Team 7 ATM Machine

Denah Thach, Nicholas Day, Jiya Stroder

INFO – C 451

Professor Jafrina Jabin

Table of Contents

Customer Problem Statement

Managing and accessing personal finances is a critical aspect of individuals' daily lives. Traditional banking systems, while effective, may not always provide the level of convenience and accessibility that modern users expect. Team Seven's proposed project aims to address this challenge by developing a comprehensive ATM System, providing users with a convenient and secure way to manage their finances.

System Requirements

Functional Requirements

No:	**Priority Weight:** **H** **M** **L**	**Name:**	**Description:**
TIC #1:	***H***	**Security Breaches**	The most critical issue is security. A system can be implemented to enhance the security of ATM transactions, such as two-factor authentication, biometric verification, and encryption of data.
TIC #2:	***H***	**Insufficient Cash**	ATMs running out of cash is a common problem. A system can be implemented to monitor cash levels and alert the bank when it's time to refill.
TIC #3:	***H***	**Outdated Software**	ATMs with outdated software can be prone to glitches and security vulnerabilities. A system for regular software updates can help resolve this issue.
TIC #4:	***H***	**Card Skimming**	This is a method used by criminals to capture data from the magnetic stripe on the back of an ATM card. A system can be implemented to

			detect and prevent skimming devices.
TIC #5:	***M***	**Transaction Failures**	Sometimes, transactions fail due to network issues or server problems. A robust system can ensure seamless connectivity and backup servers to prevent transaction failures.
TIC #6:	***M***	**Poor User-Interface**	Some ATMs have complex interfaces that confuse users. A system with a user-friendly interface can improve the user experience.
TIC #7:	***L***	**Inadequate Receipt Information**	Sometimes, ATMs do not provide enough information on receipts. A system can be implemented to provide detailed transaction information.
TIC #8:	***H***	**Cash Withdrawals**	The ability for users to withdraw a specified amount of cash from their bank account using the ATM.
TIC #9:	***H***	**Balance Inquiries**	This ability for users to check their account balance and view recent transactions using the ATM.
TIC #10:	***H***	**Deposit Functions**	The ability for users to deposit cash or checks into their account using the ATM. This may involve advanced features like automated check scanning for verification and validation.

Nonfunctional Requirements

Priority	Category	Description
High	Reliability	The ATM system should have high availability, ensuring that it is operational and accessible 24/7.
High	Usability	The user interface must be intuitive and user-friendly to facilitate ease of use for customers.
Medium	Performance	Transactions should be processed swiftly, and response times must be kept within acceptable limits.
Medium	Supportability	The system should be designed with maintainability and ease of troubleshooting in mind for efficient support.
Low	Functionality	The core functions of the ATM, such as cash withdrawals, balance inquiries, and fund transfers, must be reliable and accurate.

Functionality: Functionality refers to the core features and capabilities of the ATM system, which are directly related to its primary purpose and tasks. Here are some examples of the functional requirements of an ATM:

Usability: This refers to the ease of use and the user experience of the ATM. It includes factors such as the clarity of instructions on the screen, the intuitiveness of the interface, and the accessibility features for individuals with disabilities.

Reliability: This refers to the ATM's ability to perform consistently and accurately without failures or errors. It includes factors such as uptime, error handling, and the prevention of unauthorized access.

Performance: This refers to the speed and responsiveness of the ATM. It includes factors such as transaction processing time, response time for user interactions, and handling concurrent user requests efficiently.

Supportability: This refers to the ATM's ease of maintenance, manageability, and adaptability. It includes factors such as the availability of software updates, the ease of troubleshooting, and the ability to integrate new features or technologies.

Functional Requirements Specifications

Stakeholders

1. **Customers:** The users of the ATM system.
2. **Bank Staff:** Administrators and those responsible for working directly with the ATM systems. Workers who use ATM data to monitor performance.
3. **Maintenance Technicians:** Workers responsible for maintaining and repairing the ATM machines.

Actors and Goals

Primary Actors:

1. **Customers**
    - **Goals:** Login, withdraw cash, deposit funds, view account balances.

2. **ATM Technicians**
    - **Goals:** perform routine maintenance, diagnose, update software, and ensure
      ATMs are operational.

Secondary Actors:

1. **Bank Staff:**
    - **Goals:** Configure ATM setting, set transaction fees, generate reports on usages and transactions.

Use Cases

Customer (total: 6)

- **Withdraw Cash**
    - Allows customers to withdraw cash from their bank account using the ATM. (2)
- **Deposit Funds**
    - Enable customers to deposit funds into their bank account using the ATM. (2)
- **Check Account Balance**
    - Allows customers to check the balance of their bank account using the ATM. (1)

- **View Transaction History**
  - Enables customers to view a history of recent transactions performed through the ATM. (2)
- **Login**
  - Enables customers to access their account and ATM functions. (1)

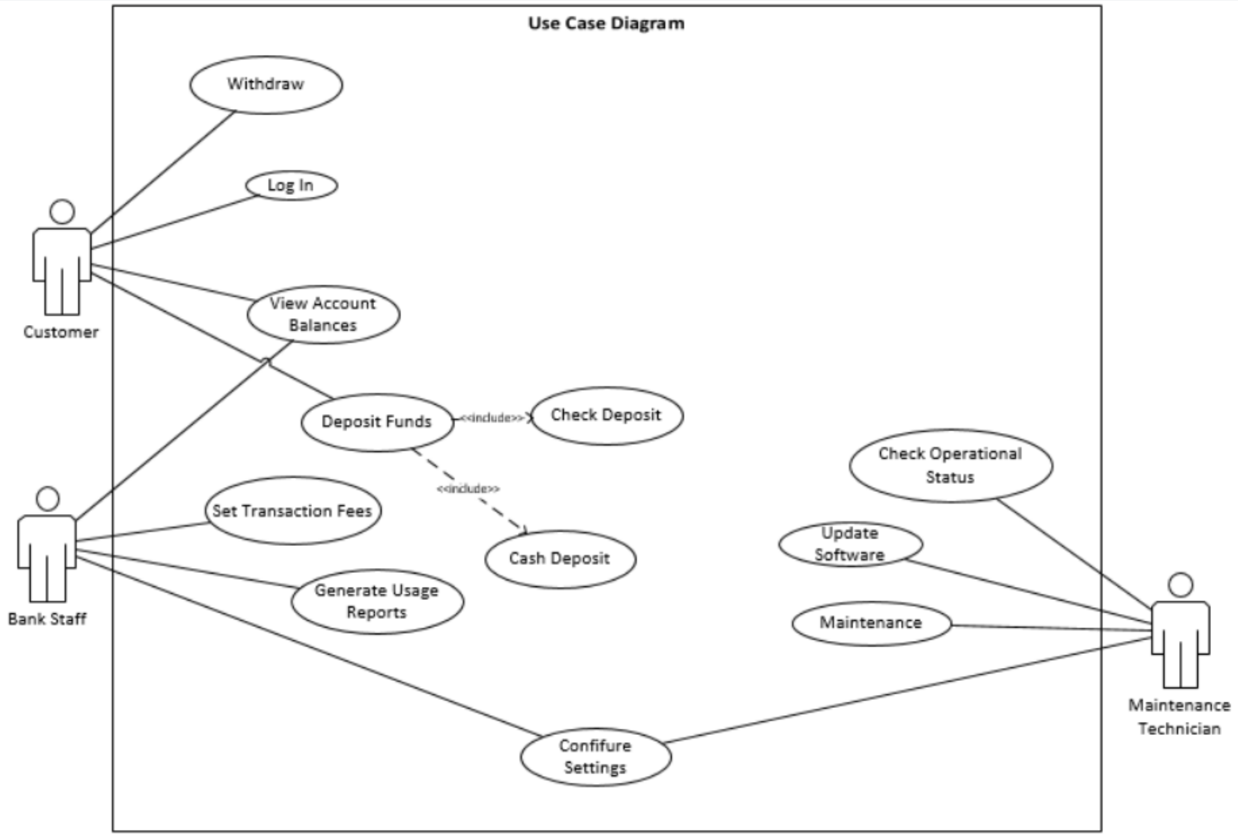
Bank Staff (total: 12)

- **Monitor System Performance**
  - Allow bank management to ensure the status of the entire system. (3)
- **Configure ATM Setting**
  - Permits bank management to configure settings related to ATM operations. (3)
- **Generate Transaction Reports**
  - Enables the generation of report details of ATM usage, transaction volume, and other relevant metrics. (4)
- **Set Transaction Fees**
  - Determining the fees associated with different ATM transactions and services. (2)

