SYSTEM DESIGN BY UML MODELLING

ATM Banking System

Maryam Fahmi, Samih Wadi, Sidra Musheer, Jessica Morcos, Tito Osemobor, Nicanor Obasi

Toronto Metropolitan University
CPS 406 - Section 6
Introduction to Software Engineering

CONTENTS

System Design Page 2

- Classes and Components Page 2 - 7
  - Attributes
  - Methods
- Class Diagram Page 8
- Sequence Diagrams Page 9 - 14
  - Credential Check
  - Deposit Funds
  - Withdraw Funds
  - Transfer Funds
  - Balance Inquiry
  - Pay Bills
- Statechart Diagrams Page 15 - 16
  - Account Verification State
  - Account Selection State
  - Transaction Selection State
  - Transaction Processing State
- Activity Diagrams Page 17 - 23
  - Credential Check
  - Change PIN
  - WIthdraw Amount
  - Deposit Amount
  - Customer Account Details
  - Balance Check
  - Pay Bills

System Objects Page 24

Object Design Page 24 - 25

SYSTEM DESIGN

The System Design refers to defining and modelling complex interactions among the components of the various classes that comprise a system, and being able to implement the system with proper and effective use of available resources.

Classes and Components

Having previously identified the functional requirements for each component of a class and tasks that said component is involved in, we can summarise the System Classes as follows -

CLASS	COMPONENTS
Account	1. Account number 2. Account type (chequing/savings/credit card) 3. Account balance
Customer	1. Account number 2. PIN 3. Account balance 4. Transactions 5. Name 6. Address
Card	1. Card number 2. Expiration date 3. Customer details 4. PIN
System/Bank	1. Customer information database 2. Transaction database 3. Authentication system 4. Bill management system
User Interface	1. Login 2. Transaction options
Card Reader	1. Insert card 2. Scan card 3. Validate card
Transactions	1. Withdrawals 2. Deposits 3. Transfers 4. Balance checks 5. Pay bills
Withdrawals	1. Account 2. Withdrawal amount

Deposits	1. Account 2. Deposit amount
Transfers	1. Account 2. Transfer amount 3. Source account 4. Destination account
Balance Checks	1. Account 2. Total amount
Pay Bills	1. Amount 2. Source account 3. Destination account
ATM	1. Display 2. Card Reader 3. Keypad 4. Bill Dispenser 5. Receipt printer
Keypad	
Bill Dispenser	1. $5 Bills 2. $10 Bills 3. $20 Bills
Display Screen	
Receipt Printer	1. Receipt
Bill Deposit Slot	
Bill Counter	
Bill Verifier	
Maintenance	1. Refill Machine 2. System malfunction checker

Attributes:

CLASS	COMPONENTS	FUNCTIONAL REQUIREMENT(S)
Account	1. Account number 2. Account type (chequing/savings/credit card) 3. Account balance	FR05, FR15
Customer	1. Account number 2. PIN 3. Account balance 4. Transactions 5. Name 6. Address	FR01, FR02, FR03, FR04, FR07
Card	1. Card number 2. Expiration date 3. Customer details 4. PIN	FR08
System/Bank	1. Customer information database 2. Transaction database 3. Authentication system 4. Bill management system	FR08, FR09, FR10, FR11, FR12
User Interface	1. Login 2. Transaction options	FR03, FR06, FR15, FR17
Card Reader	1. Insert card 2. Scan card 3. Validate card	FR08, FR13
Transactions	1. Withdrawals 2. Deposits 3. Transfers 4. Balance checks 5. Pay bills	FR05, FR14, FR16
Withdrawals	1. Account 2. Withdrawal amount	FR05, FR14, FR18, FR19
Deposits	1. Account 2. Deposit amount	FR05, FR18
Transfers	1. Account 2. Transfer amount 3. Source account 4. Destination account	FR05
Balance Checks	1. Account 2. Total amount	FR05, FR12

Pay Bills	1. Amount 2. Source account 3. Destination account	FR05, FR18
ATM	1. Display 2. Card Reader 3. Keypad 4. Bill Dispenser 5. Receipt printer	FR13, FR14, FR15, FR16, FR18, FR19
Keypad		FR03, FR16
Bill Dispenser	1. $5 Bills 2. $10 Bills 3. $20 Bills	FR14
Display Screen	1. Keypad 2. Message screen	FR09
Receipt Printer	1. Receipt	FR06, FR18
Bill Deposit Slot		FR19
Bill Counter		FR16
Bill Verifier		FR16
Maintenance	1. Refill Machine 2. System malfunction checker	FR10, FR11, FR20, FR21, FR22, FR23

