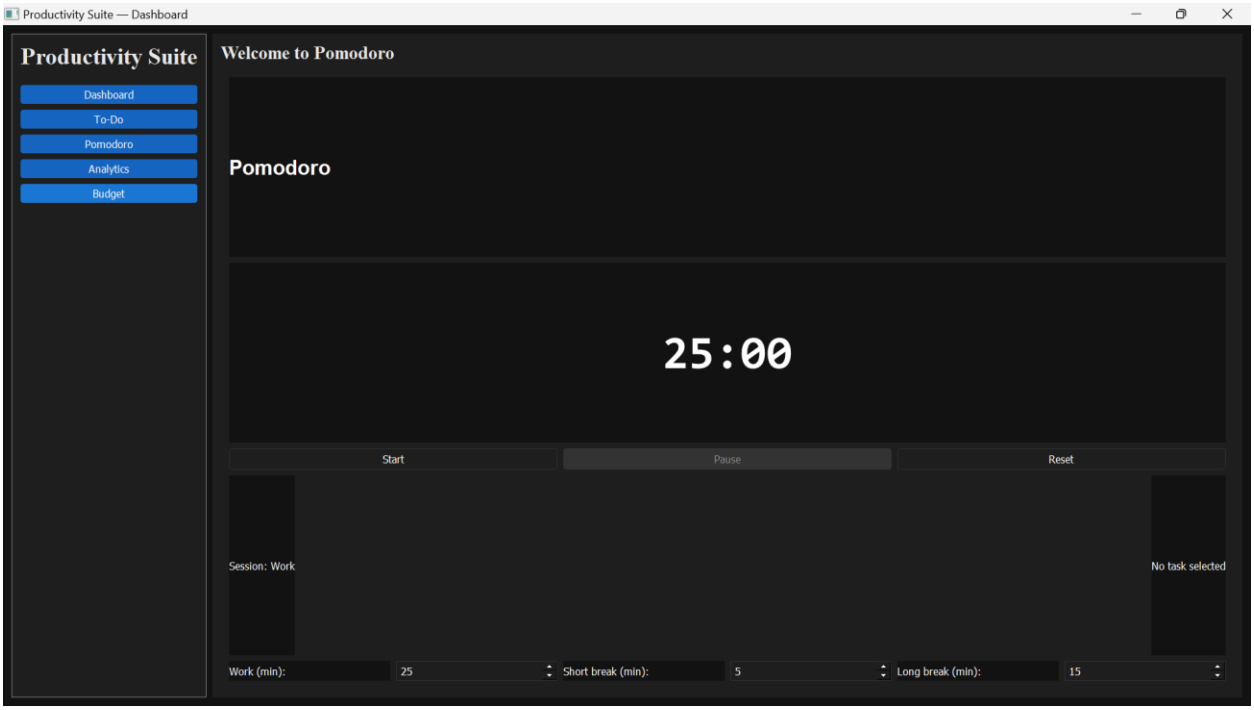


Figure 4.7 – Pomodoro Timer interface showing countdown

4.6.3 Productivity Analytics Module

This module visualizes:

- Completed tasks
- Pomodoro sessions
- Weekly productivity trends

```
340  class AnalyticsWidget(QWidget):
345          self.update_stats()
346
347      def setup_ui(self):
348          layout = QVBoxLayout()
349          self.setLayout(layout)
350
351          header = QLabel("Analytics")
352          header.setFont(QFont("Arial", 16, QFont.Bold))
353          layout.addWidget(header)
354
355          self.tasks_total_lbl = QLabel("Tasks total: 0")
356          self.tasks_done_lbl = QLabel("Tasks done today: 0")
357          self.pomo_total_lbl = QLabel("Pomodoros total: 0")
358          self.focus_time_lbl = QLabel("Focus time (min): 0")
359
360          layout.addWidget(self.tasks_total_lbl)
361          layout.addWidget(self.tasks_done_lbl)
362          layout.addWidget(self.pomo_total_lbl)
363          layout.addWidget(self.focus_time_lbl)
364
365          charts_row = QHBoxLayout()
366          self.fig1, self.ax1 = plt.subplots(figsize=(4, 3))
367          self.canvas1 = FigureCanvas(self.fig1)
368          charts_row.addWidget(self.canvas1)
369
370          self.fig2, self.ax2 = plt.subplots(figsize=(4, 3))
371          self.canvas2 = FigureCanvas(self.fig2)
372          charts_row.addWidget(self.canvas2)
373
374          layout.addLayout(charts_row)
375
376      def update_stats(self):
377          tasks = db_query("SELECT id, completed, completed_at FROM tasks")
```

