Bank Management System

Abstract

The Bank Management System is a console-based application that simulates ATM operations using **Python** and connected to a **MySQL** database. The system allows users to perform basic banking operations such as viewing accounts, depositing funds, withdrawing money, transferring funds between accounts, and viewing transaction history. All operations are recorded in the database in real-time, ensuring that account balances are always up-to-date.

Introduction

This project simulates ATM operations using Python and MySQL. Users can interact with a menu-driven system to perform deposits, withdrawals, and transfers. All accounts and transactions are stored in a database, demonstrating practical database integration in a software application. This system manages customer accounts, balances, and transactions similar to basic banking operations.

Objectives

- Provide a simple and interactive interface for ATM operations.
- Maintain accurate account balances and transaction records.
- Implement real-time updates for deposits, withdrawals, and transfers.
- Allow users to view transaction history for all accounts.

System Requirements

Hardware Requirements

- Processor: Intel Core i3 or higher
- RAM: 4 GB minimum
- Storage: 1 GB free disk space

Software Requirements

- Python 3.x
- MySQL Server
- MySQL Connector for Python (mysql-connector-python)
- IDE: VSCode

System Design

Database Schema

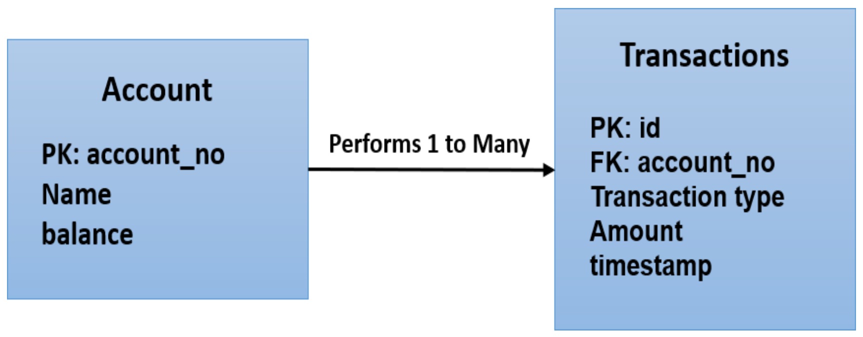


Fig.1.1: Schema Diagram

Explanation:

- `accounts.account_no` → Primary Key, uniquely identifies each account.
- `transactions.id` → Auto-incremented Primary Key.
- `transactions.account_no` → Foreign Key linking to Accounts table.
- `transactions.timestamp` → Automatically records transaction time.

Entity-Relationship Diagram (ERD)

 Accounts (account_no, name, balance)

 Transactions (id, account_no, type, amount, timestamp)

 Relationship: One account can have multiple transactions (1-to-many).

Relationship: Each account can have multiple transactions.

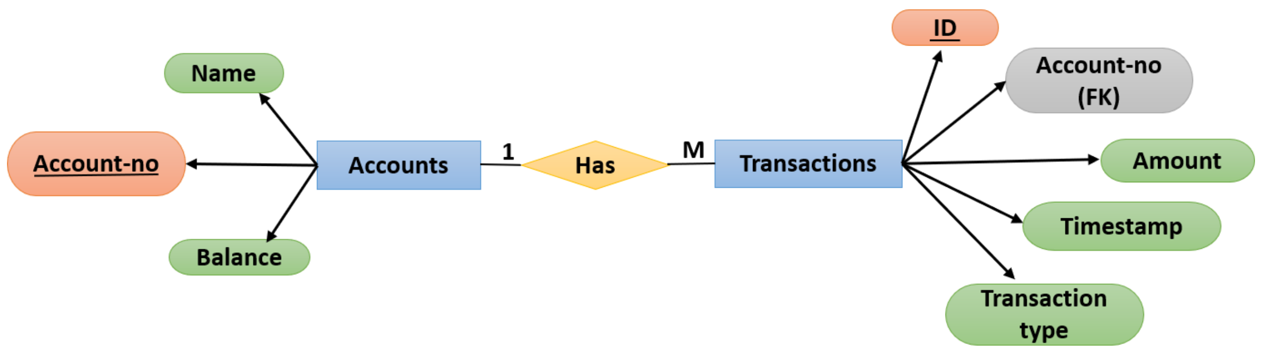


Fig.1.2: E-R Diagram

Workflow / Menu Flow

1. User enters account number.
2. System displays the menu:
    - Show Accounts
    - Deposit
    - Withdraw
    - Transfer
    - Transaction History
    - Exit
3. User selects an option and performs the operation.
4. System updates the database and displays confirmation.

