

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.

1.2 Problem 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.

1.3 Objectives 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.

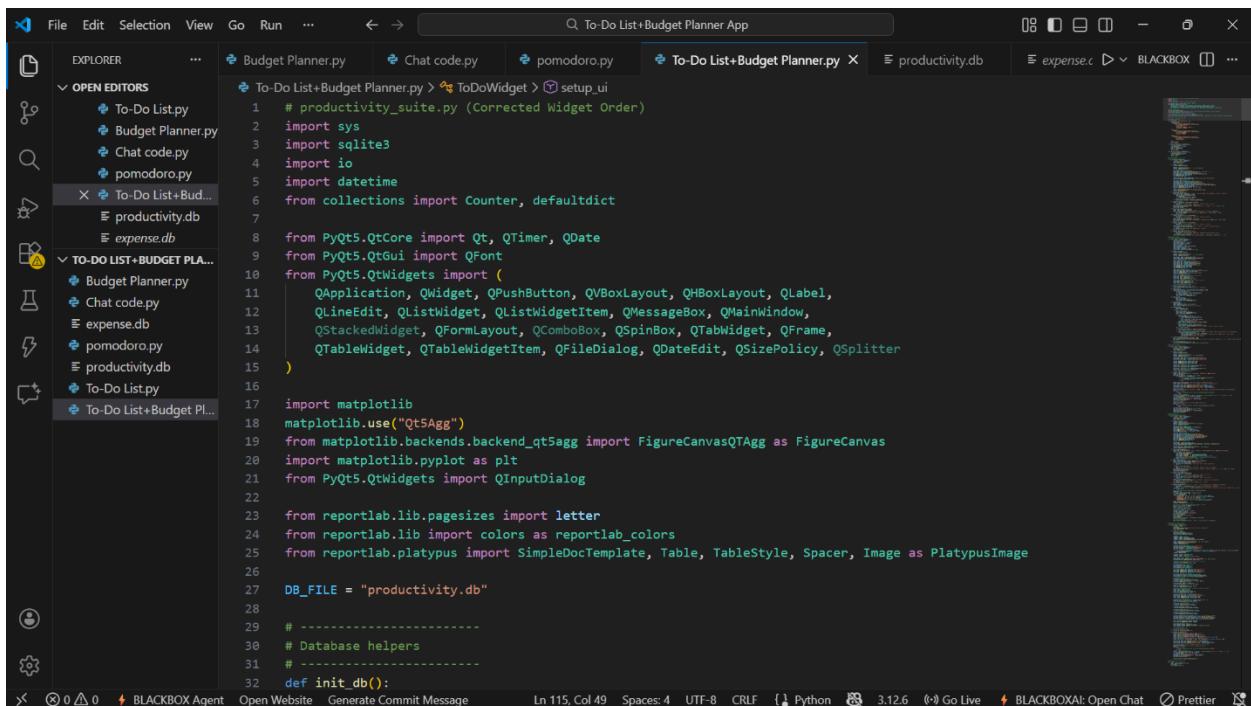
Throughout this chapter, specific placeholders will show where to insert **images of your actual code and the corresponding program output**.

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)