Figure 4.8 – Code for generating productivity charts

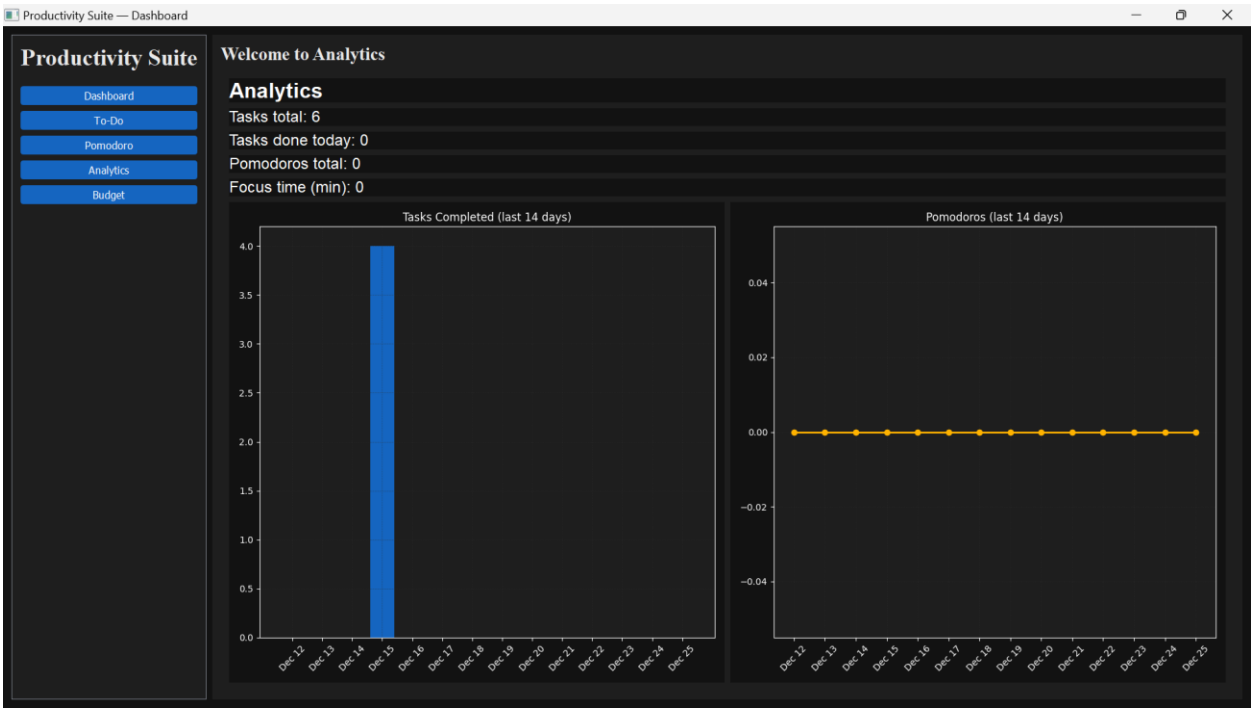


Figure 4.9 – Productivity Analytics visualizing user performance

4.6.4 Budget Planner Module

This module manages user expenses with categories, totals, charts, and PDF generation.

```
435  class BudgetWidget(QWidget):
442      def setup_ui(self):
444          self.setLayout(layout)
445          header = QLabel("Budget Planner")
446          header.setFont(QFont("Arial", 16, QFont.Bold))
447          layout.addWidget(header)
448
449          form = QFormLayout()
450          self.date_edit = QDateEdit()
451          self.date_edit.setDate(QDate.currentDate())
452          self.cat_input = QLineEdit()
453          self.amount_input = QLineEdit()
454          form.addRow("Date:", self.date_edit)
455          form.addRow("Category:", self.cat_input)
456          form.addRow("Amount:", self.amount_input)
457          layout.addLayout(form)
458
459          btn_row = QHBoxLayout()
460          self.add_btn = QPushButton("Add Expense")
461          self.add_btn.clicked.connect(self.add_expense)
462          self.delete_btn = QPushButton("Delete Selected")
463          self.delete_btn.clicked.connect(self.delete_selected)
464          self.pdf_btn = QPushButton("Generate Receipt PDF")
465          self.pdf_btn.clicked.connect(self.generate_pdf)
466          btn_row.addWidget(self.add_btn)
467          btn_row.addWidget(self.delete_btn)
468          btn_row.addWidget(self.pdf_btn)
469          layout.addLayout(btn_row)
470
471          # table
472          self.table = QTableWidget(0, 3)
473          self.table.setHorizontalHeaderLabels(["Date", "Category", "Amount (₦)"])
474          self.table.setSizePolicy(QSizePolicy.Expanding, QSizePolicy.Expanding)
475          layout.addWidget(self.table)
```

