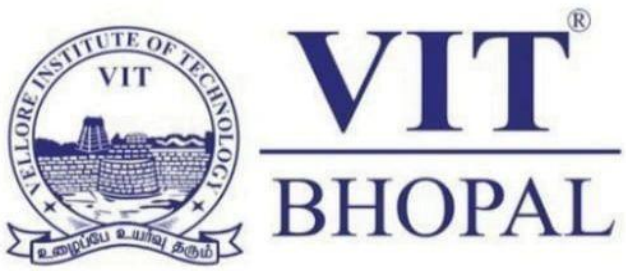


Project Report

Personal Finance Tracker

Submitted by: S Shreya

Registration Number: 25BCY10227

Course: B. Tech CSE – Cyber Security and Digital Forensics

Subject: Introduction To Problem Solving

Course Code: CSE1021

Introduction

The Personal Finance Detector is a Python application aimed at helping users efficiently monitor their financial activities. It enables the recording of income and expenses, categorization of transactions, and setting personalized savings goals. The tool uses an SQLite database for reliable data management and offers a simple command-line interface for user interaction. Key features include financial summaries and progress tracking toward savings targets. Additionally, the project incorporates various graphical visualizations such as bar charts and pie charts to provide clear insights into spending habits. By combining practical problem-solving techniques with Python's modular programming, database handling, and data visualization libraries, this project offers an accessible method for individuals to take control of their finances and make well-informed decisions for better money management.

Problem Statement

The Personal Finance Detector addresses the difficulty many individuals face in tracking income, expenses, and savings. Lack of organized records and insights leads to poor budgeting and financial stress. This project offers a simple Python tool to automate tracking, provide savings progress, and visualize data, helping users manage finances effectively.

Objective

The objective of the Personal Finance Detector project is to develop an easy-to-use Python application that helps individuals efficiently manage their finances. It aims to enable users to accurately record income and expense transactions, categorize them, and set personalized savings goals. The project seeks to provide clear financial summaries and real-time progress tracking towards savings targets.

Additionally, it incorporates data visualization features to give users insights into spending patterns and trends. By automating these tasks, the tool empowers users to make informed decisions, maintain better financial discipline, and achieve their financial goals through an intuitive, modular, and interactive interface.

Functional Requirements

1. **User Management**

- Secure login and profile creation
- Account detail updates

2. **Transaction Logging**

- Input income and expense with category and positive amount checks
- Automatic timestamping

3. **Savings Goals**

- Set and update personalized saving targets
- Track progress versus goals

4. **Data Handling**

- Store transactions securely in database
- Support retrieval for summaries and visual reports
- Ensure data integrity with validations

5. **Financial Reporting**

- Calculate and display total income, expenses, and balance
- Provide detailed breakdowns by category and date

6. **Visual Analytics**

- Show spending trends via bar, pie, line, and stacked bar charts
- Display expense distribution through histograms

7. **User Interface**

- Intuitive command/menu-driven navigation
- Contextual help and error messages

This ensures a reliable, user-friendly system that supports effective financial tracking, monitoring, and decision-making.

Non-Functional Requirements System Architecture

1. **Usability**

- Intuitive and user-friendly interface with minimal learning curve.
- Clear prompts, error messages, and help documentation.

2. **Performance**

- Responsive to user inputs within seconds.
- Efficient handling of growing transaction data.

3. **Reliability and Stability**

- Maintain data integrity without loss or corruption.
- Graceful exception handling and error notifications.

4. **Security**

- Secure storage and access control for sensitive data.
- Protection against common vulnerabilities.

5. **Portability**

- Compatible with Windows, macOS, and Linux supporting Python and SQLite.

- Easy installation without complex setup.

6. **Maintainability**

- Modular, organized, and well-documented codebase.
- Easily adjustable configurations.