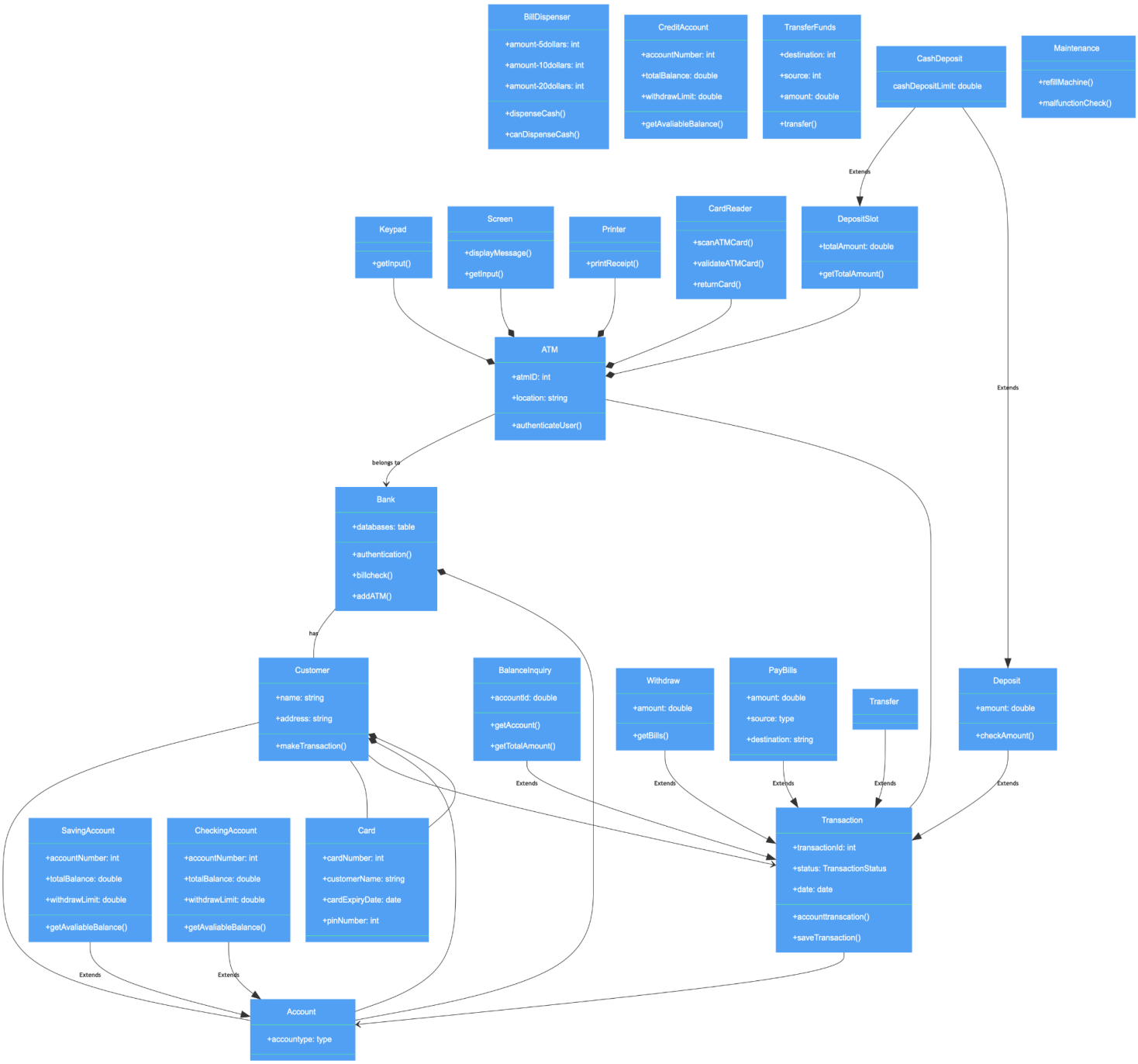
Methods:

CLASS	COMPONENTS	TASK(S)
Account	1. Account number 2. Account type (chequing/savings/credit card) 3. Account balance	1. Store account details 2. Store account type for different transactions 3. Provide account balance upon request
Customer	1. Account number 2. PIN 3. Account balance 4. Transactions 5. Name 6. Address	1. Store account details 2. Validate customer 3. Provide account balance upon request 4. Store customer details 5. Store customer details
Card	1. Card number 2. Expiration date 3. Customer details 4. PIN	1. Maintain account security 2. Validate card 3. Verify card 4. Verify card
System/Bank	1. Customer information database 2. Transaction database 3. Authentication system 4. Bill management system	1. Store customer details 2. Maintain transaction history 3. Maintain account security 4. Manage bills in machine
User Interface	1. Login 2. Transaction options	1. Security check 2. Provide available operations
Card Reader	1. Insert card 2. Scan card 3. Validate card	Verify and validate card for security
Transactions	1. Withdrawals 2. Deposits 3. Transfers 4. Balance checks 5. Pay bills	1. Allow withdrawals from available accounts 2. Allow deposits to available accounts 3. Allow transfers to available accounts 4. Provide available funds to customer 5. Allow paying bills to verified accounts
Withdrawals	1. Account 2. Withdrawal amount	1. Access available funds 2. Deduct requested amount from account if possible; otherwise error message
Deposits	1. Account 2. Deposit amount	1. Access available funds 2. Deposit funds into selected

		account
Transfers	1. Account 2. Transfer amount 3. Source account 4. Destination account	1. Access available funds 2. Deduct requested amount from account if possible; otherwise error message 3. Account to transfer from 4. Account to transfer into
Balance Checks	1. Account 2. Total amount	1. Access available funds
Pay Bills	1. Amount 2. Source account 3. Destination account	1. Access available funds, select amount to be paid 2. Account to transfer from 3. Account to transfer into
ATM	1. Display 2. Card Reader 3. Keypad 4. Bill Dispenser 5. Receipt printer	1. Provide transaction menu, other messages 2. Receive and scan card 3. Receive PIN from customer 4. Dispense amount requested 5. Print receipt for summary of transaction
Keypad		Receive PIN from customer
Bill Dispenser	1. $5 Bills 2. $10 Bills 3. $20 Bills	Dispensing amount requested in suitable bills
Display Screen		
Receipt Printer	1. Receipt	Prints receipt for transaction history
Bill Deposit Slot		Accepts bills from customer; checks validity of bills
Bill Counter		Counts bills provided or requested by customer
Bill Verifier		Validates bills deposited by customer and verifies their currency
Maintenance	1. Refill Machine 2. System malfunction checker	1. Refill bills in ATM when below threshold 2. Maintain functioning ATM software and hardware

Class Diagram

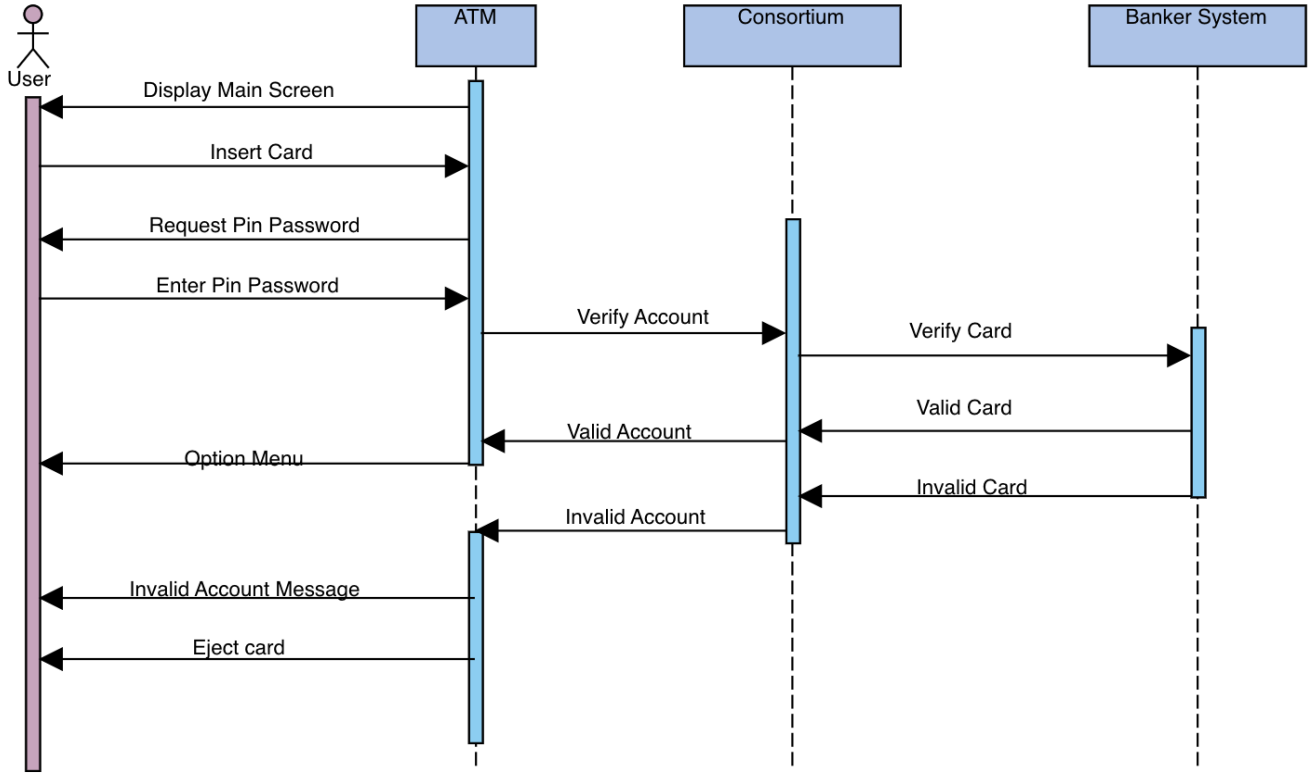
The purpose of a class diagram is to illustrate the static structure of the system. This encompasses objects, attributes and associations. It is an Object Model.