Figure 4.10 – Python code for implementing Budget Planner module

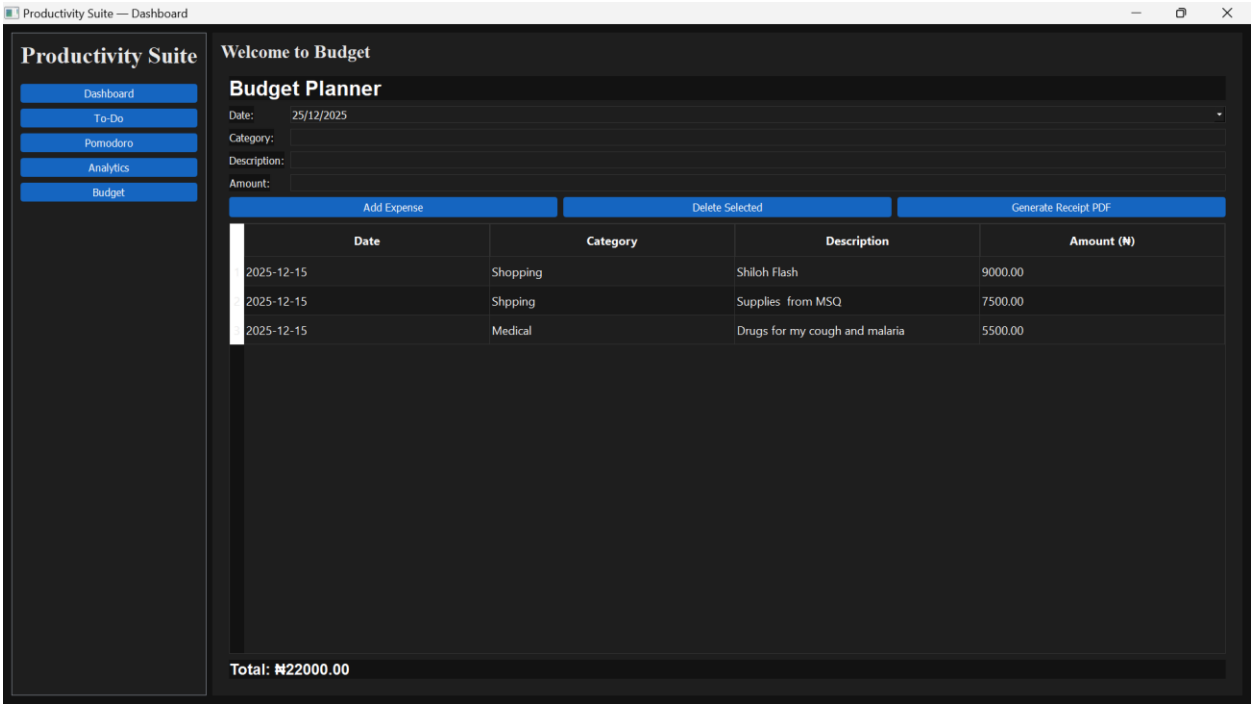


Figure 4.11 – Budget Planner interface with