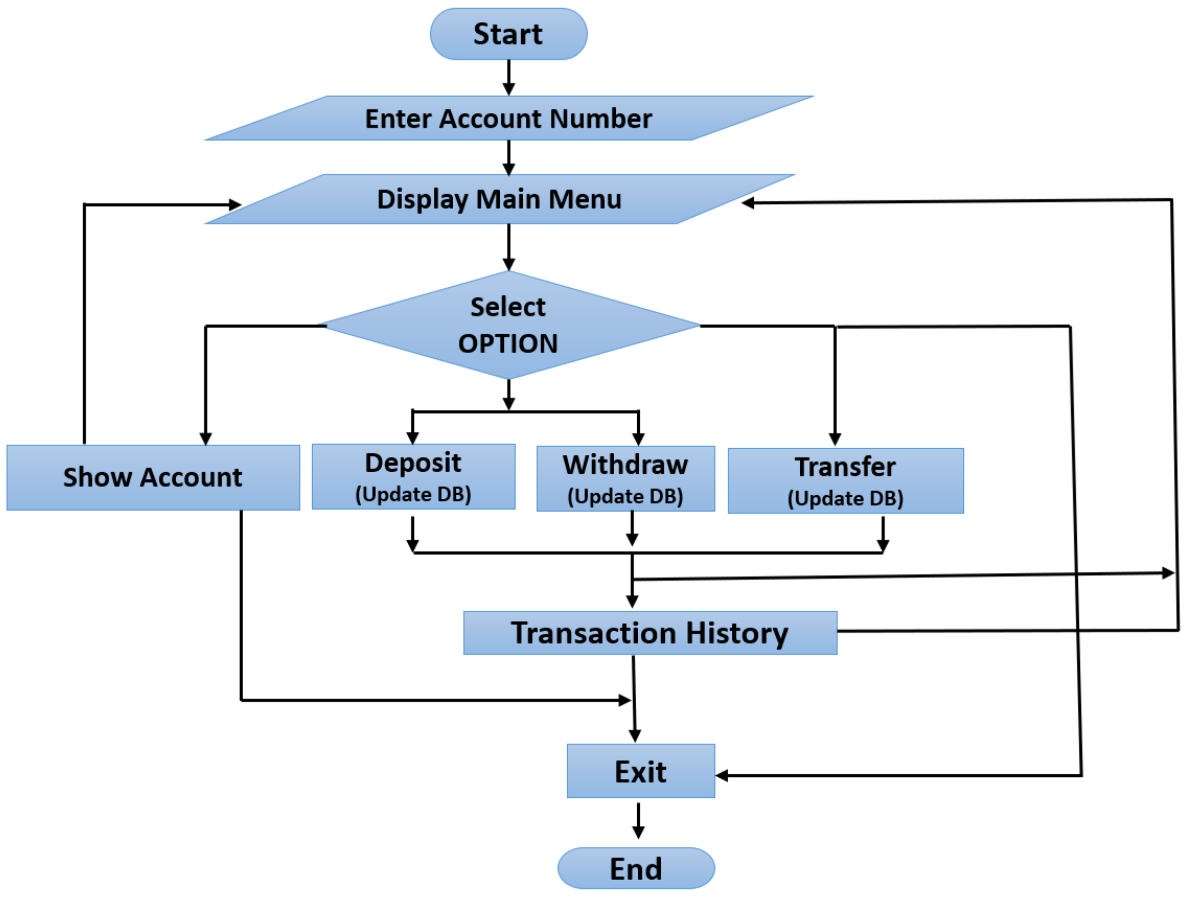
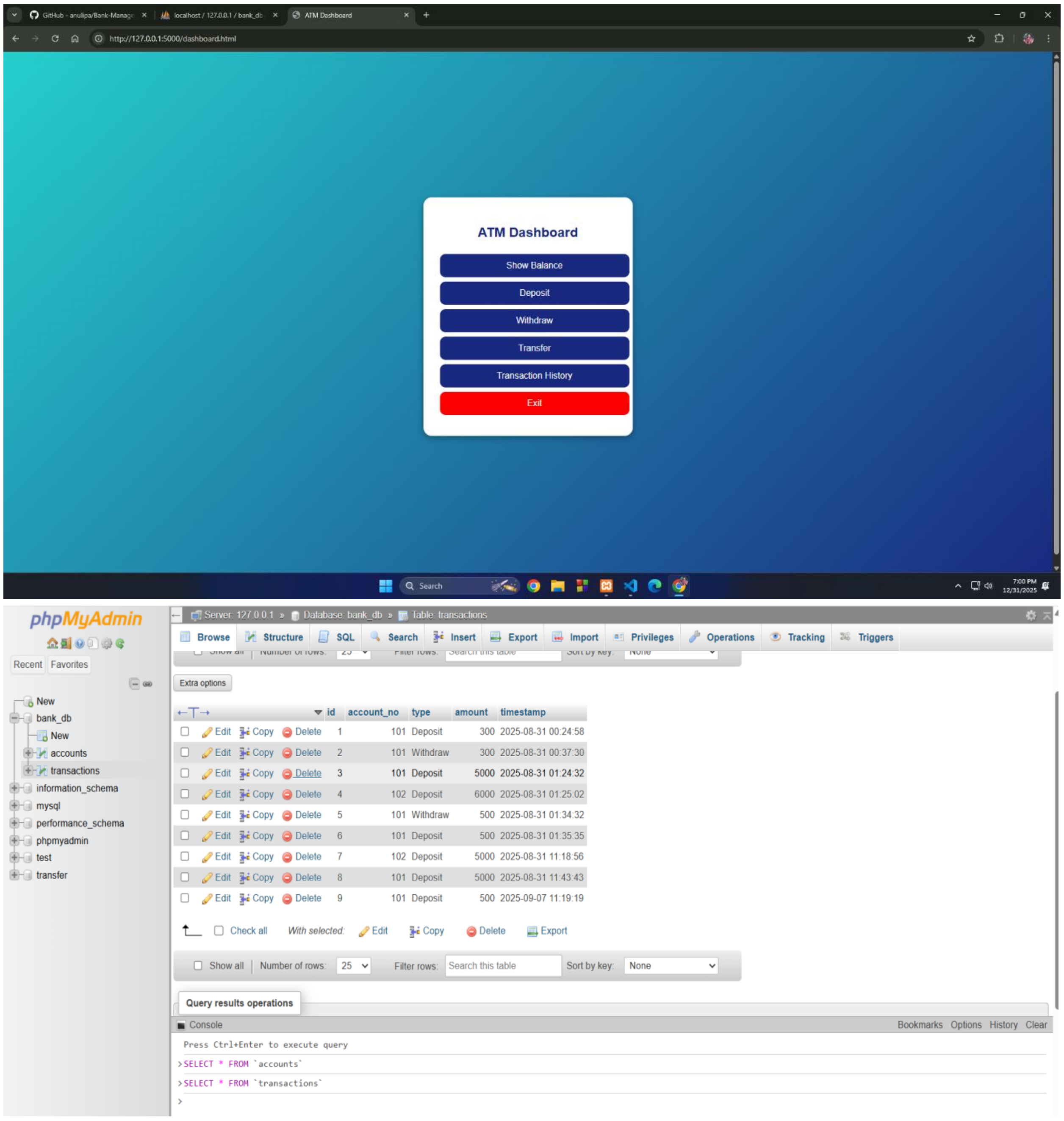