The screenshot shows the Visual Studio Code interface with the title bar "To-Do List+Budget Planner App". The Explorer sidebar on the left lists files and folders: "EXPLORER", "OPEN EDITORS", and "TO-DO LIST+BUDGET PLA...". The "OPEN EDITORS" section contains "Budget Planner.py", "Chat code.py", "pomodoro.py", and "To-Do List+Budget PLA...". The main code editor shows the "To-Do List+Budget Planner.py" file with the following Python code:

```
# productivity_suite.py (Corrected Widget Order)
1 # productivity_suite.py (Corrected Widget Order)
2 import sys
3 import sqlite3
4 import io
5 import datetime
6 from collections import Counter, defaultdict
7
8 from PyQt5.QtCore import Qt, QTimer, QDate
9 from PyQt5.QtGui import QFont
10 from PyQt5.QtWidgets import (
11     QApplication, QWidget, QPushButton, QVBoxLayout, QHBoxLayout, QLabel,
12     QLineEdit, QListWidget, QListWidgetItem, QMessageBox, QMainWindow,
13     QStackedWidget, QFormLayout, QComboBox, QSpinBox, QTabWidget, QFrame,
14     QTableWidget, QTableWidgetItem, QFileDialog, QDateEdit, QSizePolicy, QSplitter
15 )
16
17 import matplotlib
18 matplotlib.use("Qt5Agg")
19 from matplotlib.backends.backend_qt5agg import FigureCanvasQTAgg as FigureCanvas
20 import matplotlib.pyplot as plt
21 from PyQt5.QtWidgets import QInputDialog
22
23 from reportlab.lib.pagesizes import letter
24 from reportlab.lib import colors as reportlab_colors
25 from reportlab.platypus import SimpleDocTemplate, Table, TableStyle, Spacer, Image as PlatypusImage
26
27 DB_FILE = "productivity.db"
28
29 # -----
30 # Database helpers
31 # -----
32 def init_db():
33     pass
```

The status bar at the bottom shows "Ln 115, Col 49 Spaces: 4 UTR-8 CRLF { Python 3.12.6 Go Live BLACKBOXAI: Open Chat Prettier".

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.

```
CREATE TABLE expenses (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    date TEXT,
    category TEXT,
    amount REAL
)
CREATE TABLE pomodoros (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    task_id INTEGER,
    duration INTEGER,
    timestamp TEXT
)
CREATE TABLE sqlite_sequence(name,seq)
CREATE TABLE tasks (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    title TEXT NOT NULL,
    notes TEXT,
    created_at TEXT,
    completed INTEGER DEFAULT 0,
    completed_at TEXT
)
```

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.