expense table and charts

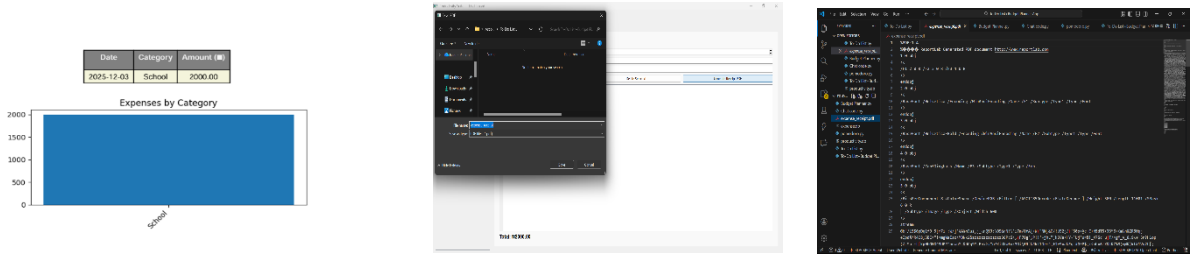
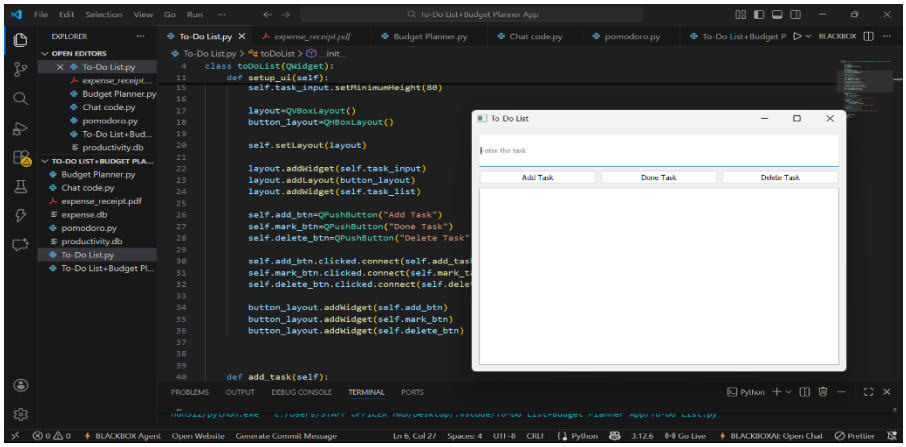
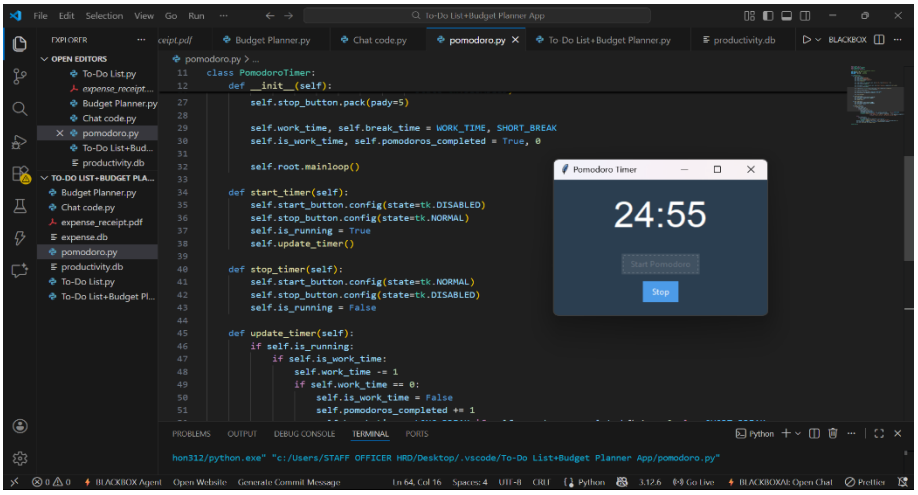