Fig.1.3: Working Procedure Flowchart of Bank Management System

Output / Testing

- Sample screenshots showing:
  - Displaying account details
  - Successful deposit/withdraw
  - Transfer between accounts
  - Transaction history display



id	account_no	type	amount	timestamp
1	101	Deposit	300	2025-08-31 00:24:58
2	101	Withdraw	300	2025-08-31 00:37:30
3	101	Deposit	5000	2025-08-31 01:24:32
4	102	Deposit	6000	2025-08-31 01:25:02
5	101	Withdraw	500	2025-08-31 01:34:32
6	101	Deposit	500	2025-08-31 01:35:35
7	102	Deposit	5000	2025-08-31 11:18:56
8	101	Deposit	5000	2025-08-31 11:43:43
9	101	Deposit	500	2025-09-07 11:19:19

```
Press Ctrl+Enter to execute query
>SELECT * FROM `accounts`
>SELECT * FROM `transactions`
>
```

Fig.1.4: Project Output

Implementation

Function	Description
`show_accounts()`	Displays all accounts with account number, name, and balance
`deposit()`	Adds a specified amount to an account and records the transaction
`withdraw()`	Deducts a specified amount from an account if sufficient balance exists
`transfer()`	Transfers funds from one account to another and records both transactions
`transaction_history()`	Displays all transactions in descending order of timestamp

Results and Discussion

- The system successfully manages accounts and transactions in real-time.
- The menu-based interface allows users to perform operations efficiently.
- Limitations:
    - No PIN or card verification (security risk).
    - Console-based; GUI or web interface could improve usability.
    - Only basic banking operations implemented.

Conclusion

The Bank Management System demonstrates practical integration of Python and MySQL to simulate banking operations. It provides a foundation for extending the system with security features, GUI, or web-based interfaces for real-world applications.

References

- Python Official Documentation: https://docs.python.org/3/
- MySQL Official Documentation: https://dev.mysql.com/doc/
- Tutorials and online resources for Python-MySQL integration