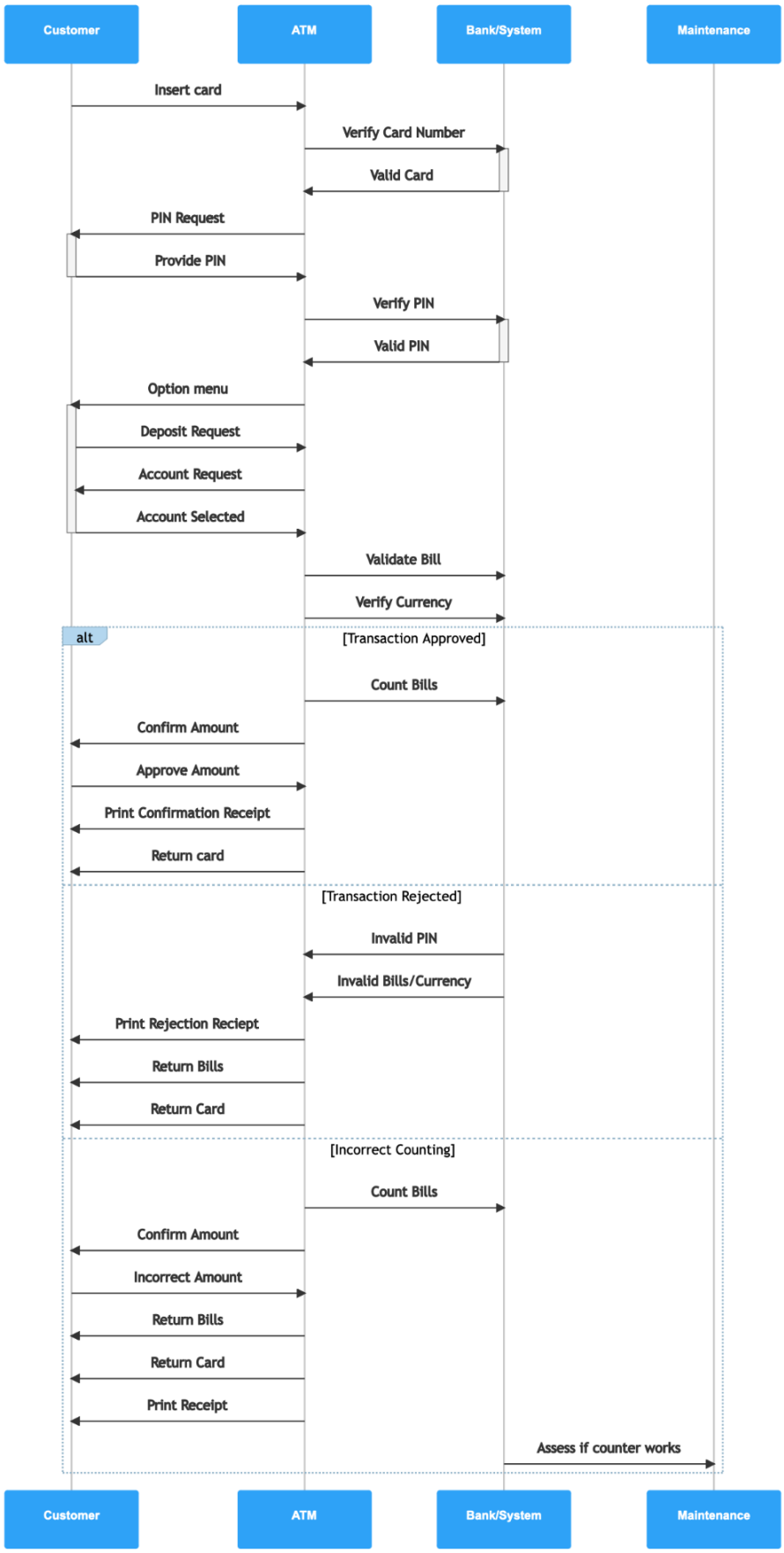
Sequence Diagrams

The purpose of a sequence diagram is to illustrate the dynamic behaviour of between the objects of the system. It represents the behaviour of a system as messages or interactions between several objects. It is a Dynamic Model.