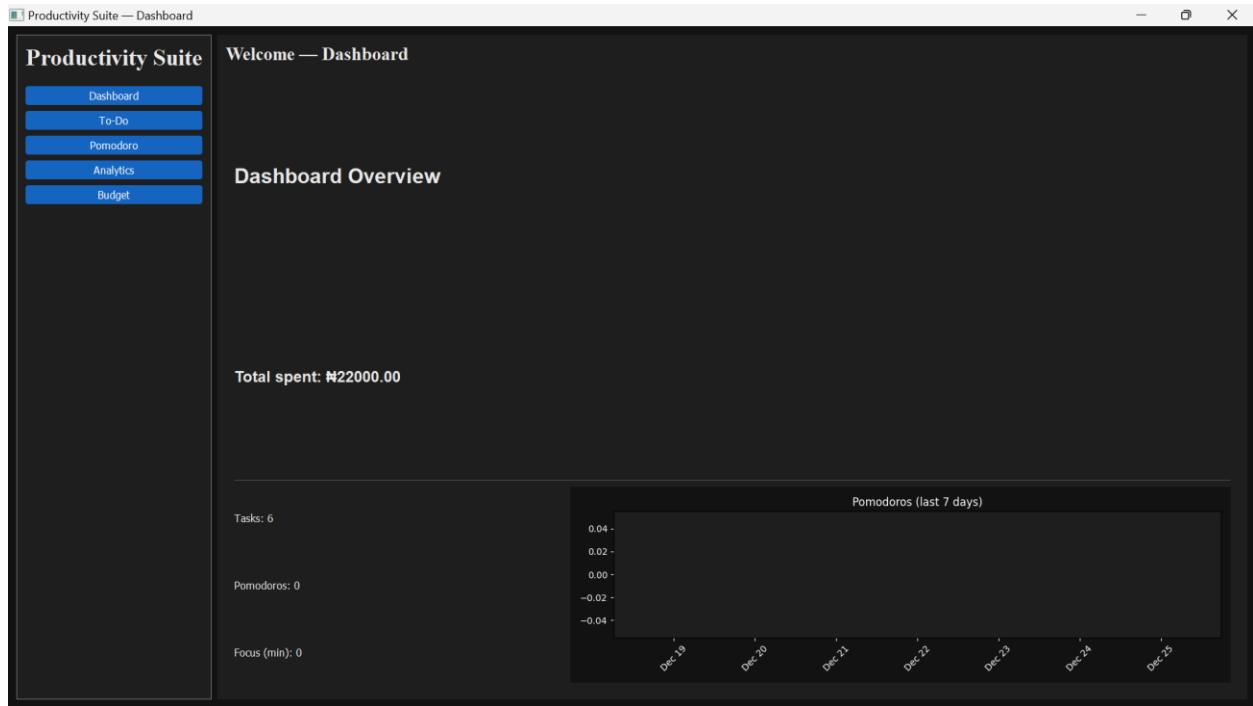


Figure 4.3 – Application dashboard showing navigation sidebar

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.

The screenshot shows a dark-themed instance of VS Code with multiple tabs open. The active tab is 'To-Do List.py'. The code implements a 'ToDoList' class using PyQt5's QWidgets and layouts. It includes methods for adding, marking, and deleting tasks, and connects them to QPushButton click events.

```
4  class ToDoList(QWidget):  
5  
6     def __init__(self):  
7         super().__init__()  
8  
9         self.task_input = QLineEdit(self)  
10        self.task_list = QListWidget(self)  
11  
12        self.add_btn=QPushButton("Add Task", self)  
13        self.mark_btn=QPushButton("Done Task", self)  
14        self.delete_btn=QPushButton("Delete Task", self)  
15  
16        layout=QVBoxLayout()  
17        button_layout=QHBoxLayout()  
18  
19        self.setLayout(layout)  
20  
21        layout.addWidget(self.task_input)  
22        layout.addLayout(button_layout)  
23        layout.addWidget(self.task_list)  
24  
25        self.add_btn.clicked.connect(self.add_task)  
26        self.mark_btn.clicked.connect(self.mark_task)  
27        self.delete_btn.clicked.connect(self.delete_task)  
28  
29  
30        button_layout.addWidget(self.add_btn)  
31        button_layout.addWidget(self.mark_btn)  
32        button_layout.addWidget(self.delete_btn)  
33  
34  
35  
36  
37  
38  
39  
40    def add_task(self):
```

