**Project Title**

**Swift Bank: Secure Banking System**

**Project Overview**

The Swift Bank system is a secure and user-friendly banking application designed to facilitate essential banking operations for customers. This project aims to create a platform that allows users to manage their accounts, perform transactions, and access additional financial services like insurance and loans. The system provides a streamlined interface for account management while ensuring the confidentiality and security of user data.

The primary focus of this project is to digitize core banking processes, minimize errors, and improve customer convenience. By integrating advanced features such as password hashing for authentication and persistent storage, Swift Bank ensures data integrity and user security.

**Objectives**

1. To provide a secure platform for users to manage their bank accounts.
2. To enable seamless transactions, including deposits, withdrawals, and fund transfers.
3. To offer financial services like insurance and loans with predefined categories and limits.
4. To ensure data security through password hashing and persistent data storage.
5. To provide a user-friendly interface for efficient account management.

**Functional Requirements**

**Key Features and Functionalities**

* **User Authentication**:
  + Secure account creation with hashed passwords.
  + User login with account number and password validation.
* **Account Management**:
  + View account details, including name, email, contact number, Aadhaar number, and balance.
* **Transaction Handling**:
  + Deposit and withdraw funds.
  + Transfer funds to other accounts.
  + View transaction history and credits received.
* **Financial Services**:
  + **Insurance**:
    - Apply for insurance plans such as Home, Term Life, Motor, and General Insurance.
  + **Loans**:
    - Apply for loan categories such as Home Loan, Car Loan, Education Loan, Business Loan, and Loan Against Property.
    - Loan limit capped at 50,00,000.
* **Data Management**:
  + Store and retrieve customer and transaction data from persistent files.

**Example Outputs**

* Display of transaction history with timestamps.
* Notification of successful deposits, withdrawals, fund transfers, and loan or insurance applications.

**Non-Functional Requirements**

* **Performance**: The system should handle up to 1,000 customers and 5,000 transactions efficiently.
* **Security**: Sensitive user data, such as passwords, must be hashed. Persistent files must be protected against unauthorized access.
* **Scalability**: The system should support future expansions, including additional financial services and more customers.
* **Reliability**: The application must ensure consistent performance and accurate data storage.
* **Usability**: The interface should be intuitive, enabling non-technical users to interact seamlessly.

**System Architecture Overview**

The SwiftBank application architecture is modular, comprising the following components:

1. **User Interface**: Text-based menus for user interactions.
2. **Business Logic**:
   * Functions for account management, transactions, and financial services.
3. **Data Management**:
   * Structures for customer and transaction records.
   * File operations for saving and loading data.
4. **Security Layer**:
   * Password hashing for secure authentication.

**Technologies Used**

* **Programming Language**: C
* **File Storage**: Binary files for persistent storage of customer and transaction data.
* **Hashing Algorithm**: Custom hash function for password security.

**Data Requirements**

**Customer Data**

* Account number
* Name
* Email
* Contact number
* Aadhaar number
* Hashed password
* Account balance

**Transaction Data**

* Initiator account number
* Recipient account number (for fund transfers)
* Transaction type (e.g., Deposit, Withdraw, Transfer)
* Amount
* Timestamp

**Assumptions and Constraints**

**Assumptions**

1. Customers will provide valid inputs (e.g., Aadhaar numbers, emails).
2. All customer operations will be performed sequentially (single-threaded environment).

**Constraints**

1. Loan limit is capped at 50,00,000.
2. File operations are used for data storage instead of a database.
3. The application is console-based and does not include a graphical user interface.

**System Components Summary**

**Number of Functions**

* **25 functions**: Including core functionalities like createAccount, login, manageAccount, transaction handling, and financial services (insurance and loans).

**Data Structures**

* **2 structures**:
  + Customer: Stores account details (e.g., account number, name, hashed password, balance).
  + Transaction: Manages transaction records (e.g., type, amount, timestamp).

**Arrays**

* **2 arrays**:
  + customers[1000]: Stores up to 1,000 customer records.
  + transactions[5000]: Stores up to 5,000 transaction records.

**Strings**

* **String Usage**:
  + Multiple strings for customer data: names, emails, contact numbers, Aadhaar numbers, and hashed passwords.
  + Strings for transaction types and timestamps.

**Timeline and Milestones**

**Phase 1: Requirement Gathering and Design**

* Define functional and non-functional requirements.
* Design system architecture.

**Timeline**: 1 week

**Phase 2: Development**

* Implement core features: account management, transactions, and authentication.
* Integrate insurance and loan modules.

**Timeline**: 2 weeks

**Phase 3: Testing**

* Conduct functional testing for all features.
* Verify data persistence and security mechanisms.

**Timeline**: 1 week

**Phase 4: Final Presentation and Submission**

* Prepare documentation.
* Demonstrate the application.

**Timeline**: 2 days

**References**

1. [C Programming Language Documentation](https://en.cppreference.com/)
2. File handling tutorials for persistent storage.
3. Hashing algorithm implementation guides.

**Appendices**

**Additional Diagrams**

* **Flowchart**: Account creation, login, and management.
* **Data Flow Diagram**: Interaction between UI, business logic, and file storage.

**Sample Outputs**

* Login success message: "Welcome [Name]!"
* Transaction success: "Transfer successful. Current balance: 10,000."
* Insurance application: "Insurance plan 'Home Insurance' applied successfully.