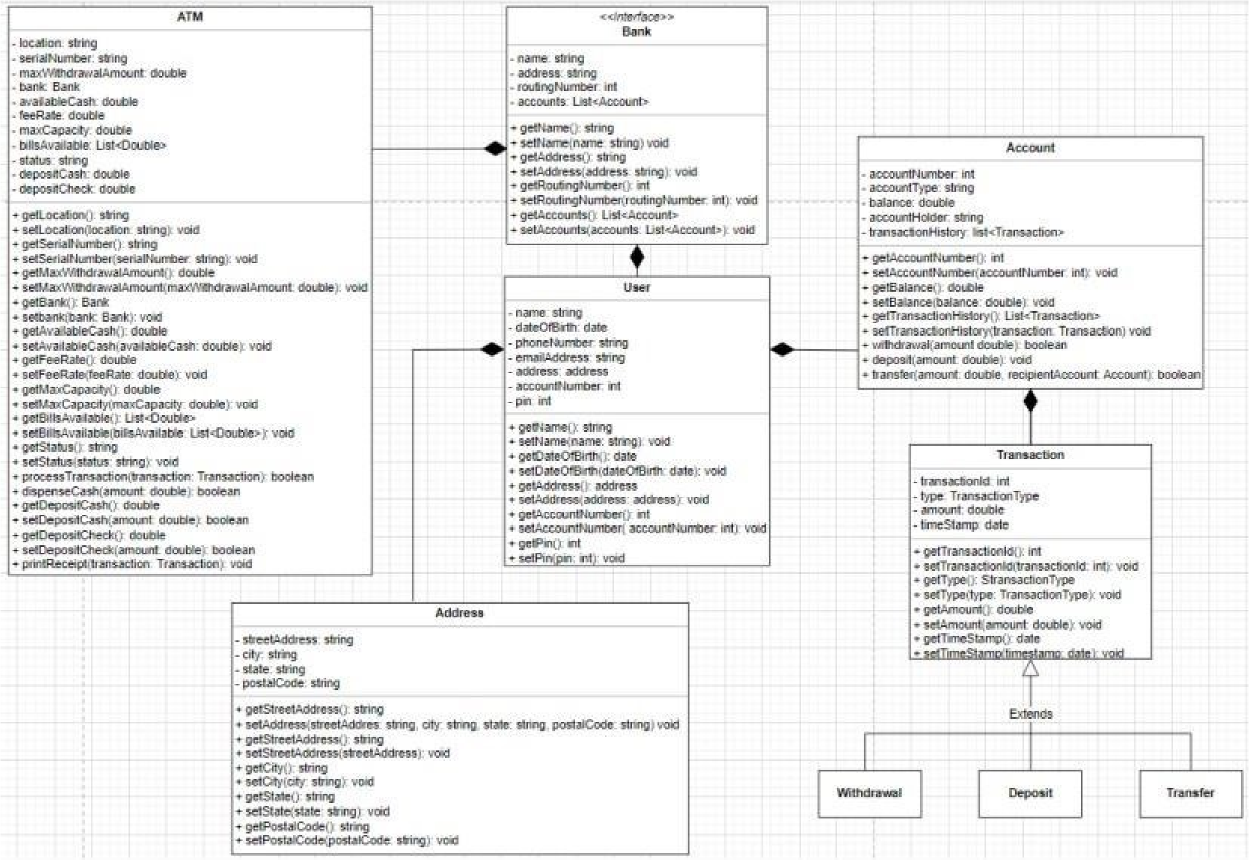
ATM Technician (total: 4)

- **Update Software**
  - Involves updating the software running on each ATM to ensure the latest features and security patches are working properly. (4)

Use Case Diagram



Class Diagram


ATM
- location: string
- serialNumber: string
- maxWithdrawalAmount: double
- bank: Bank
- availableCash: double
- feeRate: double
- maxCapacity: double
- billsAvailable: List<Double>
- status: string
- depositCash: double
- depositCheck: double
+ getLocation(): string
+ setLocation(location: string): void
+ getSerialNumber(): string
+ setSerialNumber(serialNumber: string): void
+ getMaxWithdrawalAmount(): double
+ setMaxWithdrawalAmount(maxWithdrawalAmount: double): void
+ getBank(): Bank
+ setbank(bank: Bank): void
+ getAvailableCash(): double
+ setAvailableCash(availableCash: double): void
+ getFeeRate(): double
+ setFeeRate(feeRate: double): void
+ getMaxCapacity(): double
+ setMaxCapacity(maxCapacity: double): void
+ getBillsAvailable(): List<Double>
+ setBillsAvailable(billsAvailable: List<Double>): void
+ getStatus(): string
+ setStatus(status: string): void
+ processTransaction(transaction: Transaction): boolean
+ dispenseCash(amount: double): boolean
+ getDepositCash(): double
+ setDepositCash(amount: double): boolean
+ getDepositCheck(): double
+ setDepositCheck(amount: double): boolean
+ printReceipt(transaction: Transaction): void
<<interface>>
Bank
- name: string
- address: string
- routingNumber: int
- accounts: List<Account>
+ getName(): string
+ setName(name: string) void
+ getAddress(): string
+ setAddress(address: string): void
+ getRoutingNumber(): int
+ setRoutingNumber(routingNumber: int): void
+ getAccounts(): List<Account>
+ setAccounts(accounts: List<Account>): void
User
- name: string
- dateOfBirth: date
- phoneNumber: string
- emailAddress: string
- address: address
- accountNumber: int
- pin: int
+ getName(): string
+ setName(name: string): void
+ getDateOfBirth(): date
+ setDateOfBirth(dateOfBirth: date): void
+ getAddress(): address
+ setAddress(address: address): void
+ getAccountNumber(): int
+ setAccountNumber(accountNumber: int): void
+ getPin(): int
+ setPin(pin: int): void
Account
- accountNumber: int
- accountType: string
- balance: double
- accountHolder: string
- transactionHistory: list<Transaction>
+ getAccountNumber(): int
+ setAccountNumber(accountNumber: int): void
+ getBalance(): double
+ setBalance(balance: double): void
+ getTransactionHistory(): List<Transaction>
+ setTransactionHistory(transaction: Transaction) void
+ withdrawal(amount double): boolean
+ deposit(amount: double): void
+ transfer(amount: double, recipientAccount: Account): boolean
Transaction
- transactionId: int
- type: TransactionType
- amount: double
- timeStamp: date
+ getTransactionId(): int
+ setTransactionId(transactionId: int): void
+ getType(): StransactionType
+ setType(type: TransactionType): void
+ getAmount(): double
+ setAmount(amount: double): void
+ getTimeStamp(): date
+ setTimeStamp(timestamp: date): void
Extends
Withdrawal
Deposit
Transfer
Address
- streetAddress: string
- city: string
- state: string
- postalCode: string
+ getStreetAddress(): string
+ setAddress(streetAddres: string, city: string, state: string, postalCode: string) void
+ getStreetAddress(): string
+ setStreetAddress(streetAddress): void
+ getCity(): string
+ setCity(city: string): void
+ getState(): string
+ setState(state: string): void
+ getPostalCode(): string
+ setPostalCode(postalCode: string): void