7. **Scalability**

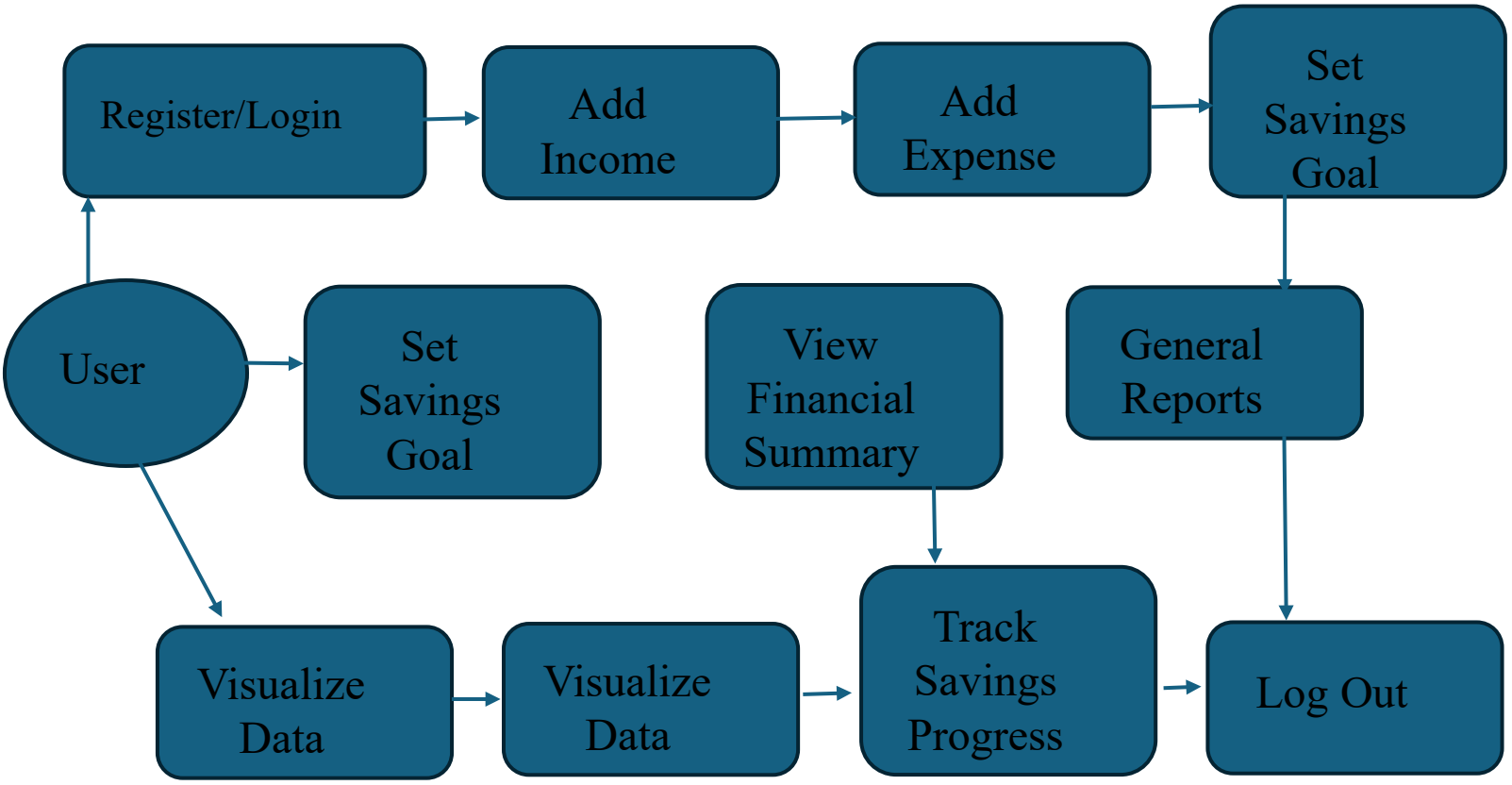
- Designed for future feature expansions without major rewrites.