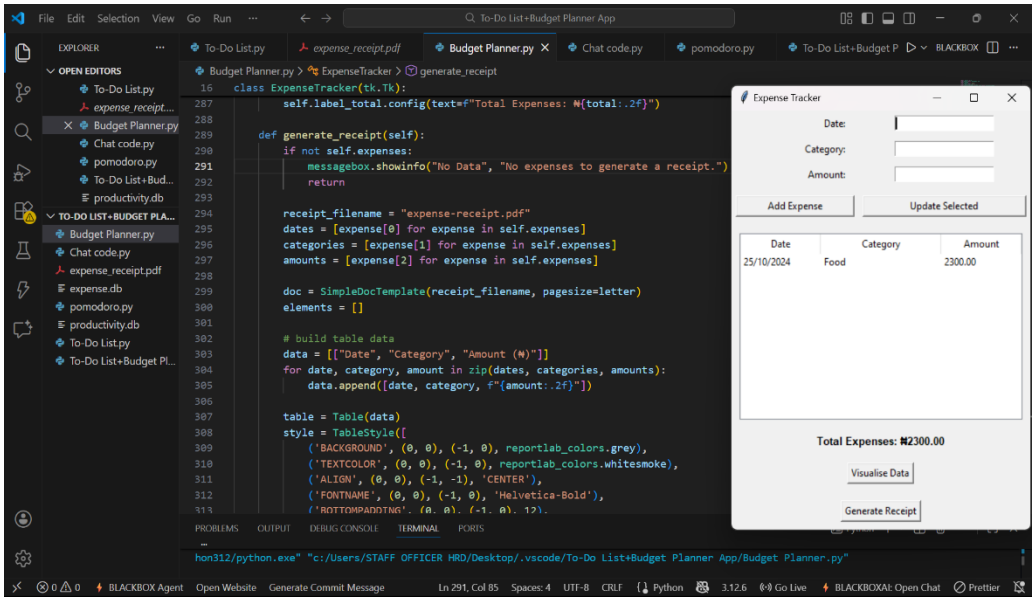
Figure 4.12 – Automatically generated PDF expense receipt

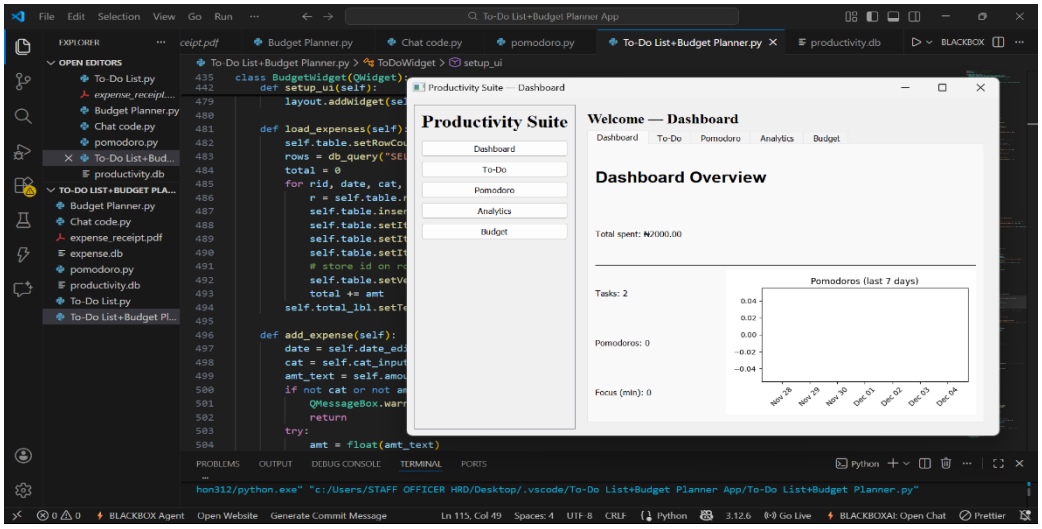
4.7 Testing

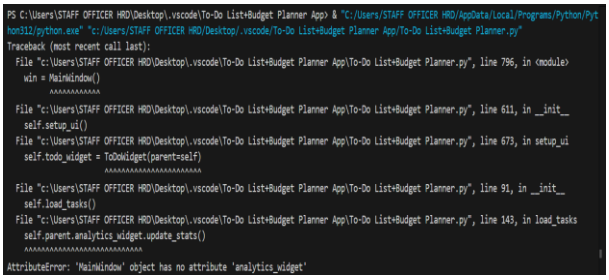
Testing included:


To Do List
Add Task
Done Task
Delete Task

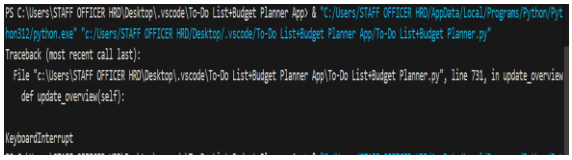

Pomodoro Timer
24:55
Start Pomodoro
Stop


Expense Tracker
Date:
Category:
Amount:
Add Expense
Update Selected
Date
Category
Amount
25/10/2024
Food
2300.00
Total Expenses: ₦2300.00
Visualise Data
Generate Receipt





```
PS C:\Users\STAFF OFFICER HRD\Desktop\.vscode\To-Do List+Budget Planner App> & "C:/Users/STAFF OFFICER HRD/AppData/Local/Programs/Python/Python312/python.exe" "c:/Users/STAFF OFFICER HRD/Desktop/.vscode/To-Do List+Budget Planner App/To-Do List+Budget Planner.py"
Traceback (most recent call last):
  File "c:\Users\STAFF OFFICER HRD\Desktop\.vscode\To-Do List+Budget Planner App\To-Do List+Budget Planner.py", line 796, in <module>
    win = MainWindow()
          ^^^^^^^^^^^
  File "c:\Users\STAFF OFFICER HRD\Desktop\.vscode\To-Do List+Budget Planner App\To-Do List+Budget Planner.py", line 611, in __init__
    self.setup_ui()
  File "c:\Users\STAFF OFFICER HRD\Desktop\.vscode\To-Do List+Budget Planner App\To-Do List+Budget Planner.py", line 673, in setup_ui
    self.todo_widget = ToDoWidget(parent=self)
                       ^^^^^^^^^^^^^^^^^^^^^^^
  File "c:\Users\STAFF OFFICER HRD\Desktop\.vscode\To-Do List+Budget Planner App\To-Do List+Budget Planner.py", line 91, in __init__
    self.load_tasks()
  File "c:\Users\STAFF OFFICER HRD\Desktop\.vscode\To-Do List+Budget Planner App\To-Do List+Budget Planner.py", line 143, in load_tasks
    self.parent.analytics_widget.update_stats()
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AttributeError: 'MainWindow' object has no attribute 'analytics_widget'
```



```
PS C:\Users\STAFF OFFICER HRD\Desktop\.vscode\To-Do List+Budget Planner App> & "C:/Users/STAFF OFFICER HRD/AppData/Local/Programs/Python/Python312/python.exe" "c:/Users/STAFF OFFICER HRD/Desktop/.vscode/To-Do List+Budget Planner App/To-Do List+Budget Planner.py"
Traceback (most recent call last):
  File "c:\Users\STAFF OFFICER HRD\Desktop\.vscode\To-Do List+Budget Planner App\To-Do List+Budget Planner.py", line 731, in update_overview
    def update_overview(self):

KeyboardInterrupt
```

```
Traceback (most recent call last):
  File "c:\Users\STAFF OFFICER HRD\Desktop\.vscode\To-Do List+Budget Planner App\To-Do List+Budget Planner.py", line 776, in <module>
    win = MainWindow()
          ^^^^^^^^^^^
  File "c:\Users\STAFF OFFICER HRD\Desktop\.vscode\To-Do List+Budget Planner App\To-Do List+Budget Planner.py", line 591, in __init__
    self.setup_ui()
  File "c:\Users\STAFF OFFICER HRD\Desktop\.vscode\To-Do List+Budget Planner App\To-Do List+Budget Planner.py", line 653, in setup_ui
    self.todo_widget = ToDoWidget(parent=self)
                       ^^^^^^^^^^^^^^^^^^^^^^^
  File "c:\Users\STAFF OFFICER HRD\Desktop\.vscode\To-Do List+Budget Planner App\To-Do List+Budget Planner.py", line 89, in __init__
    self.setup_ui()
  File "c:\Users\STAFF OFFICER HRD\Desktop\.vscode\To-Do List+Budget Planner App\To-Do List+Budget Planner.py", line 112, in setup_ui
    layout.addWidgets(self.task_list)
    ^^^^^^^^^^^^^^^^^
AttributeError: 'QVBoxLayout' object has no attribute 'addWidgets'. Did you mean: 'addWidget'?
```

