ATM Machine Simulation

A Python-based ATM system designed for the Information System Analysis & Design.

- **Prepared For**: Anamul Haque, Lecturer, CSE, Feni University
- **Course**: Information System Analysis & Design
- **Department**: Computer Science & Engineering
- **Feni University**

- **Prepared by** : Md. Abdul Al Noman
- **ID**: 232031044
- **Batch**: 31st (UG)

Technologies Used:

Python (Tkinter for GUI, SQLite3 for database)

Introduction

This project serves as a practical simulation of a real-world ATM system, built entirely using Python. It offers a comprehensive demonstration of how software can interact with users through a graphical interface and manage data persistently.

Graphical User Interface (GUI)

Intuitive and responsive interface for seamless user interaction, mimicking a physical ATM.

SQLite Database Integration

Robust backend for storing essential data like user accounts, transaction records, and login/logout histories.

Secure Authentication

Demonstrates a foundational layer of security for user access and basic banking operations.

Objectives: Building a Functional ATM

Our primary goal was to develop a fully functional ATM prototype, focusing on both user experience and backend data management. This involved several key objectives:

1. **User-Friendly Interface**

   Designing an intuitive ATM interface using Tkinter to ensure ease of use for all banking operations.

2. **Secure Login Mechanism**

   Implementing a secure login process requiring a valid card number and PIN for account access.

3. **Essential Banking Operations**

   - Enabling users to perform core functions:
   - checking balances,
   - withdrawing,
   - depositing cash, and
   - viewing transaction history.

4. **Accountability & Logging**

   Maintaining comprehensive transaction logs and login/logout records to ensure transparency and accountability.

Project Methodology: From Design to Implementation

The project followed a structured approach, breaking down the development into distinct, manageable steps. This ensured a systematic progression from conceptual design to functional testing.

GUI Design with Tkinter

Crafting the visual components, including the login screen, main menu, and various transaction windows, for an engaging user experience.

SQLite Database Creation

Establishing the database schema with tables for **user accounts** (card, pin, name, balance), **transactions** (id, card, type, amount, balance_after, timestamp), and **activity logs** (id, card, name, action, timestamp).