Activity Diagrams

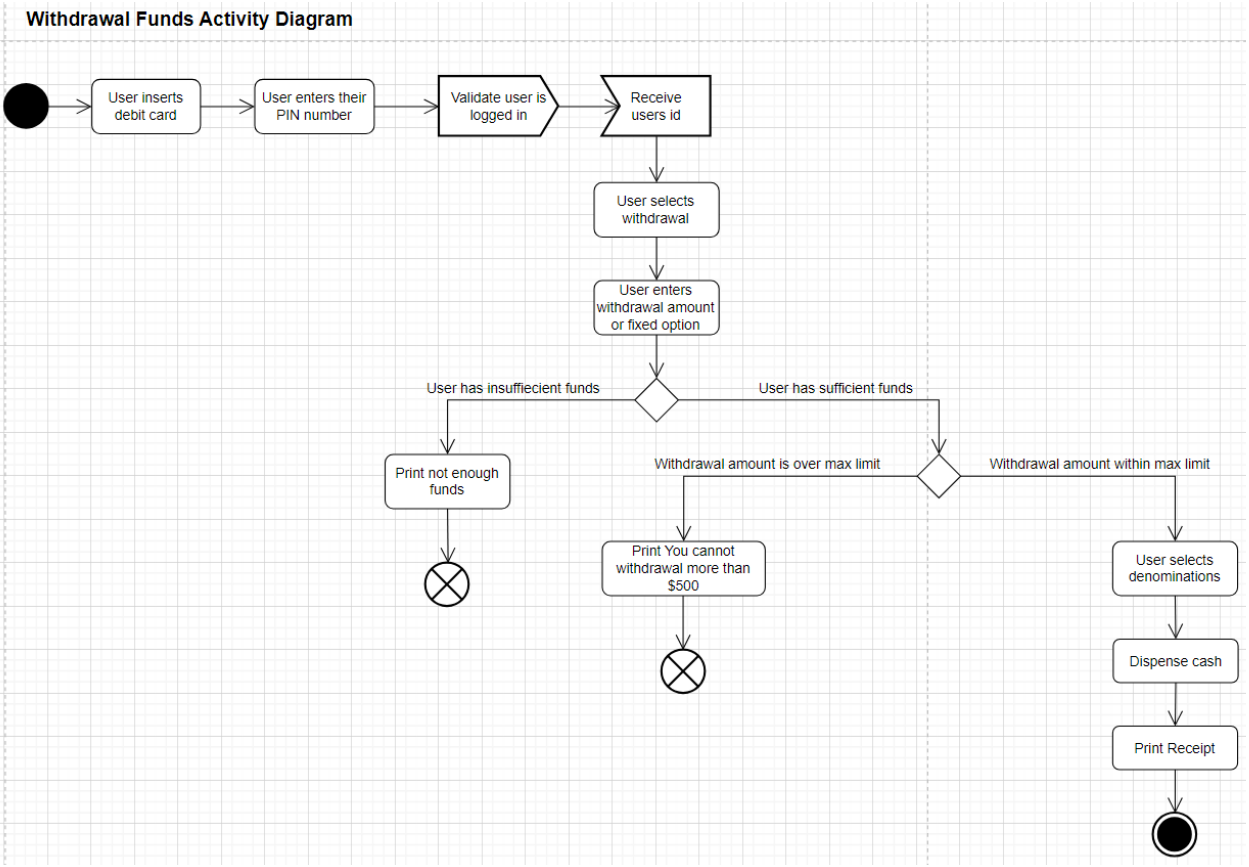
Withdraw Funds

States

- Initial State: User begins the withdrawal funds process.
- Final State: 1. The user receives confirmation ID for the withdrawal transaction, cash is dispensed to the user in the denominations they chose, and the user has the option to select either “print” or “no receipt” for their transaction. 2. The withdrawal is denied because the withdrawal exceeds the daily limit. 3. The withdrawal is denied because there are insufficient funds.

Actions

The user begins by inserting their debit card into the ATM. The user enters their PIN number, and the system validates the user. The user selects withdrawal and either selects from the fixed options or enter a custom amount. The system approves/denies the withdrawal based on sufficient/insufficient funds, then the system approves/denies based on if the daily withdrawal limit has been exceeded or not. The user selects the denominations they prefer, and the cash is dispensed. The user chooses to print or refuse a receipt.


Withdrawal Funds Activity Diagram
User inserts debit card
User enters their PIN number
Validate user is logged in
Receive users id
User selects withdrawal
User enters withdrawal amount or fixed option
User has insuffiecient funds
User has sufficient funds
Print not enough funds
Withdrawal amount is over max limit
Withdrawal amount within max limit
Print You cannot withdrawal more than $500
User selects denominations
Dispense cash
Print Receipt

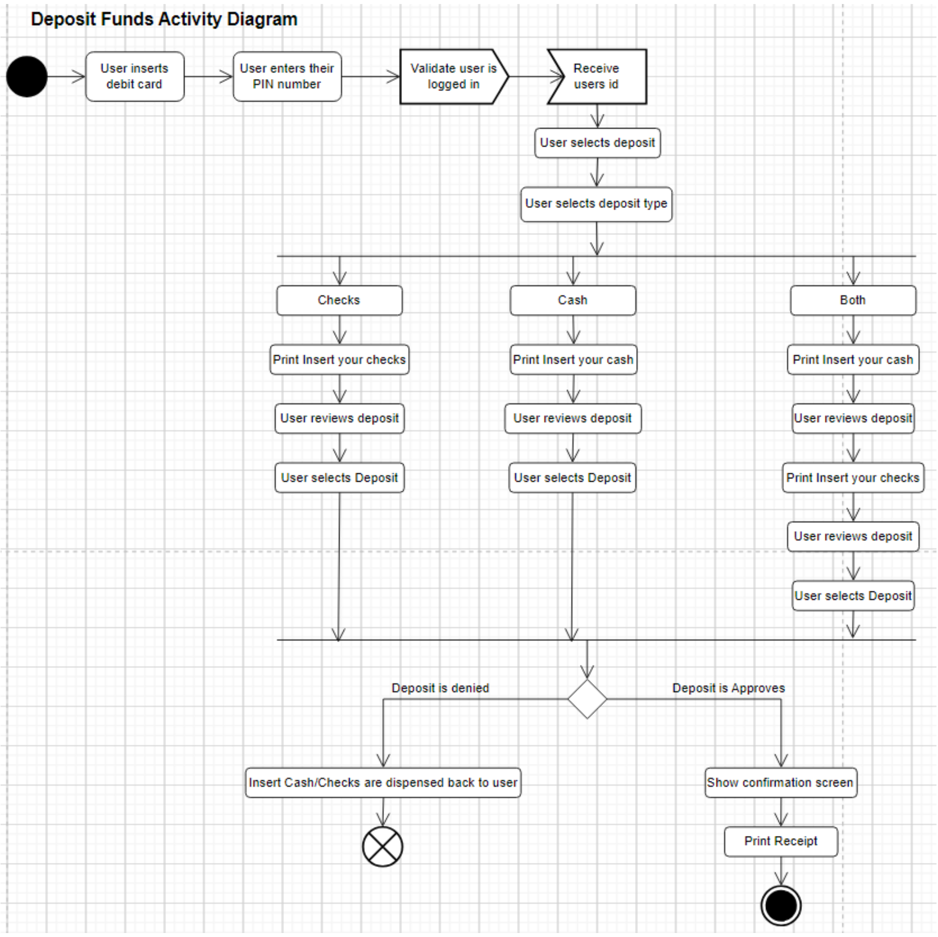
Deposit Funds

State

- Initial State: User begins the deposit funds process.
- Final State: 1. The user receives a confirmation ID for the deposit they made, and the user has the option to select either “print” or “no receipt” for their transaction. 2. The deposit is denied.

Actions

The user begins by inserting their debit card into the ATM. The user enters their PIN number, and the system validates the user. The user selects deposit and the options of cash, checks, or both. The user inserts cash/checks into the ATM. If the user is inserting both, cash will be first and checks will be done after all cash is deposited. The user selects deposit once all cash/checks have been inserted and the account to deposit into. The system approves/denies the deposit. The system records the transaction. The user chooses to print or refuse a receipt.



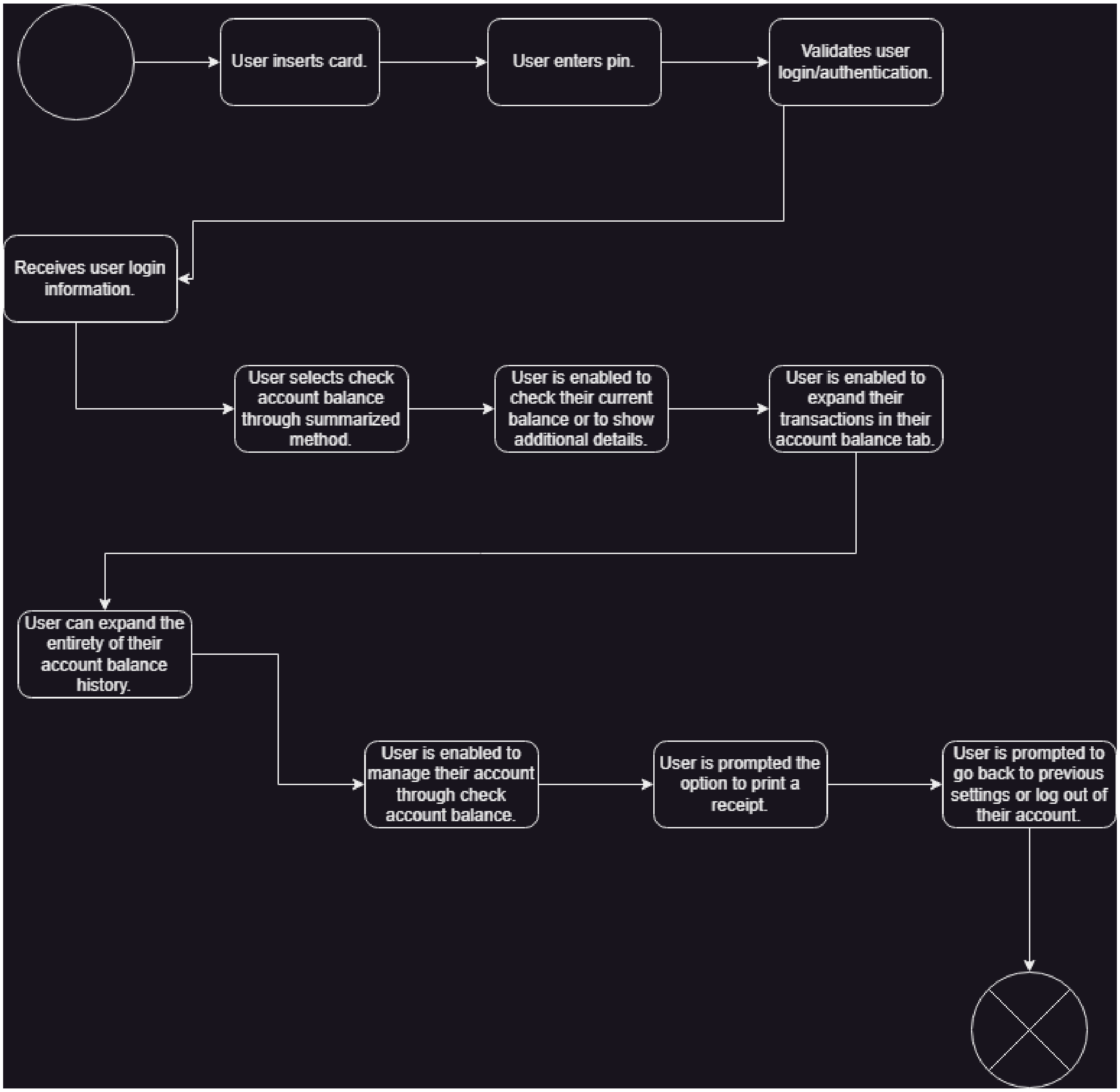
Check Account Balance

States

- Initial Sate: The user begins with inserting their debit card to obtain their checking account balance.
- Final State: The user receives the requested information regarding their checking account and is offered additional account management details.