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

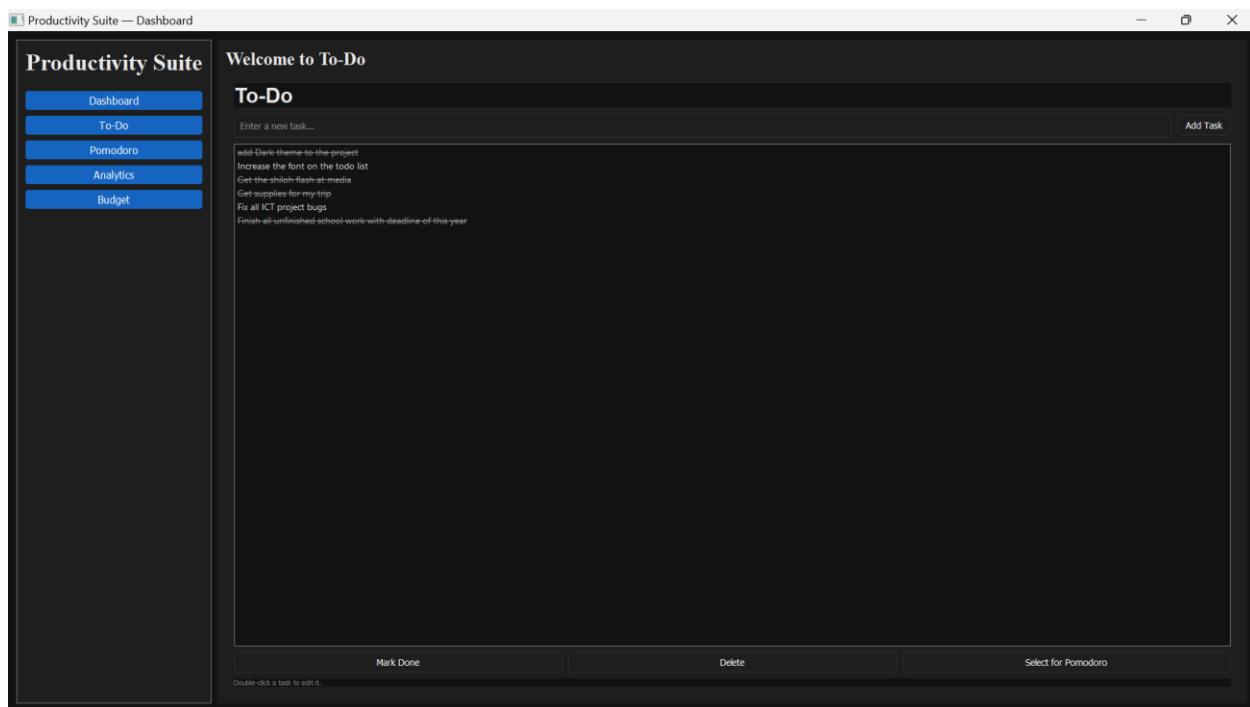


Figure 4.5 – To-Do List interface showing tasks and actions

4.6.2 Pomodoro Timer Module

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

```
194  class PomodoroWidget(QWidget):
195      def __init__(self, parent=None):
196          self.parent = parent
197          self.setup_ui()
198          self.work_duration = 25 * 60
199          self.short_break = 5 * 60
200          self.long_break = 15 * 60
201          self.is_running = False
202          self.is_work = True
203          self.remaining = self.work_duration
204          self.pomodoros_done = 0
205          self.current_task_id = None
206          self.timer = QTimer()
207          self.timer.setInterval(1000)
208          self.timer.timeout.connect(self._tick)
209
210      def setup_ui(self):
211          layout = QVBoxLayout()
212          self.setLayout(layout)
213
214          header = QLabel("Pomodoro")
215          header.setFont(QFont("Arial", 16, QFont.Bold))
216          layout.addWidget(header)
217
218          self.timer_label = QLabel("25:00")
219          self.timer_label.setFont(QFont("Consolas", 36))
220          self.timer_label.setAlignment(Qt.AlignCenter)
221          layout.addWidget(self.timer_label)
222
223          row = QHBoxLayout()
224          self.start_btn = QPushButton("Start")
225          self.start_btn.clicked.connect(self.start)
226          self.pause_btn = QPushButton("Pause")
227          self.pause_btn.clicked.connect(self.pause)
228          self.reset_btn = QPushButton("Reset")
229          self.reset_btn.clicked.connect(self.reset)
```