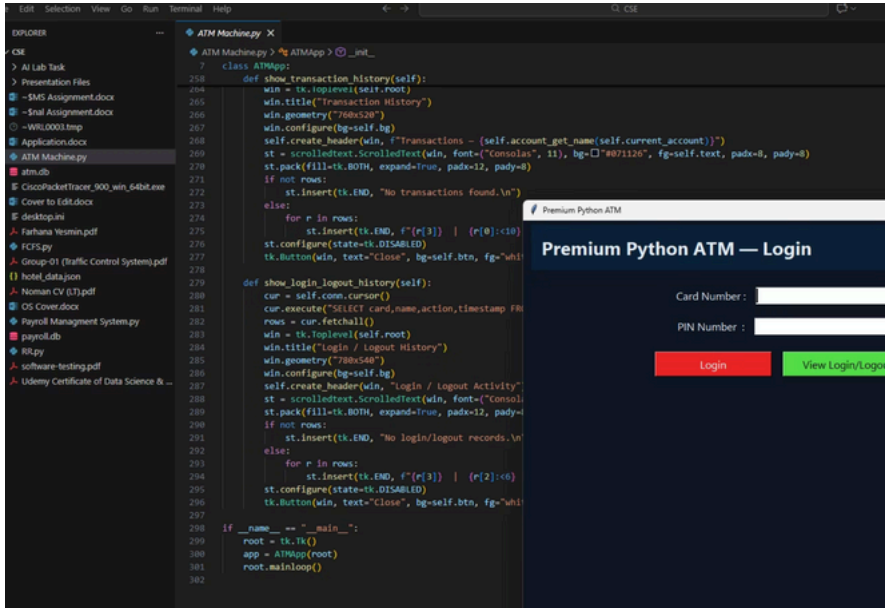
Core Functionality Implementation

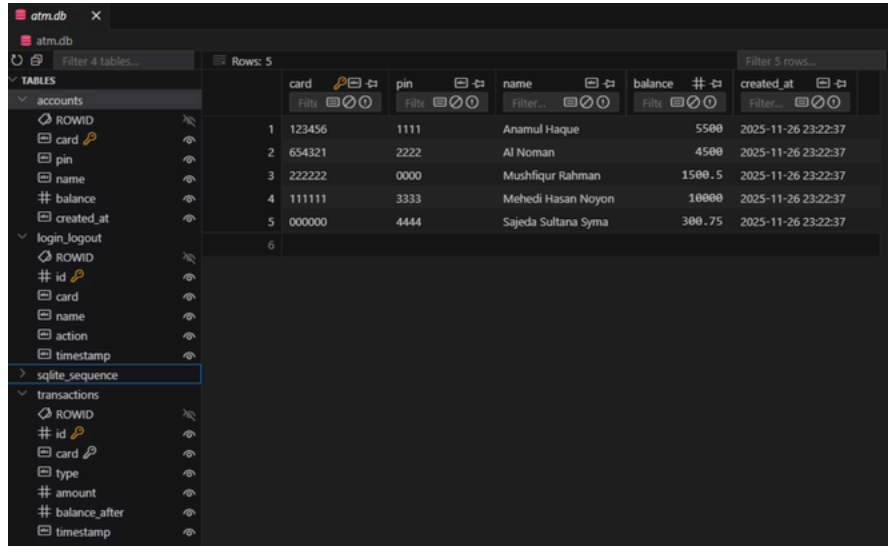
Developing the backend logic for account creation, login authentication, deposit/withdrawal processes, balance updates, and comprehensive logging of all activities.

System Testing and Validation

Rigorous testing with demo accounts to validate all functionalities, ensuring accuracy and reliability across various transaction scenarios.

Visual Demonstration of Implementing



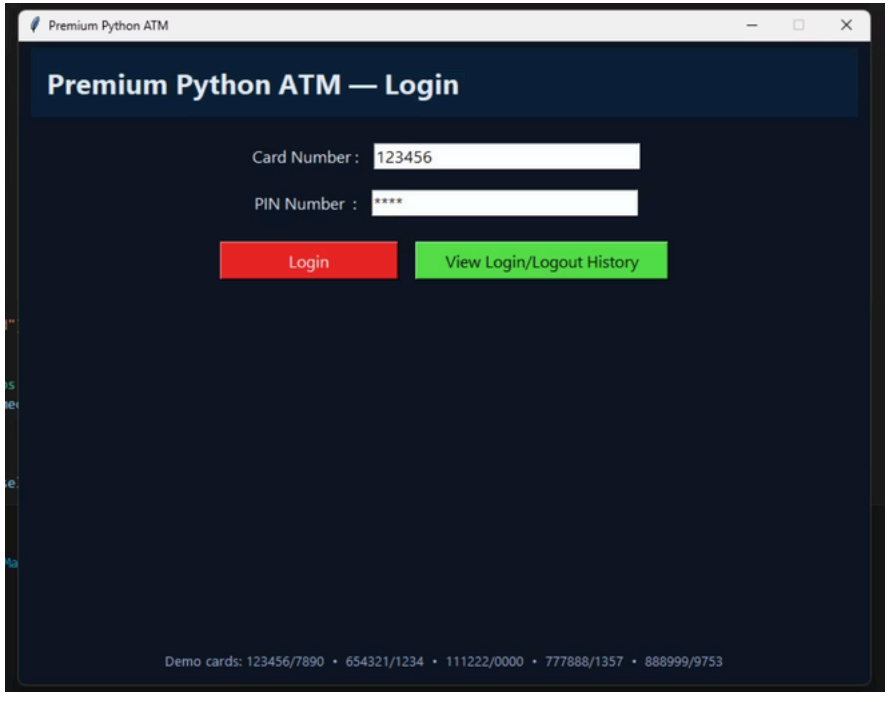


The 1st screenshot displays the Python-based ATM system in action, featuring a Tkinter GUI for login and transaction history.

The 2nd screenshot displays the internal structure of the ATM system's SQLite database, highlighting the "Account" table. It stores user details including card number, PIN, name, balance, and account creation timestamp for secure transaction handling. - "atm.db" file.

Results and Analysis

The project successfully culminated in a functional ATM simulation, demonstrating a robust GUI and a reliable backend. Key achievements include seamless operation of banking tasks and comprehensive error handling.



- The login screen allows users to access the ATM system using demo card numbers and PINs. It also provides access to login/logout history.