Actions

The user begins by inserting their debit card and is prompted for their PIN. The system initiates validation of the user's information and proceeds to the main menu where the user inputs to check their checking account balance information. The system displays the information and offers users additional account management tools and resources.


User inserts card.
User enters pin.
Validates user login/authentication.
Receives user login information.
User selects check account balance through summarized method.
User is enabled to check their current balance or to show additional details.
User is enabled to expand their transactions in their account balance tab.
User can expand the entirety of their account balance history.
User is enabled to manage their account through check account balance.
User is prompted the option to print a receipt.
User is prompted to go back to previous settings or log out of their account.

Sequence Diagrams

Deposit Funds

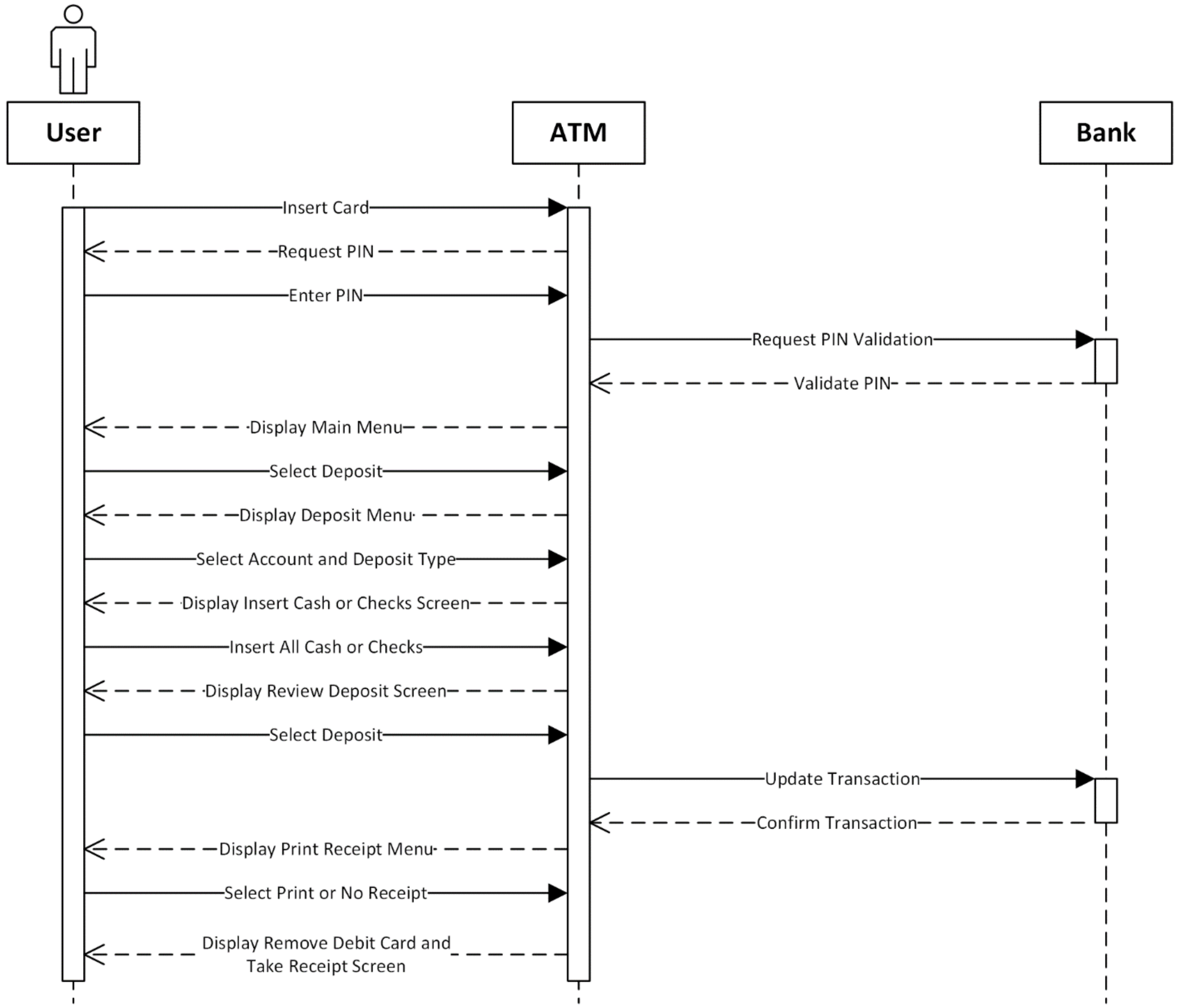
Actor: User

Objects: ATM, Bank

1. Customer inserts debit card, PIN number, account to deposit into, and all other information.
2. ATM receives debit card, PIN verification, and all cash/checks.
3. If deposit is approved, the customer receives a confirmation ID for the transaction and the option to print a receipt.

4. If deposit is denied, the ATM dispenses back all the cash/checks to the user and an error message or the option to try again.

Deposit Funds


User
ATM
Bank
Insert Card
Request PIN
Enter PIN
Request PIN Validation
Validate PIN
Display Main Menu
Select Deposit
Display Deposit Menu
Select Account and Deposit Type
Display Insert Cash or Checks Screen
Insert All Cash or Checks
Display Review Deposit Screen
Select Deposit
Update Transaction
Confirm Transaction
Display Print Receipt Menu
Select Print or No Receipt
Display Remove Debit Card and Take Receipt Screen

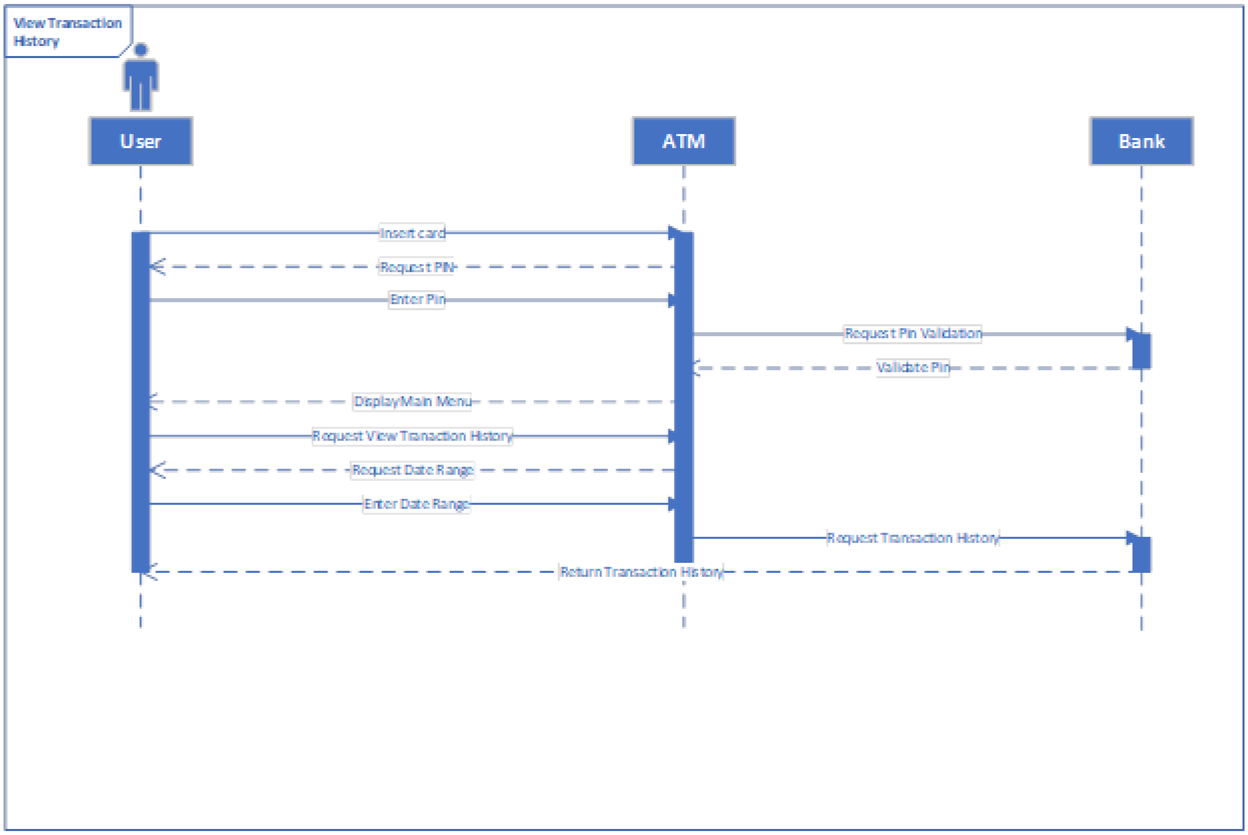
View Transaction History

Actor: User

Objects: ATM, Bank

1. User inserts their card.
2. The ATM requests their PIN.
3. Users enter their PIN.
4. The ATM requests that the Bank validate the entered PIN.
5. The Bank Validates the PIN.