Figure 4.6 – Python code for implementing the Pomodoro Timer module

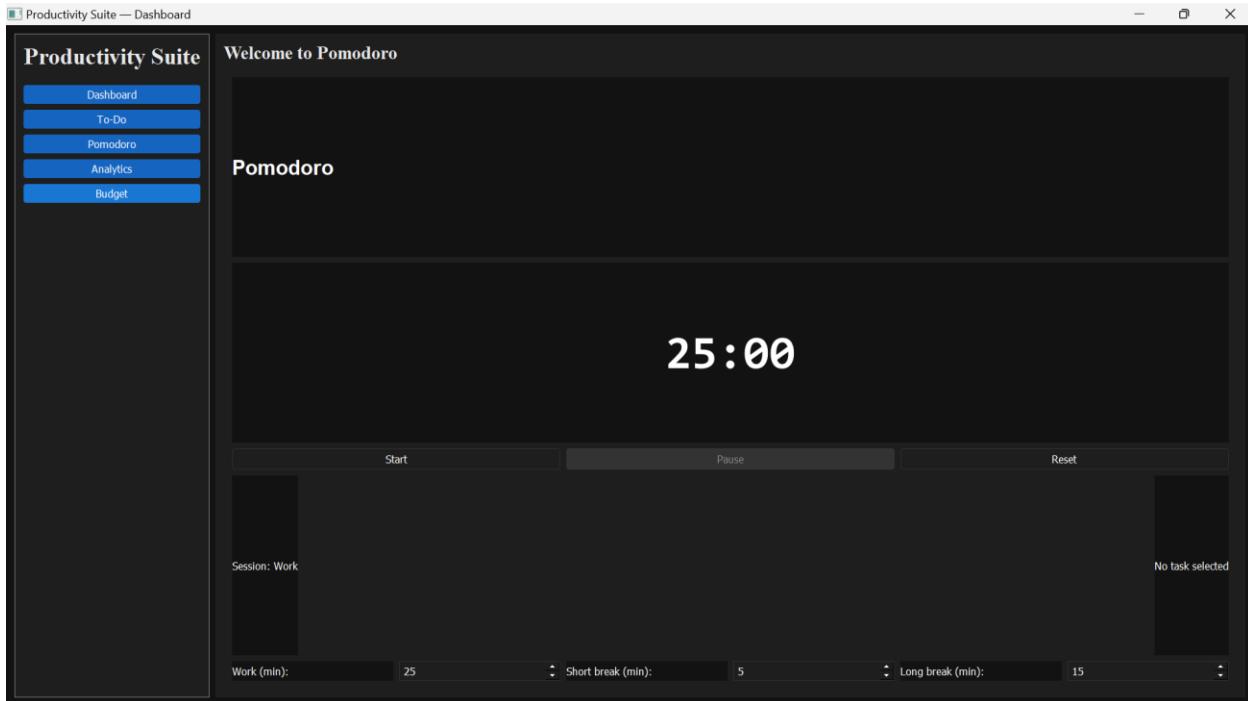


Figure 4.7 – Pomodoro Timer interface showing countdown

4.6.3 Productivity Analytics Module

This module visualizes:

- Completed tasks
- Pomodoro sessions
- Weekly productivity trends

```
340 class AnalyticsWidget(QWidget):
341     ...
342     self.update_stats()
343
344     def setup_ui(self):
345         layout = QVBoxLayout()
346         self.setLayout(layout)
347
348         header = QLabel("Analytics")
349         header.setFont(QFont("Arial", 16, QFont.Bold))
350         layout.addWidget(header)
351
352         self.tasks_total_lbl = QLabel("Tasks total: 0")
353         self.tasks_done_lbl = QLabel("Tasks done today: 0")
354         self.pomo_total_lbl = QLabel("Pomodoros total: 0")
355         self.focus_time_lbl = QLabel("Focus time (min): 0")
356
357         layout.addWidget(self.tasks_total_lbl)
358         layout.addWidget(self.tasks_done_lbl)
359         layout.addWidget(self.pomo_total_lbl)
360         layout.addWidget(self.focus_time_lbl)
361
362         charts_row = QHBoxLayout()
363         self.fig1, self.ax1 = plt.subplots(figsize=(4, 3))
364         self.canvas1 = FigureCanvas(self.fig1)
365         charts_row.addWidget(self.canvas1)
366
367         self.fig2, self.ax2 = plt.subplots(figsize=(4, 3))
368         self.canvas2 = FigureCanvas(self.fig2)
369         charts_row.addWidget(self.canvas2)
370
371         layout.addLayout(charts_row)
372
373     def update_stats(self):
374         tasks = db_query("SELECT id, completed, completed_at FROM tasks")
375
376         self.tasks_total_lbl.setText(f"Tasks total: {len(tasks)}")
377         self.tasks_done_lbl.setText(f"Tasks done today: {len([task for task in tasks if task['completed']])}
```