8. **Data Backup and Recovery**

- Support for regular backups and data restoration mechanisms.

These qualities ensure a dependable, secure, and user-friendly finance tracker for long-term use.

Design Diagrams



Implementation

The Personal Finance Detector translates design into a functional Python application featuring database setup, modules for income and expense management, goal setting, visualization, and a user interface.

- **Database Setup**
  Uses SQLite for lightweight, reliable storage. Defines tables like ledger for transactions and targets for savings goals with integrity constraints. Includes safe connection management.
- **Income and Expense Modules**
  Functions to add records with input validation. Summaries calculate totals and current balance from the database.
- **Goal Management**
  Users set personal saving goals stored in the database. Progress is tracked and feedback provided.

- **Visualization Module**
  Utilizes Matplotlib to create bar, pie, line charts, and histograms by querying aggregated data.
- **User Interface**
  Command-line interface accepts commands to log transactions, set goals, view reports, and display charts. Includes help commands and error handling.
- **Integration and Testing**
  Ensures smooth module interaction with clear calls. Performs unit and integration tests to validate data accuracy and usability.

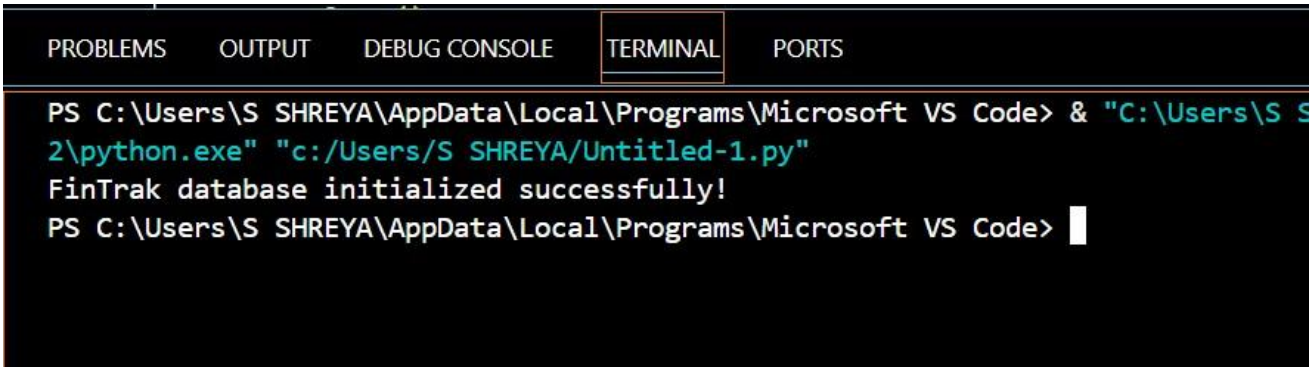
This structured implementation ensures a modular, reliable, and user-friendly personal finance tracking solution.

Screenshots

C: > Users > S SHREYA > Untitled-1.py > ...

```python
import sqlite3
from os import makedirs
from os.path import dirname, exists

DB_LOC = "data/fintrak_storage.db"

def connect_fintrak_db():
    """
    Establish a connection to the FinTrak SQLite database.
    Ensures the directory structure exists before connecting.
    """
    folder = dirname(DB_LOC)
    if not exists(folder):
        makedirs(folder)  # Create missing folders for DB path

    return sqlite3.connect(DB_LOC)

def setup_fintrak_db():
    """
    Create essential tables: ledger and targets for incomes/expenses and saving goal
    Applies data integrity constraints for realistic finance tracking.
    """
    conn = connect_fintrak_db()
    cur = conn.cursor()

    # Ledger stores all financial transactions with strict type and positive amounts
    cur.execute("""
        CREATE TABLE IF NOT EXISTS ledger (
            entry_id INTEGER PRIMARY KEY AUTOINCREMENT,
            entry_type TEXT NOT NULL CHECK(entry_type IN ('income','expense')),
            category TEXT NOT NULL,
            amount REAL NOT NULL CHECK(amount > 0),
            entry_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP
```

```
18  def setup_fintrak_db():
34          )
35      """)
36
37      # Single-row table for setting personal saving target, goal amount cannot be negative
38      cur.execute("""
39          CREATE TABLE IF NOT EXISTS targets (
40              id INTEGER PRIMARY KEY CHECK(id = 1),
41              saving_goal REAL NOT NULL CHECK(saving_goal >= 0)
42          )
43      """)
44
45      conn.commit()
46      conn.close()
47
48  if __name__ == "__main__":
49      setup_fintrak_db()
50      print("FinTrak database initialized successfully!")
```