6. The ATM logs the user in and displays the main menu.
7. User selects Request View Transaction History.
8. The ATM requests a date range.
9. User provides a date range.
10. ATM requests transaction history from the bank.
11. Bank returns the transaction history.



Check Account Balance Sequence Diagram.

Check Account Balance

Actor: User

Objects: ATM, Bank, and Exit Panel

Steps of check account balance interaction:

1: The customer inserts the card into the ATM.

2: The system requests verification of the user's pin to access their account information.

3: The system verifies the user's information.

4: The system proceeds to the main menu screen.

5: The user prompts arise of the check account balance details.

6: The system loads the initial account balance screen.

7: The user inputs transaction details of their account.

8: The system loads the entirety of transactions completed onto this account.

9: The user inputs the back function.

10: The system returns to the previous screen.

11: The user inputs manage account details.

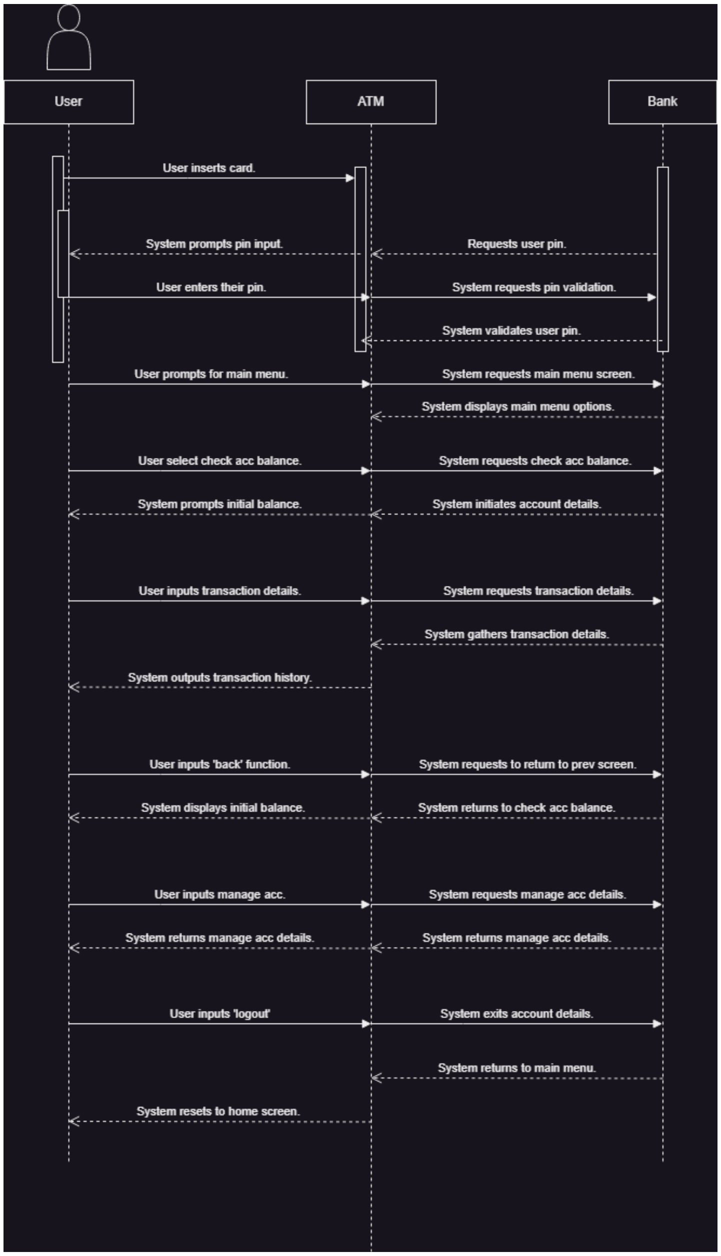
12: The system loads the user's account details that can be managed.

13: The user inputs the back function.

14: The system returns to the previous screen.

15: The user inputs the logout function.

16: The system logs out the user.


User
ATM
Bank
User inserts card.
System prompts pin input.
Requests user pin.
User enters their pin.
System requests pin validation.
System validates user pin.
User prompts for main menu.
System requests main menu screen.
System displays main menu options.
User select check acc balance.
System requests check acc balance.
System prompts initial balance.
System initiates account details.
User inputs transaction details.
System requests transaction details.
System gathers transaction details.
System outputs transaction history.
User inputs 'back' function.
System requests to return to prev screen.
System displays initial balance.
System returns to check acc balance.
User inputs manage acc.
System requests manage acc details.
System returns manage acc details.
System returns manage acc details.
User inputs 'logout'
System exits account details.
System returns to main menu.
System resets to home screen.

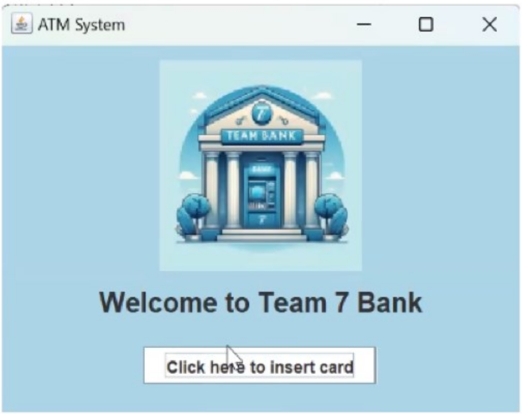
User Interface Specifications

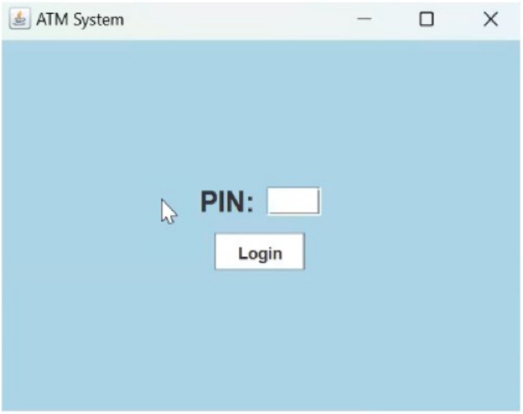
Use Case: Login UI (User Interface).

The ATM prompts the user to enter their debit card into the machine. The ATM then prompts the customer to enter their four-digit PIN number to authenticate their bank account. After the user authenticates their bank account they can proceed to the main menu where they will find a plethora of different options.

The 'Login' user interface is designed to be user-friendly while maintaining a modernized look. This screen and its steps are self-explanatory. The user will proceed like normal with entering their credentials. If the user makes a mistake and inserts the wrong card or simply does not need to utilize this ATM further, they can click the 'back' or 'x' button. The back key can also erase undesired inputs. The user may hit the accessibility option if they find themselves having complications.

User Case Login UI navigation path: Login > PIN Req > PIN Authentication > Main Menu

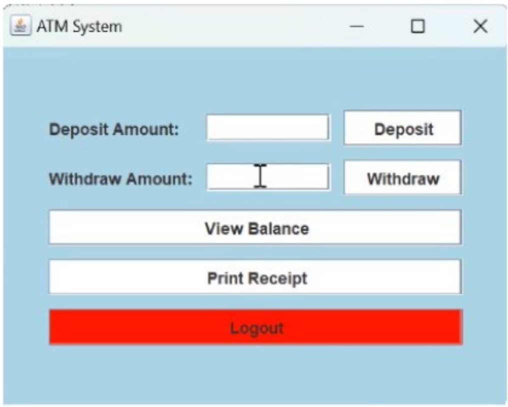


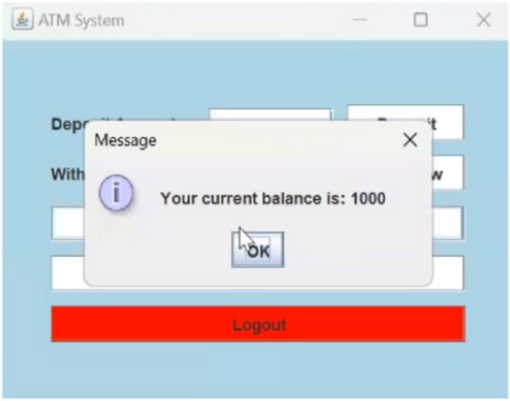


Use Case: View Balance

The user simply clicks on the “View Balance” button, then a display message appears showing their current balance.

