10Alytics

SecureSave

Developing a Bank Account Management System for SecureSave Bank

Practical Python Capstone Project

Taiwo Akinyefa

10Alytics

Business Introduction

SecureSave

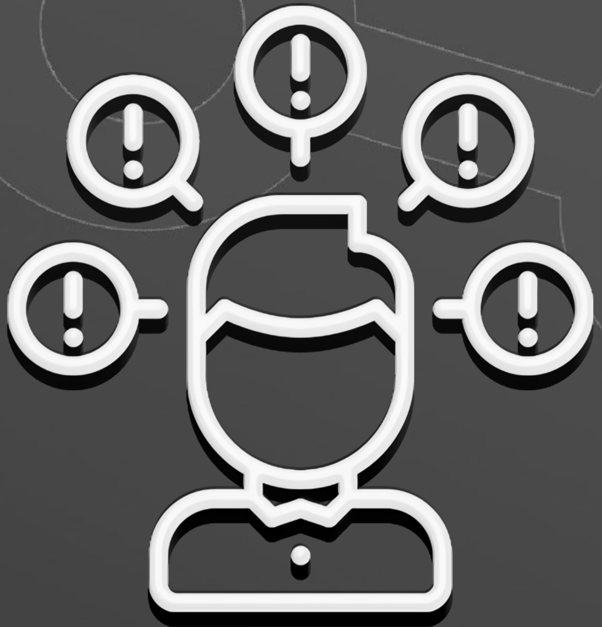
SecureSave Bank is a newly established financial institution aiming to provide its customers with a secure and efficient way to manage their money. As part of their initial service offerings, they require a basic system to handle core banking operations: creating accounts, depositing and withdrawing funds, checking balances, and viewing transaction history.

10Alytics

PROBLEM STATEMENT

SecureSave Bank's initial operations will rely on a simple, command-line interface (CLI) system to manage customer accounts. They need a robust and user-friendly application.

The current manual methods are inefficient and prone to errors. An automated system will streamline these processes, improve accuracy, and provide a foundation for future expansion of their services.



10Alytics

Detailed Challenges

- Create new customer accounts with unique account numbers and initial balances.
- Process deposit transactions, updating account balances and recording the transaction.
- Process withdrawal transactions, ensuring sufficient funds are available and recording the transaction.
- Allow customers (or staff on their behalf) to check the current balance of a specific account.
- Provide a history of all transactions (deposits and withdrawals) for a given account.

10Alytics

Proposed Solution

SecureSave Bank proposes the development of a Python-based CLI application to manage bank accounts. This system will store account information in memory during runtime and provide a menu-driven interface for performing the required operations. The system will utilize Python data structures (dictionaries and lists) and fundamental programming concepts (functions, loops, conditional statements) to manage account data and transaction history.

10Alytics

Implementation Plan

- Define Data Structures: Design dictionaries to store account details (account number, balance, transaction history) and a main dictionary to hold all accounts.
- Implement Account Creation: Create a function to generate unique account numbers and store new account details.
- Implement Deposit Functionality: Develop a function to handle deposits into specified accounts, updating balances and recording transactions.
- Implement Withdrawal Functionality: Develop a function to handle withdrawals from specified accounts, including balance checks and transaction recording.
- Implement Balance Inquiry: Create a function to retrieve and display the balance of a given account.
- Implement Transaction History Viewing: Create a function to display the transaction history for a specified account.
- Develop User Interface: Build a CLI menu using loops and conditional statements to allow users to interact with the system and choose operations.
- Implement Basic Error Handling: Include checks for invalid account numbers and insufficient funds.

10Alytics

Goodluck