ATM Interface Project Report

Introduction:

The ATM interface application is a Java-based console application designed to provide account management functionality to users, such as checking account balances, withdrawing money, depositing money, and transferring money to another account. The purpose of this project report is to provide an overview of the design, implementation, and testing of the ATM interface application.

Requirements:

The ATM interface application must have the following requirements:

1. Account management functionality, such as checking account balances, withdrawing money, depositing money, and transferring money to another account.
2. Transaction history functionality, to allow users to view their transaction history.
3. Security features, such as password authentication, to ensure the security of user accounts.

Design:

The ATM interface application consists of the following classes:

1. ATM: The main class of the application, which provides the user interface and handles user input.
2. Account: A class to represent a user account, which contains information such as account number, balance, and transaction history.
3. Database: A class to handle the connection to the database and retrieve account information.

The ATM interface application includes the following menu options:

1. Check the account balance
2. Withdraw money
3. Deposit money
4. Transfer money
5. View transaction history
6. Exit

The database schema includes the following tables:

1. Account: A table to store account information, such as account number, balance, and password.
2. Transaction: A table to store transaction information, such as transaction type, amount, and date.

Implementation:

The ATM interface application involves several key components. Here are the steps involved:

1. Database Connection:

- Use JDBC (Java Database Connectivity) to establish a connection with the database.
- Set up the necessary database drivers and connection parameters.
- Use the Connection object to establish a connection to the database.

2. Account Validation:

- Prompt the user to enter their account number and password.
- Validate the account number and password against the database records.

- If the credentials are valid, proceed with the application; otherwise, display an error message and terminate the program.

3. Menu and User Input:

- Display the main menu options to the user.
- Prompt the user to enter their choice.
- Read and validate the user's input.

4. Menu Option Implementation:

- Based on the user's choice, implement the functionality for each menu option.
- For example, if the user chooses to check their account balance, retrieve the account balance from the database and display it to the user.
- For withdrawal, deposit, or transfer options, prompt the user for the necessary details (e.g., amount) and update the account balance accordingly.

5. Error Handling:

- Implement error-handling mechanisms to handle exceptional scenarios, such as insufficient funds, invalid inputs, or database connection failures.
- Display appropriate error messages to the user.

6. User Interface and Flow:

- Design the user interface to provide a clear and intuitive experience for the user.
- Ensure proper flow between menu options and handle user navigation.

7. Testing and Debugging:

Perform thorough testing of the application, covering all menu options and scenarios.
Debug and fix any issues encountered during testing.
Ensure the application functions correctly and handles all edge cases.

Conclusion:

The ATM interface application is a console-based Java application that provides users with account management functionalities such as checking account balances, withdrawing money, depositing money, transferring funds, and viewing transaction history. Throughout the project, we successfully implemented the necessary components to create a functional ATM interface.