Simple Banking Application

Group – C

Team Members: Muhammad Farzad Ali: 126559

Aishwarya Kole: 126792

Anusha Rallapalli: 125800

Introduction

The CLI-Based Simple Banking Application aims to provide users with essential banking functionalities through a command-line interface (CLI). It offers a straightforward platform for managing personal finances, encompassing registration, login, transaction management, and balance inquiries. The application ensures data integrity and security by leveraging a CSV file system for storing user data and transaction history securely.

System Requirements

The application operates on Java Development Kit (JDK) version 15 or higher, ensuring compatibility across different operating systems. It relies solely on a command-line interface (CLI) for user interaction, facilitating accessibility and ease of use. Additionally, access to a CSV file system is essential for maintaining user data and transaction records securely.

Features

User Registration and Login

Upon launching the application, users are prompted to register or log in with unique credentials. The registration process entails providing a username and password, ensuring the uniqueness of each user's identity. Subsequently, registered users can securely log in using their credentials.

Post-Login Functionalities

Following successful login, users gain access to a range of banking functionalities. They can check their current account balance, add funds, withdraw money, view transaction history, and log out securely to end their session.

Constraints and Validations

To uphold data integrity and ensure a seamless user experience, the application enforces various constraints and validations. Users are restricted from withdrawing more money than their current balance, and all monetary inputs must be positive numbers. Additionally, the system guarantees unique usernames to prevent duplicate accounts and prompts users to retry upon selecting an invalid option.

Command Line Interface (CLI) Structure

The CLI is structured into main and user menus to streamline user interaction. The main menu offers options for registration and login, while the user menu, accessible post-login, provides

various banking functionalities such as balance inquiries, fund transfers, and transaction history reviews.

Operational Requirements

Operational requirements mandate that the application returns to the login screen after a user logs out. Additionally, the system alerts users if they select an invalid option and prompts them to retry after invalid input, ensuring a seamless and error-free user experience.

Transaction Handling

All user transactions are meticulously recorded and displayed in their transaction history. The system maintains separate transaction histories for each user, showcasing transaction details including date, time, type, and amount. This feature empowers users to monitor their financial activities over time effectively.

Security Features

Security remains a top priority within the application, with stringent measures implemented to safeguard user data and prevent unauthorized access. These measures include account locking after multiple failed login attempts, secure password storage, and input validation to prevent invalid or malicious data entry.

User Experience Enhancements

The application is designed to prioritize user experience, offering clear and user-friendly messages for all actions. Users receive alerts for invalid options and are prompted to retry, ensuring a smooth and intuitive interface. Additionally, users can conveniently view their recent transactions in a summarized form, enabling quick access to their most recent financial activities.

Application Architecture

Class Descriptions

- AppCLI: Manages CLI interactions and application logic, handling menu displays, user input validation, and execution of banking operations.
- Bank: Encapsulates core banking functionalities such as user registration, login, and data storage, ensuring data integrity and security.
- User: Represents individual users and manages account details, including balances and transaction history.
- Transaction: Handles transaction details and formatting, providing methods for converting transaction data to CSV format and string format for display.
- CSVHandler: Manages CSV file operations for loading, saving, and updating user and transaction data, ensuring data persistence and integrity.

Reason for GUI Removal

The decision to remove the graphical user interface (GUI) and retain the command-line interface (CLI) was influenced by the email received from Group P. As per the email communication, the Group P informed Group C that the supervisor suggested them to focus on one interface for submission. Hence, the CLI was deemed more suitable due to its simplicity and ease of implementation.

Possible Improvements:

1. **Transaction Categorization:** Add functionality to categorize transactions (e.g., groceries, utilities) for better financial tracking and analysis.
2. **Scheduled Transactions:** Allow users to schedule future transactions (e.g., bill payments, transfers) to automate repetitive tasks and improve convenience.
3. **Performance Optimization:** Conduct performance optimizations to improve application responsiveness and reduce loading times, ensuring a seamless user experience.
4. **Billing for Utility Payments:** Introduce functionality for users to pay utility bills directly through the application, streamlining the bill payment process and offering added convenience.
5. **Transfer Funds:** Enable users to transfer funds between their accounts or to external accounts, facilitating seamless money management and transactions within the application.