Use Case View Balance navigation path: View Balance > Display Message



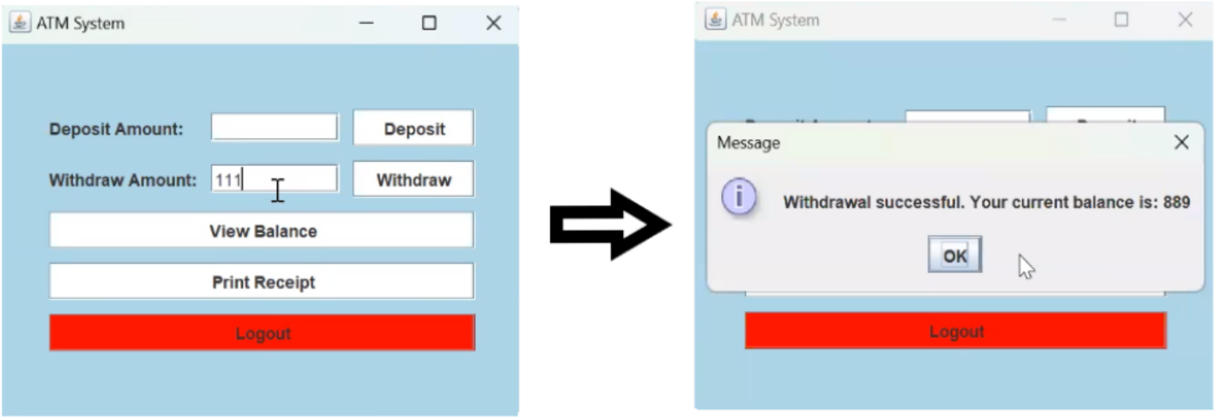


Your current balance is: 1000

Use Case: Withdraw funds from the ATM.

The user enters the amount they would like to withdraw from their account. If the amount they enter is equal to or less than their balance, then they can withdraw funds, if not they will get a message saying "insufficient funds".

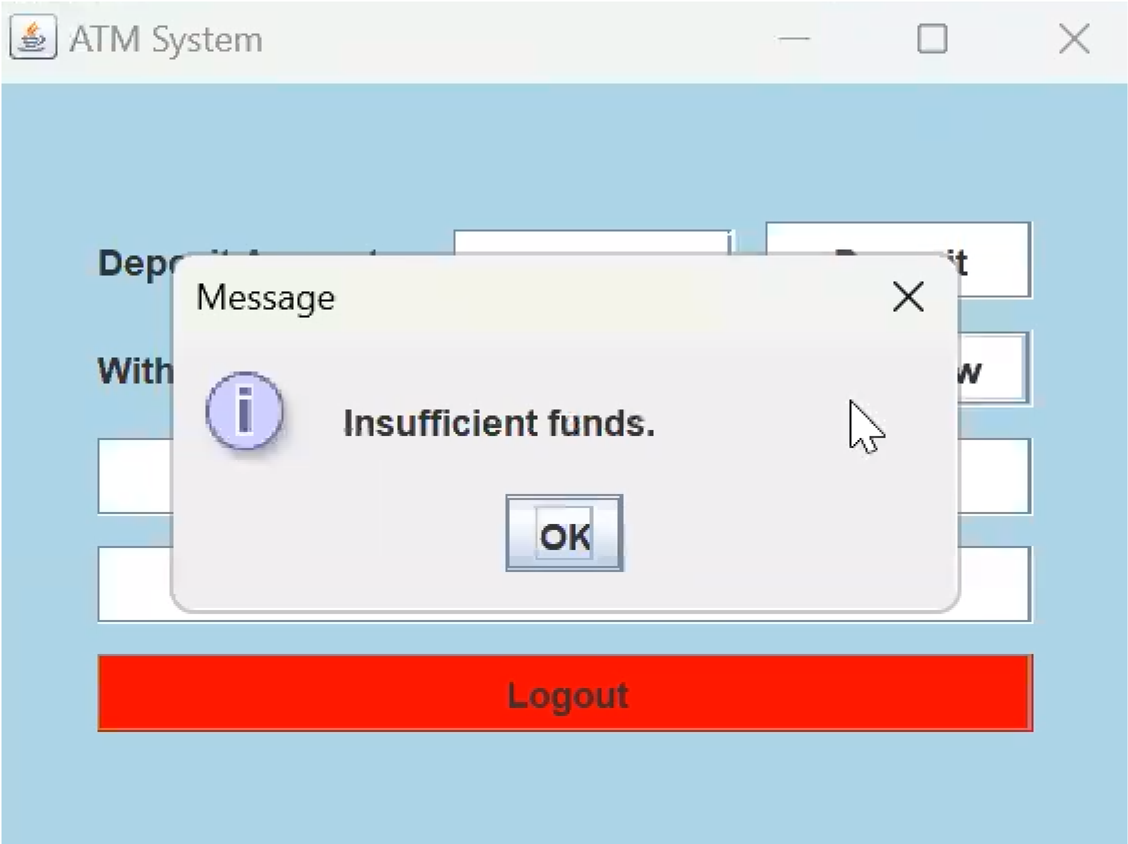
Use Case Withdraw Funds navigation path: Type amount in "withdraw amount" field > click "withdraw" button.



Deposit Amount:

Withdraw Amount: 111

Withdrawal successful. Your current balance is: 889

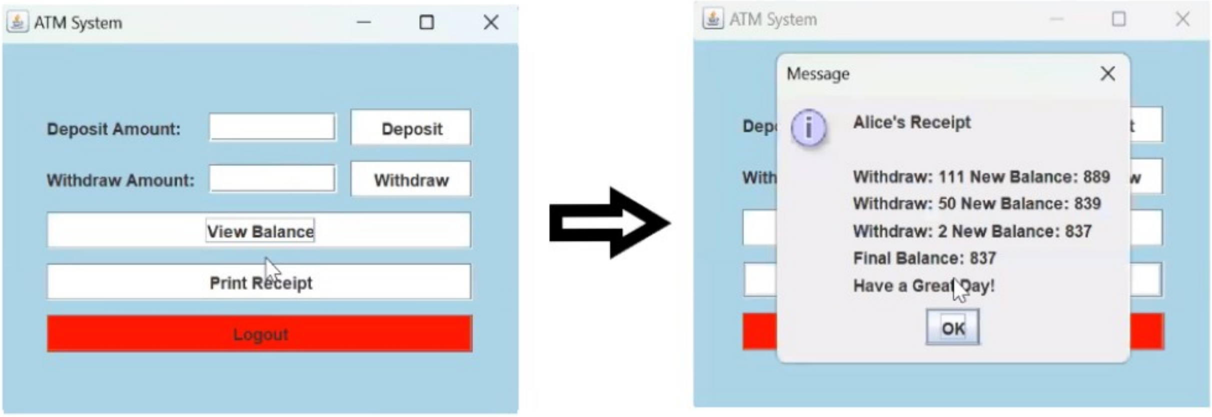


Insufficient funds.

Use Case: Print Receipt from the ATM.

The user simply clicks the “Print Receipt” button, and a message display appears showing their transactions during that login.

Use Case Print Receipt navigation path: Click “Print Receipt” button > Display Message