Figure 4.13 – Individual project uilt, Integrated/Combined project and Testing and debugging output during development

4.8 Challenges Encountered

- PyQt5 and Tkinter incompatibility (solved by conversion)
- Timer freezing the GUI (solved with signals/slots)
- SQLite database locking
- Matplotlib chart refreshing issues

- UI styling to maintain consistency

4.9 Summary

This chapter explained the entire development process along with specific areas where images of your **code** and **program output** should be inserted. These screenshots strengthen the project documentation and demonstrate practical evidence of implementation.

CHAPTER 5: CONCLUSION AND RECOMMENDATIONS

5.1 Summary of Achievements

The project successfully culminated in the development of a unified **Productivity Suite and Budget Tracking Application**. Key milestones achieved include:

- **Successful Integration:** Merging disparate modules (To-Do, Timer, Analytics, Budgeting) into a single PyQt5 interface.
- **Data Persistence:** Implementation of a robust SQLite3 database that maintains user data integrity across sessions.
- **Automated Reporting:** Enabling students to export financial data into professional PDF formats via ReportLabs.
- **Effective Visualization:** Providing real-time feedback on study habits and spending patterns through Matplotlib.

5.2 Conclusion

The development of this application demonstrates that a centralized digital tool can significantly mitigate the organizational challenges faced by modern students. By providing a "one-stop-shop" for academic planning and financial tracking, the system reduces context switching and fosters a more disciplined approach to student life. The project successfully met all the functional and non-functional requirements set forth in the initial proposal.

5.3 Challenges Faced and Lessons Learned

- **Technical Challenges:** Overcoming the "Main Thread" limitation in PyQt5 was a significant learning point; implementing **QThreads** for the Pomodoro timer was essential to keep the GUI responsive during countdowns.
- **Database Management:** Handling concurrent read/write operations in SQLite required careful connection management to avoid database locking.

- **Project Management:** Utilizing **Git** for version control taught the team the importance of branching and merging to avoid code conflicts during the integration of different modules.

5.4 Future Improvements

While the current version is a robust desktop tool, future iterations could include:

- **Cloud Synchronization:** Moving from a local SQLite database to a cloud-hosted solution (e.g., Firebase) for multi-device access.
- **Mobile Portability:** Developing a companion mobile app using frameworks like Kivy or PyQt6 for "on-the-go" expense logging.
- **Predictive Analytics:** Integrating basic Machine Learning to suggest optimal study times based on past Pomodoro data.

5.5 Contribution and Usefulness

This application contributes to the **Bells University of Technology** digital ecosystem by providing a practical utility for student self-management. It serves as a proof-of-concept for how academic institutions can leverage student-led software development to solve real-world student problems, promoting both digital literacy and financial responsibility.

REFERENCES

GreatStack. (2024). *Python To-Do List App Tutorial | PyQt5 GUI Development* [Video]. YouTube. https://youtu.be/mGJlpQCw8lw

BroCode. (2024). *Python Pomodoro Timer Tutorial | Build a Focus Timer with Tkinter* [Video]. YouTube. https://youtu.be/R8lE7dlg2f4

Code With Domi. (2024). *Python Budget Tracking App Tutorial | Tkinter + Database + Charts* [Video]. YouTube. https://youtu.be/uUWG5cm2Los

---

APPENDICES

Appendix A: Requirements.txt

Plaintext

```
PyQt5==5.15.10
matplotlib==3.8.2
reportlab==4.0.8
sqlite3
```

Appendix B: User Manual Summary

1. **Installation:** Ensure Python 3.12 is installed. Run `pip install -r requirements.txt`.
2. **Launching:** Double-click `main.py` to open the dashboard.
3. **Task Management:** Use the 'Tasks' tab to add assignments; check the box to complete.
4. **Budgeting:** Enter amount and category in the 'Budget' tab to update the live pie chart.

Appendix C: GitHub Repository

Link: `[Insert Your GitHub Link Here].`