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):
436         def setupUi(self):
437             self.setLayout(layout)
438             header = QLabel("Budget Planner")
439             header.setFont(QFont("Arial", 16, QFont.Bold))
440             layout.addWidget(header)
441
442             form = QFormLayout()
443             self.date_edit = QDateEdit()
444             self.date_edit.setDate(QDate.currentDate())
445             self.cat_input = QLineEdit()
446             self.amount_input = QLineEdit()
447             form.addRow("Date:", self.date_edit)
448             form.addRow("Category:", self.cat_input)
449             form.addRow("Amount:", self.amount_input)
450             layout.addLayout(form)
451
452             btn_row = QHBoxLayout()
453             self.add_btn = QPushButton("Add Expense")
454             self.add_btn.clicked.connect(self.add_expense)
455             self.delete_btn = QPushButton("Delete Selected")
456             self.delete_btn.clicked.connect(self.delete_selected)
457             self.pdf_btn = QPushButton("Generate Receipt PDF")
458             self.pdf_btn.clicked.connect(self.generate_pdf)
459             btn_row.addWidget(self.add_btn)
460             btn_row.addWidget(self.delete_btn)
461             btn_row.addWidget(self.pdf_btn)
462             layout.addLayout(btn_row)
463
464             # table
465             self.table = QTableWidget(0, 3)
466             self.table.setHorizontalHeaderLabels(["Date", "Category", "Amount (₦)"])
467             self.table.setSizePolicy(QSizePolicy.Expanding, QSizePolicy.Expanding)
468             layout.addWidget(self.table)
469
470
471
472
473
474
475

```

Figure 4.10 – Python code for implementing Budget Planner module

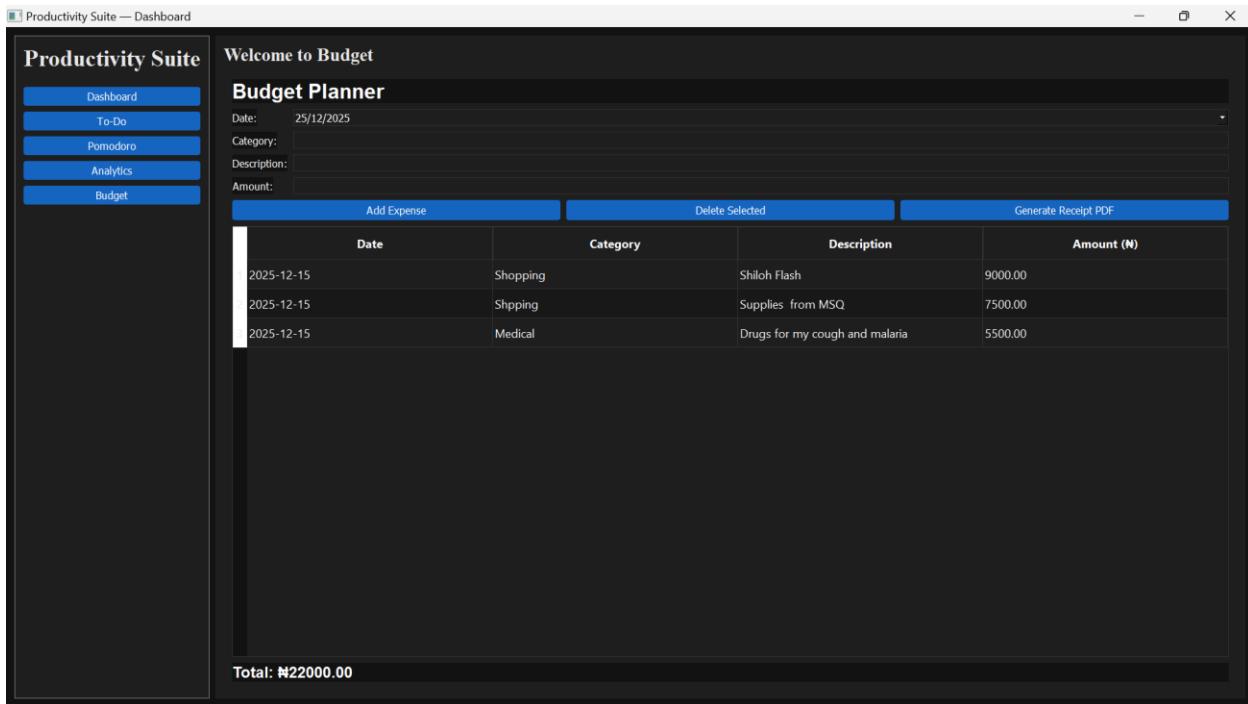


Figure 4.11 – Budget Planner interface with expense table and charts

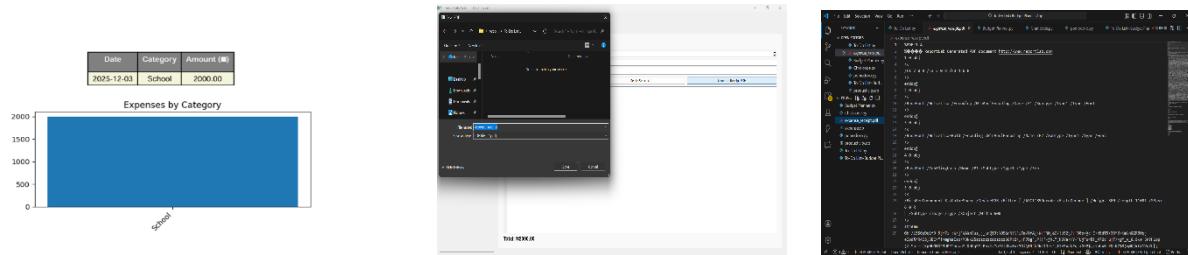


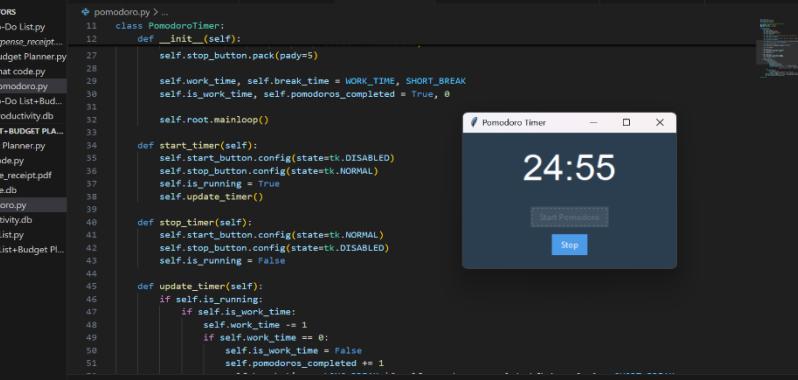
Figure 4.12 – Automatically generated PDF expense receipt

4.7 Testing

Testing included:

The screenshot shows a Python application interface. The title bar reads "To-Do List+Budget Planner App". The main window contains a "To Do List" section with a text input field and three buttons: "Add Task", "Done Task", and "Delete Task". Below this is a scrollable list of tasks. The left sidebar, titled "EXPLORER", lists several files: "To Do List.py", "expense_receipt.pdf", "Budget Planner.py", "Chat code.py", "pomodoro.py", "To Do List+Bud... productivity.db", "TO-DO LIST+BUDGET PLA... Budget Planner.py", "Chat code.py", "expense_receipt.pdf", "expense.db", "pomodoro.py", "productivity.db", "To Do List.py", and "To Do List+Budget Pl...". The bottom navigation bar includes "PROBLEMS", "OUTPUT", "DEBUG CONSOLE", and "PORTS". The status bar at the bottom shows "In 6 Col 21" and "Source: 4 UTE-B CPU: 1 Python 3.12.6 Go Live".