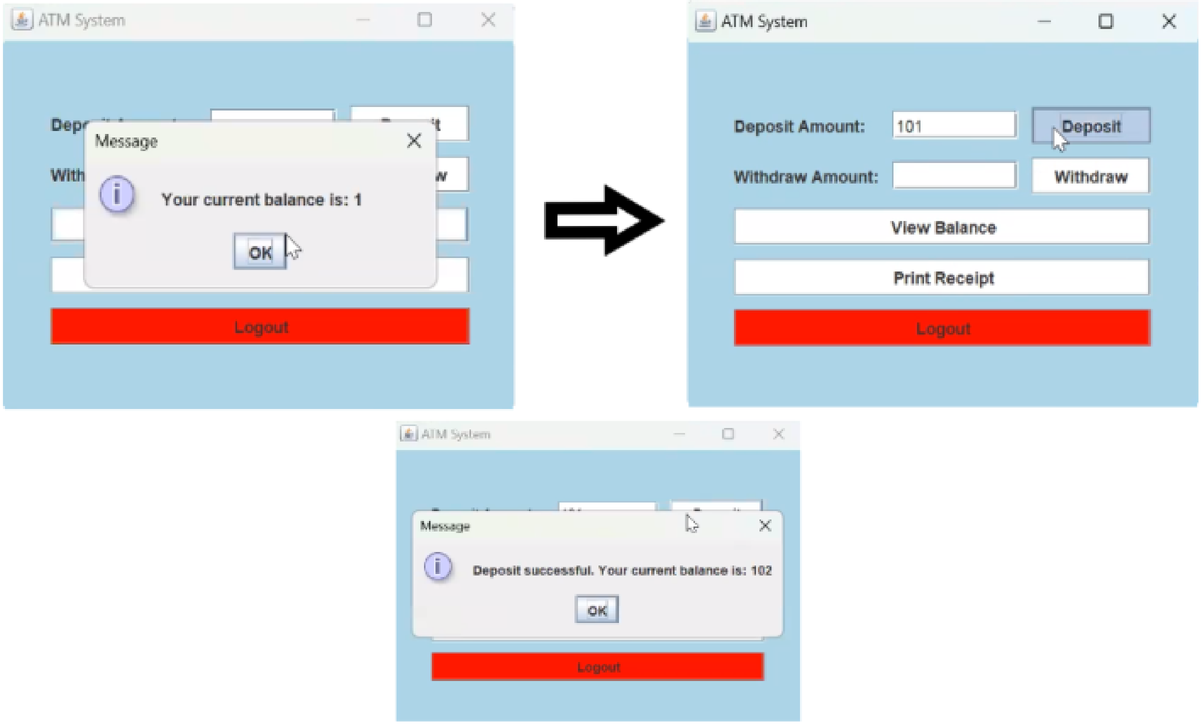
Alice's Receipt

Withdraw: 111 New Balance: 889
Withdraw: 50 New Balance: 839
Withdraw: 2 New Balance: 837
Final Balance: 837
Have a Great Day!

Use Case: Deposit Funds from the ATM.

The user enters the amount they want to deposit and clicks the “deposit” button.

Use Case deposit funds navigation path: Enter amount in deposit field > click “deposit” button > display message.



Your current balance is: 1

Deposit Amount: 101

Withdraw Amount:

Deposit successful. Your current balance is: 102

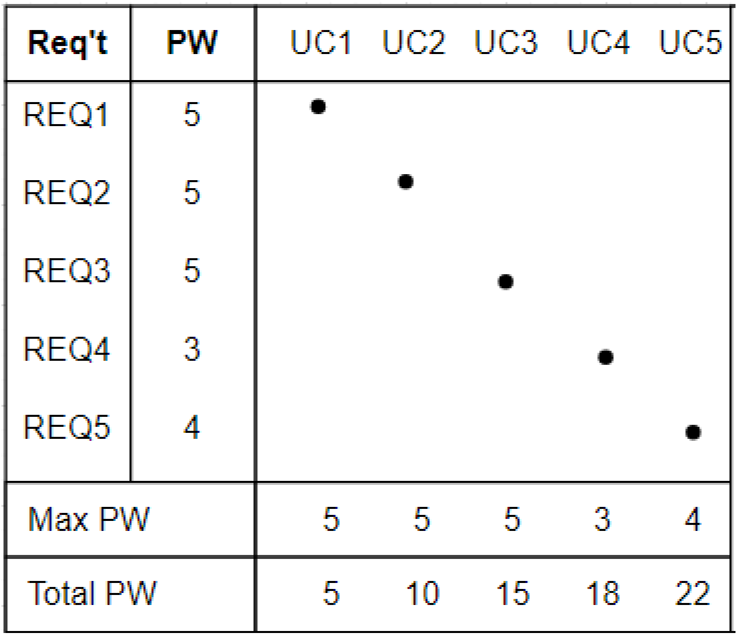
Use Case	Navigation	Clicks	Keystrokes
Login	Authenticates the user and signs into their bank accounts. Prompts input of four-digit PIN.	12	25 (max)
Check account balance	Check account balance	1	0
Withdraw Funds	Withdraw, Enter Amount, Get Funds	1	3+
Deposit Funds	Deposit Cash	1	3+
Print Receipt	View Balances	1	0

Traceability Matrix

No.	**Priority Weight. 1-5 (1L-5H).**	**Description**
REQ1	5	Users have accessibility through their PIN number to login/logout/access the system.
REQ2	5	Users can deposit cash or checks to an account linked to their debit card via PIN in the system.
REQ3	5	Users can withdraw cash from an account linked to their debit card via PIN in the system.
REQ4	3	Users can choose to print a receipt with their account balance and transaction summary on it.
REQ5	4	User can manually log out of the system by tapping a button

Use Cases

No.	Description.
UC1	Use case one enables user authentication and accessibility via PIN (digits) into the ATM system.
UC2	Use case two enables the user to deposit cash or checks via debit card into the ATM.
UC3	Use case three enables the user to withdraw electronic or cash currency via debit card from the ATM.
UC4	Use case eight enables' users to print receipts of their transactions and balances.
UC5	Use case twelve signs the user out of the ATM system when they tap the "sign out" button.



Req't	PW	UC1	UC2	UC3	UC4	UC5
REQ1	5	●				
REQ2	5		●			
REQ3	5			●		
REQ4	3				●	
REQ5	4					●
Max PW		5	5	5	3	4
Total PW		5	10	15	18	22

System Architecture and System Design

System Overview: The ATM system is designed using Java for the backend logic and MySQL for persistent data storage. The system provides basic functionalities such as account balance inquiry, cash withdrawal, cash deposit, and account transfer.

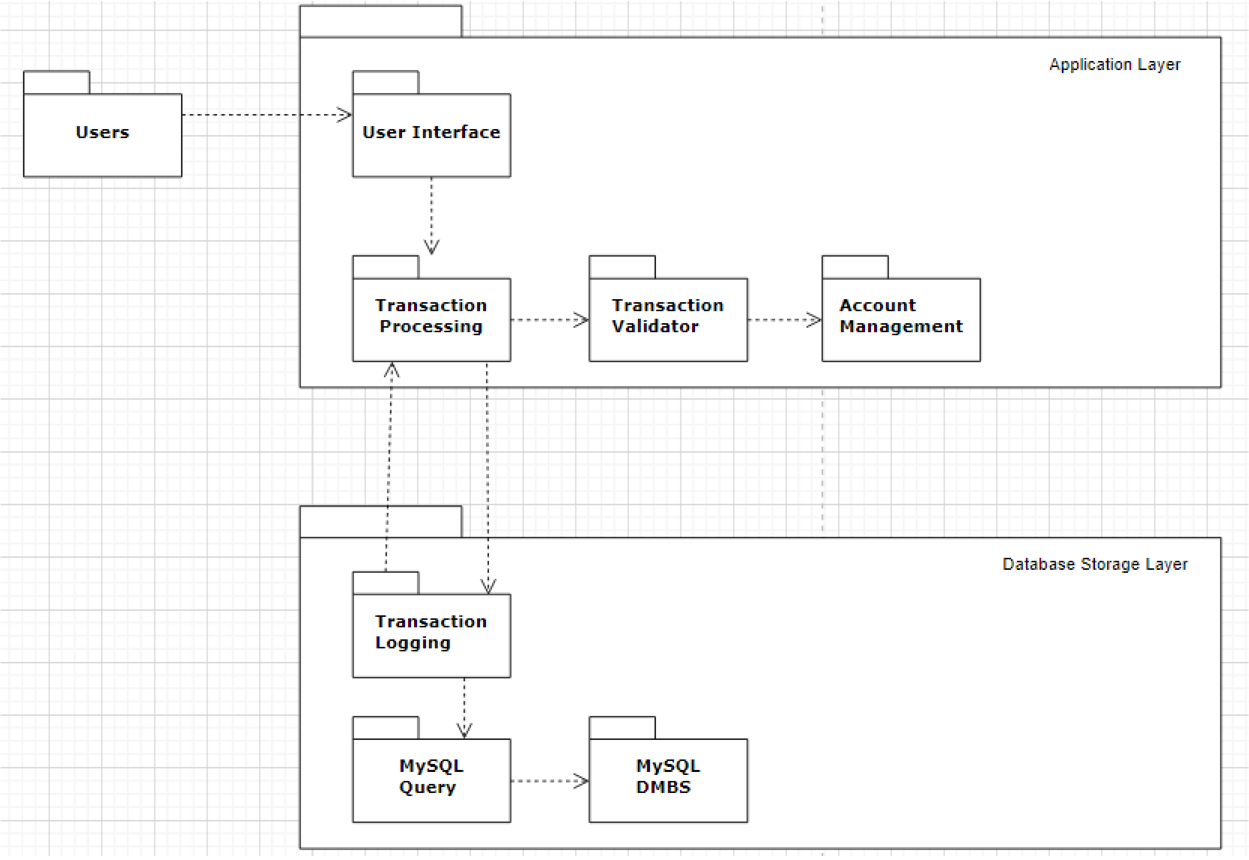
Architectural Styles: The architectural style adopted for this system is the client-server architecture. The client (ATM machine) interacts with the server (backend system) to perform various transactions. The server handles the business logic and data storage, while the client provides the interface for users to interact with the system.