Testing Approach

The testing approach for the personal finance tracker focuses on verifying the correctness, reliability, and usability of all features through the following types of testing:

1. **Unit Testing**
    - Test individual functions such as adding income, adding expenses, setting goals, and performing database queries.
    - Validate input handling, data insertion, and retrieval correctness.
2. **Integration Testing**
    - Verify that different modules (database management, input handling, visualization) work together seamlessly.
    - Test workflows like logging transactions followed by overview display or visual report generation.
3. **System Testing**
    - Assess the complete system's performance as a whole, including the command-line interface responsiveness and error management.
    - Ensure that all functionalities operate correctly under typical user scenarios.
4. **Usability Testing**

- Evaluate ease of use through user feedback to confirm commands are intuitive and error messages are clear.
- Adjust interface prompts to improve user experience.

5. **Boundary and Edge Case Testing**

- Test input validation with extreme or invalid values to ensure the system handles them gracefully without crashing.

6. **Performance Testing**

- Check application responsiveness with increasing transaction data volume to confirm the system remains efficient.

Challenges Faced

Developing a personal finance tracker involves several challenges:

1. **Data Accuracy and Validation**

    - Ensuring accurate user inputs for amounts and categories to prevent invalid or inconsistent financial records.

2. **User Engagement and Usability**

    - Designing an interface that is both simple and comprehensive so users consistently use the tracker without feeling overwhelmed or confused.

3. **Data Security and Privacy**

    - Protecting sensitive financial data to maintain user trust and comply with privacy norms.

4. **Handling Diverse Financial Scenarios**

    - Accommodating different income types, complex expense patterns, and varying saving goals across users.

5. **Scalability and Performance**

- Maintaining efficient performance as the volume of transactions grows over time.

6. **Error Handling and Robustness**

- Preventing crashes or data corruption by managing unexpected inputs and operational faults gracefully.

Learnings & Key Takeaways

Key learnings from developing a personal finance tracker include the importance of clear and simple user interfaces, robust data validation, and secure handling of sensitive financial information. Structuring the system modularly allows easy maintenance and scalability. Visualizations play a vital role in helping users understand their spending habits and achieve financial goals. Testing at all levels ensures reliability and smooth user experience. Overall, balancing functionality with usability and security is crucial to delivering a practical and engaging financial management tool that users trust and rely on.

Future Enhancements

- Integrate AI for automated expense prediction and categorization.
- Add multi-currency and international transaction support.
- Develop mobile applications for easy, anytime access.
- Enable real-time bank account synchronization.
- Incorporate advanced budgeting and loan/EMI tracking.

References

Majumdar, Tamal. "Developing a Python-Based Personal Finance Tracker for Efficient Money Management." *Medium Blog*, May 2025. Accessible at: https://medium.com/@tamalmajumdar/dev-personal-finance-tracker-efficient-management-2025

IJESAT Journal. “Personal Finance Tracker - Research Paper.” 2025. https://www.ijesat.com/ijesat/files/V25I5046_1747220247.pdf

GeeksforGeeks. “Personal Finance Tracker using Django.” January 2024. https://www.geeksforgeeks.org/python/personal-finance-tracker-using-django