OEL

Section
Section-B

Subject

Software Construction & Development

Submitted by

Member Name	Registration Number		
M.Abdullah	Fa-22/BSSE/184		

Submitted To
Sir Farhan Sarwar

**Department of Software Engineering,
Lahore Garrison University, Lahore**

[Date: 16-06-2025]

Personal Finance Management System using Python and Tkinter

GitHub repository link: https://github.com/Abdullah789034/OEL-M.Abdullah-184-SCD

1. Introduction and Problem Statement

Effective personal finance management is essential in today's fast-paced world. The objective of this project was to develop a desktop-based Personal Finance Management System using Python and Tkinter. The system allows users to track their income and expenses, plan budgets, monitor savings, and generate financial summaries with persistent storage in SQLite.

2. Functional and Non-Functional Requirements

Functional Requirements

- Add, update, and delete income/expense records.
- Categorize transactions (e.g., Food, Travel, Utilities).
- Set and monitor monthly budgets.
- Display financial summaries and savings goals.

Non-Functional Requirements

- Responsive and user-friendly GUI.
- Proper input validation and error handling.
- Modular code following object-oriented practices.
- Scalable design for future enhancements.

3. Design Approach and Architecture

Object-Oriented Structure

Transaction: Handles individual financial entries.
Budget: Manages monthly budget limits.
CategoryManager: Manages transaction categories.
DatabaseHandler: Handles all SQLite database operations.
GUIManager: Manages all UI interactions with Tkinter.

Layered Architecture

Presentation Layer: Tkinter GUI
Logic Layer: Business logic, budget and transaction handling

Data Layer: SQLite database interface

4. Code Structure and Module Descriptions

gui.py: UI layout and event handling
logic.py: Transaction, budget, category logic
db.py: CRUD operations and database schema
test.py: Unit tests for core functionality

5. Exception and Error Handling Strategy

Handled invalid inputs using try-except blocks.
GUI displays user-friendly error messages.
Database errors (e.g., insertion failure) logged and shown to user.

6. Testing Strategy and Results

Testing Framework
Python unittest module was used.
Test Cases
Input validation: Prevents empty fields and invalid formats.
Budget overflow test: Checks warning when expenses exceed budget.
Database operations: Validates insert, update, and fetch functionality.
Results
All unit tests passed successfully ensuring stability of core features.

7. Refactoring Examples and Benefits

1. Reduced large function complexity by breaking into smaller units.
2. Removed duplicate code and improved naming conventions.
3. Used meaningful comments and docstrings for better maintainability.

8. Conclusion and Future Work

The Personal Finance Management System successfully meets the design requirements. The modular and maintainable structure makes it scalable for additional features such as: - Currency conversion - Cloud-based backup - Graphical data visualization
This project has enhanced my understanding of object-oriented design, GUI programming, testing, and modular code organization in a real-world software development scenario.