Identifying Subsystems:

- **Client (ATM Machine):** Responsible for user interaction and sending requests to the server.
- **Server:** Handles business logic, database interactions, and responds to client requests.
-

Mapping Subsystems to Hardware: The system will primarily run on a single computer where both the client (ATM machine) and the server (backend system) will reside. Therefore, there is no need for subsystems to run on different machines.

UML Package Diagram: The below shows the ATM Transaction subsystem. The Users package interacts with the User Interface package. The user uses buttons available to select the transaction they would like to make then each transaction is processed, validated, and recorded into the MySQL database.



Persistent Data Storage: The system requires saving data beyond a single execution. Persistent objects include user account information, transaction history, and system configuration. We've selected a relational database (MySQL) as the storage management strategy for its robustness, scalability, and ACID compliance.

Network Protocol: As the system operates on a single machine, network protocols are not applicable.

Global Control Flow: The system is event-driven, where users initiate actions based on their requirements. There are no specific timers or real-time constraints. Concurrency is utilized within the server subsystem to handle multiple client requests concurrently. Synchronization mechanisms are implemented to ensure data consistency in multi-threaded environments.

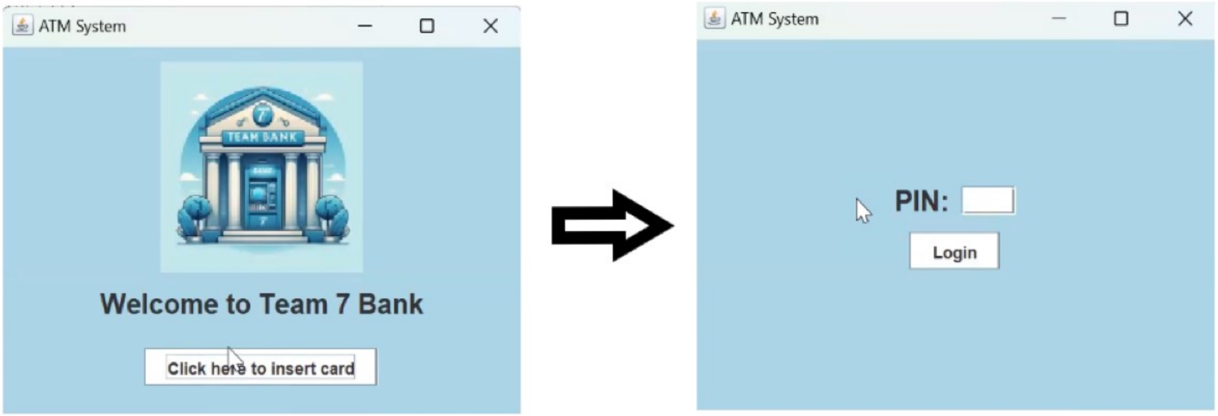
Hardware Requirements:

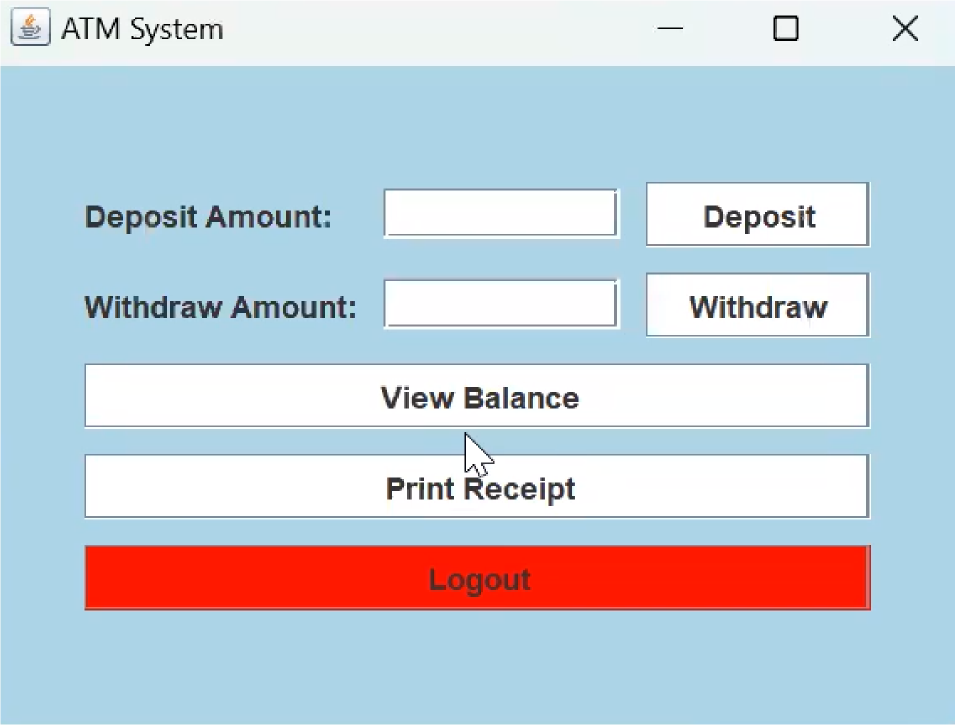
- **Screen Display:** Minimum resolution of 800x600 pixels, supporting basic graphics rendering.
- **Input Devices:** Standard keyboard and ATM keypad for user input.
- **Storage:** Sufficient disk space to accommodate database files and system execution.
- **Memory:** Adequate RAM to support concurrent user sessions and database operations.
- **Processor:** Multi-core processor for efficient handling of concurrent requests.
- **Communication Network:** While the system primarily operates offline, it may require network connectivity for periodic updates or maintenance tasks.

Overall, the system is designed to provide a seamless and secure banking experience for users while ensuring scalability and maintainability for future enhancements.

User Interface Design and Implementation

The design of our system and interfaces can be described in one word, simplicity, to make sure the user can accomplish each task on their own without any assistance. The initial idle screen will prompt the user to enter their debt card (or demo will be a click because we cannot physically show that). Then they enter their PIN number to be taken to the main menu to do the task they wish (deposit, withdraw, print receipt, view balance, and logout).





Unit Testing Test Cases:

Test Case 1: Successful Debit Card Insertion
Description: Verify that a user can be successfully registered with valid input data (PIN).
Input: 4-digit PIN.
Expected Outcome: User should be able to access their account if successful.

Test Case 2: Invalid PIN Format
Description: Verify that the system correctly handles invalid PIN formats during user registration.
Input: PIN that is not a 4-digit number.
Expected Outcome: System should display an error message indicating that the PIN must be a 4-digit number.

Test Case 4: Duplicate Username
Description: Verify that the system prevents registration with a username that already exists.
Input: Username that already exists in the system.
Expected Outcome: System should display an error message indicating that the username is already registered.

Test Case 5: Successful Deposit
Description: Verify that a user can successfully deposit funds into their account.
Input: deposit amount.
Expected Outcome: System should update the user's balance with the deposited amount and display a success message.

Test Case 6: Insufficient Funds for Withdrawal
Description: Verify that the system prevents withdrawal of funds exceeding the available balance.
Input: Withdrawal amount greater than the current balance.
Expected Outcome: System should display an error message indicating insufficient funds.

Test Case 7: Successful Withdrawal
Description: Verify that a user can successfully withdraw funds from their account.
Input: Valid username and withdrawal amount.
Expected Outcome: System should deduct the withdrawal amount from the user's balance and display a success message.

Test Coverage:

The unit tests cover various aspects of the ATM system, including user registration, input validation, deposit, and withdrawal functionalities. These tests ensure that the system behaves as expected under different scenarios and handles user interactions accurately.

Integration Testing Strategy:

For integration testing, the focus will be on testing the interaction between different components/modules of the ATM system. This will involve simulating user interactions and verifying that data is passed correctly between the registration and transaction panels, as well as ensuring that user actions trigger the appropriate functionalities.

Testing Plans for Algorithms, Non-functional Requirements, and UI Requirements:

Algorithms: Algorithms used for deposit, withdrawal, and balance calculation will be tested to ensure accuracy and efficiency.

Non-functional Requirements: Performance testing will be conducted to evaluate the system's responsiveness and scalability under various load conditions.

UI Requirements: User interface testing will involve verifying that the UI elements are displayed correctly, user inputs are properly validated, and error messages are displayed as expected. Additionally, usability testing will be performed to assess the overall user experience and ease of use of the system.

Project Plan

Weeks 1-2: Define the project framework, set up the development environment, and establish the basic structure.

Weeks 3-4: Implement user authentication via PIN and basic account management features.

Weeks 5-7: Develop core functionalities such as balance inquiries, cash withdrawals, and fund transfers.

Week 8: Test and record a demo showcasing the main features for the mid-term evaluation.

Weeks 9-11: Implement transaction history tracking and enhance security features.

Weeks 12-14: Write test cases for implemented features and conduct thorough testing.

Week 15: Record a final demo presentation showcasing the complete ATM System.

References

Tilley, S., & Rosenblatt, H. (2017). Systems analysis and design (11th ed).

Tiley, S. (2020). Systems analysis and design (12th ed).

Whitten, J. & Bentley, L. (2007). System analysis and design methods (7th ed.).