PROJECT REPORT

BANKING SYSTEM (PYTHON)

Project Title: *BANKING SYSTEM*
Course: *C.S.E (CORE)*
Submitted by: *TANISH*
Roll No: 25BCE11137
Submitted to: *Dr. ADYASHA SAHU*
Project: VITYARTHI
Date: 24/11/2025

Introduction

This project implements a simple command-line based **Banking System** using Python.
It allows users to create accounts, log in using credentials, deposit and withdraw money, and check their balance.
The project was developed to understand the concepts of data structures, control flow, functions, and basic authentication.

Problem Statement

The objective is to design a basic banking application that enables users to perform simple banking operations without using a database or GUI.
The system should be able to:

- Create and store user accounts
- Authenticate users based on credentials
- Allow deposit, withdrawal, and balance inquiry
- Manage multiple users during a single execution

Functional Requirements

Requirement	Description
Account Creation	Users can create an account with a username and PIN.
Login	Users must enter correct credentials to access their account.
Deposit Money	Users can add money to their account balance.
Withdraw Money	Users can withdraw money if sufficient funds exist.
Check Balance	Users can view their current balance.
Logout	Users can exit their session safely.
Exit System	The program should terminate on user request.

Non-functional Requirements

- **Usability:** The system should have simple text-based interaction.
- **Reliability:** Must correctly validate inputs and credentials.
- **Performance:** All operations should execute instantly since data is stored in memory.

- **Security:** Simple PIN-based authentication is provided.
- **Maintainability:** Code is modular using functions for each operation.

System Architecture

The system uses a **list of dictionaries** to store account details in memory.
Each account is represented as:

```
{
 "user": username,
 "pin": pin_code,
 "balance": 0.0
}
```

The architecture follows a **linear flow**:
User Interaction → Authentication → Operation (Deposit/Withdraw/View) → Logout

---

Design Diagrams

A. Use Case Diagram

Actors: User
Use Cases: Create Account, Login, Deposit, Withdraw, View Balance, Logout

(You can draw this simple diagram in Word or draw.io)

B. Workflow Diagram

1. Start
2. Main Menu
3. Choose Create/Login/Quit
4. If Login → Show Account Menu
5. Perform operations
6. Logout → Return to Main Menu

C. Sequence Diagram

User → System

- Enter credentials → Validate → Allow operations
- Request deposit → Update balance

- Request withdrawal → Check funds → Update
- Logout

D. Class/Component Diagram

Since no OOP is used, components include:

- Main Module
- Account Store (List)
- Functional Components: create, find, deposit, withdraw, view

E. ER Diagram (if storage used)

✔ Not required — system uses in-memory list instead of a database.
(You may include a simple table representation if needed.)

Design Decisions & Rationale

1. **List of dictionaries** chosen because it is simple and easy to manipulate.
2. **No database** used to keep the project lightweight for beginners.
3. **Procedural programming** used instead of OOP for simplicity.

4. **PIN as string** to allow leading zeros (e.g., “0123”).

5. **Input validation** added to avoid crashes from invalid entries.

Implementation Details

The system is implemented in Python using:

- Functions: make_acount, find_acount, add_money, take_money, view_balance
- A global list Bank_Account storing all account dictionaries
- A main loop providing menu-driven navigation

(Key code features include):

- Credential matching using simple iteration
- Error handling using try-except
- Balance updates stored in memory

Screenshots / Results

```
PS C:\Users\Tanish\OneDrive\Desktop\New folder> python
..... BANKING SYSTEM .....
1) Create Account
2) Login
3) Quit
Select: 1

..... Create a New Account .....
Name: Tanish
Choose a 4-digit PIN: 2314
Account created!

..... BANKING SYSTEM .....
1) Create Account
2) Login
3) Quit
Select: 2

..... Login .....
Name: Tanish
PIN: 2314
Logged in successfully.


..... Account Options .....
1) Deposit
2) Withdraw
3) Check Balance
4) Logout
Select: 1

..... Deposit .....
Amount to be deposited: 250000
New balance: 250000.0
```

```
..... Account Options .....
1) Deposit
2) Withdraw
3) Check Balance
4) Logout
Select: 2

..... Withdraw .....
Amountt to be withdrawn: 212100
Remaning balance: 37900.0


..... Account Options .....
1) Deposit
2) Withdraw
3) Check Balance
4) Logout
Select: 3

..... Acount Summary .....
Account Holder: Tanish
Current Balance: 37900.0


..... Account Options .....
1) Deposit
2) Withdraw
3) Check Balance
4) Logout
Select: 4
Logged out.

..... BANKING SYSTEM .....
1) Create Account
2) Login
3) Quit
Select: 3
Goodbye........
```

Testing Approach

Test Cases:

Test No.	Input	Expected Output
1	Create new account	Account saved successfully
2	Login with correct PIN	Login success
3	Login with wrong PIN	Error message
4	Deposit valid amount	Balance updated
5	Withdraw more than balance	Insufficient funds
6	Withdraw valid amount	Balance deducted

All primary functions were tested using manual testing.

Challenges Faced

- Implementing login without using a database.
- Handling invalid user inputs.

- Managing multiple user accounts using lists.
- Ensuring the program doesn't crash on wrong input formats.

Learnings & Key Takeaways

- Improved understanding of Python functions, loops, and conditionals.
- Learned how to structure a basic software project.
- Understood user authentication concepts.
- Gained experience in debugging and handling runtime errors.
- Enhanced logical thinking for real-world system design.

Future Enhancements

- Add Object-Oriented structure with classes.
- Integrate file storage or a database (SQLite, MySQL).
- Add password hashing for better security.

- Provide interest calculation, transaction history.
- Create a GUI using Tkinter or a web interface.
- Implement admin panel for managing all users.

References

- Python Official Documentation.
- Online tutorials on functions and data structures.
- Classroom notes and examples.
- W3Schools Python Reference.
- GeeksforGeeks Python articles.