```
 1 <--> To Do List.py <--> expense_receipt.pdf <--> Budget Planner.py <--> Chat code.py <--> pomodoro.py <--> To Do List+Budget P ... BLACKBOX <--> ...
 1
 2
 3
 4
 5
 6
 7
 8
 9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
288
289
289
290
291
292
293
294
295
296
297
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
888
888
889
889
890
891
892
893
894
895
896
897
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
987
988
988
989
989
990
991
992
993
994
995
996
997
997
998
999
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1097
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1188
1189
1189
1190
1191
1192
1193
1194
1195
1196
1197
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1288
1289
1289
1290
1291
1292
1293
1294
1295
1296
1297
1297
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1388
1389
1389
1390
1391
1392
1393
1394
1395
1396
1397
1397
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1488
1489
1489
1490
1491
1492
1493
1494
1495
1496
1497
1497
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1588
1589
1589
1590
1591
1592
1593
1594
1595
1596
1597
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1688
1689
1689
1690
1691
1692
1693
1694
1695
1696
1697
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1797
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1888
1889
1889
1890
1891
1892
1893
1894
1895
1896
1897
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1997
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2088
2089
2089
2090
2091
2092
2093
2094
2095
2096
2097
2097
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2188
2189
2189
2190
2191
2192
2193
2194
2195
2196
2197
2197
2198
2199
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2288
2289
2289
2290
2291
2292
2293
2294
2295
2296
2297
2297
2298
2299
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2388
2389
2
```



The screenshot shows a Python IDE interface with multiple tabs open. The main code editor displays a file named `pomodoro.py` containing the following code:

```
11 class PomodoroTimer:
12     def __init__(self):
13         self.root = tk.Tk()
14
15         self.work_time, self.break_time = WORK_TIME, SHORT_BREAK
16         self.is_work_time, self.pomodoros_completed = True, 0
17
18         self.root.mainloop()
19
20     def start_timer(self):
21         self.start_button.config(state=tk.DISABLED)
22         self.stop_button.config(state=tk.NORMAL)
23         self.is_running = True
24         self.update_timer()
25
26     def stop_timer(self):
27         self.start_button.config(state=tk.NORMAL)
28         self.stop_button.config(state=tk.DISABLED)
29         self.is_running = False
30
31     def update_timer(self):
32         if self.is_running:
33             if self.is_work_time:
34                 self.work_time -= 1
35                 if self.work_time == 0:
36                     self.is_work_time = False
37                     self.pomodoros_completed += 1
38
39
40
41
42
43
44
45
46
47
48
49
50
51
```

To the right of the code editor, a window titled "Pomodoro Timer" is displayed, showing a digital clock at 24:55. Below the clock are two buttons: "Start Pomodoro" and "Stop".

The screenshot shows a Visual Studio Code (VS Code) interface with the following details:

- File Explorer:** Shows the project structure with files like `To-Do List.py`, `expense_receipt.pdf`, `Budget Planner.py`, `Chat code.py`, `pomodoro.py`, `To Do List+Budget...`, and `productivity.db`.
- Editor:** Displays the `Budget Planner.py` file content, which includes code for generating expense receipts using `simpleDocTemplate` and `Table` from ReportLab.
- Terminal:** Shows the command `hon312/python.exe "c:/Users/STAFF OFFICER HRD/Desktop/vscode/To-Do List+Budget Planner App/Budget Planner.py"`.
- Output:** Shows the output of the script execution.
- Debug Console:** Shows the output of the debug session.
- Problems:** Shows the count of problems found in the code.
- Right Panel:** A floating panel titled "Expense Tracker" contains fields for Date, Category, and Amount, along with buttons for Add Expense and Update Selected. It also displays a table of expenses with columns for Date, Category, and Amount, showing a single entry for 25/10/2024 under Food with an amount of 2300.00.
- Bottom Status Bar:** Shows the file path and various system status icons.

The screenshot shows a development environment with the following components:

- Code Editor:** On the left, the code editor displays several files including `To-Do List.py`, `Budget Planner.py`, and `Chat code.py`. The `Budget Planner.py` file is open, showing Python code for a `BudgetWidget` class.
- Terminal:** At the bottom, two terminal windows are visible. The left terminal shows a stack trace for an `AttributeError` related to a `QMainWindow` object. The right terminal shows a stack trace for an `KeyboardInterrupt`.
- Running Application:** A window titled "Productivity Suite - Dashboard" is running in the background. It displays a "Dashboard Overview" with sections for "Tasks: 2", "Pomodoros: 0", and "Focus (min): 0". It also includes a chart titled "Pomodoros (last 7 days)" showing data from November 26 to December 04.
- Status Bar:** The status bar at the bottom provides information about the current file (`Budget Planner.py`), line number (115), column (49), spaces (4), and encoding (UTF-8).

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/R8IE7dlg2f4>

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].