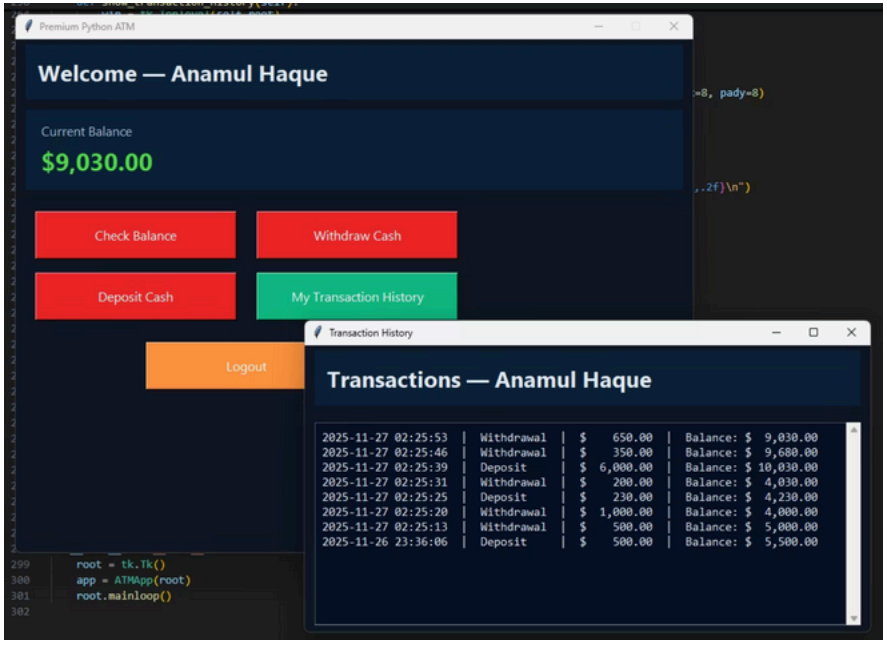
Key Achievements

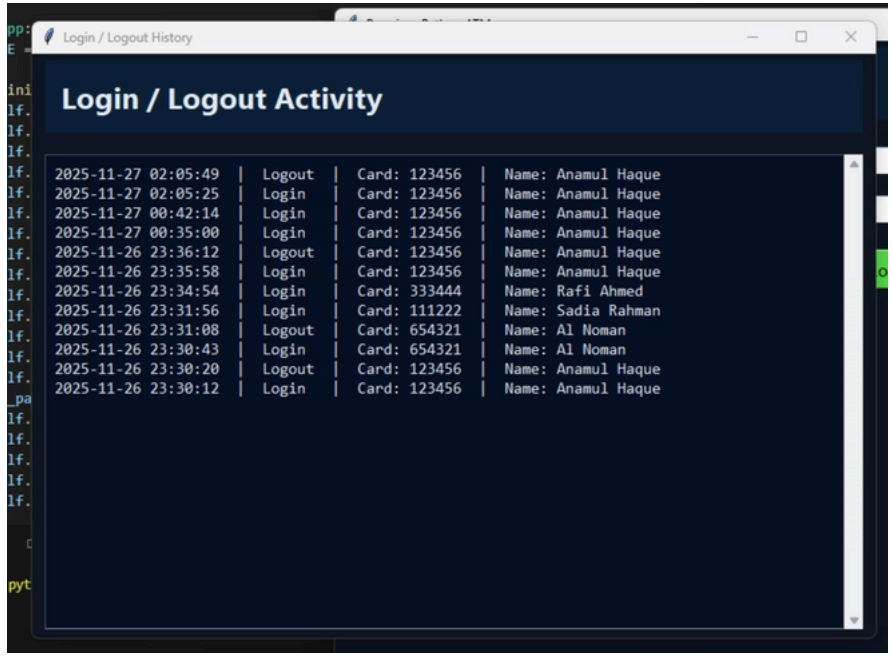
- **Operational GUI:** All ATM operations are fully functional through the graphical interface.
- **Accountability:** Login/logout tracking provides a clear audit trail.
- **Detailed History:** Transaction history is clearly displayed in a scrollable window.

Robust Error Handling

- **Authentication:** Handles invalid card/PIN attempts.
- **Financial Integrity:** Prevents transactions with insufficient balance or invalid amounts.
- **Validation:** Ensures all inputs are valid before processing.

Visual Demonstration of Output





- **The 1st screenshot showcases the ATM system's main dashboard and transaction history for user Anamul Haque. It demonstrates real-time balance updates and a clear log of deposits and withdrawals with timestamps.**

- **The 2nd one displays detailed login and logout records for each user, including timestamps and card numbers.**

Conclusion & Future Works

This project has laid a solid foundation for understanding and implementing a banking system. While successful, it also opens doors for significant enhancements and further development.

Conclusion

The ATM simulation effectively integrates a GUI with database management, showcasing a functional prototype. This project provides invaluable insights into software development for financial applications.

Future Works

- **Multi-user Account Management:** Implement user registration and multiple account types.
- **Enhanced Security:** Integrate encryption for sensitive data like PINs.
- **Real-time Integration:** Connect to external bank servers for live data processing.
- **User Convenience:** Add features like receipt printing and mobile transaction notifications.

References

- Python Official Documentation
  https://docs.python.org
- Tkinter GUI Library Resources
- SQLite Database Documentation
  https://sqlite.org/docs.html