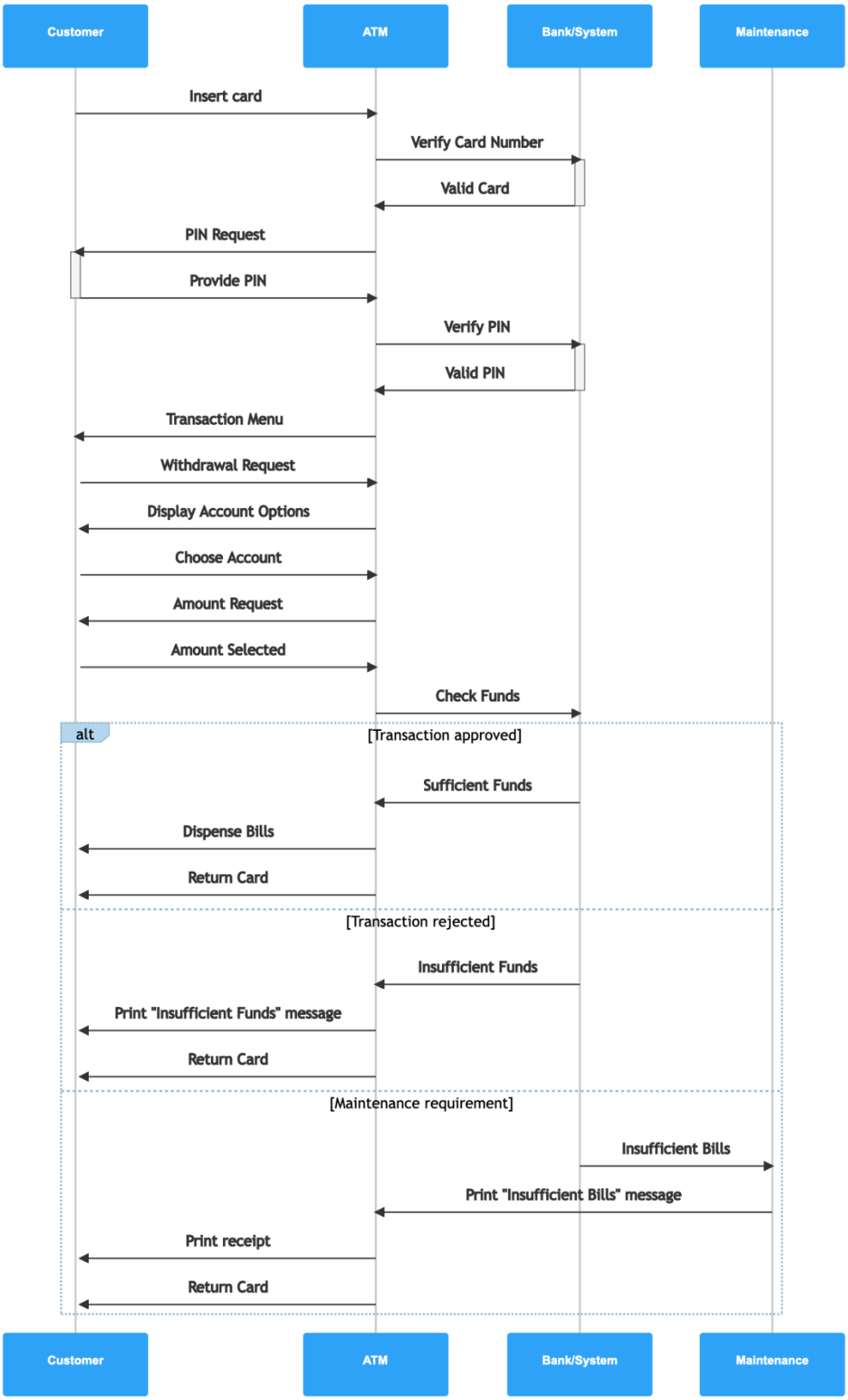
- Credential Check



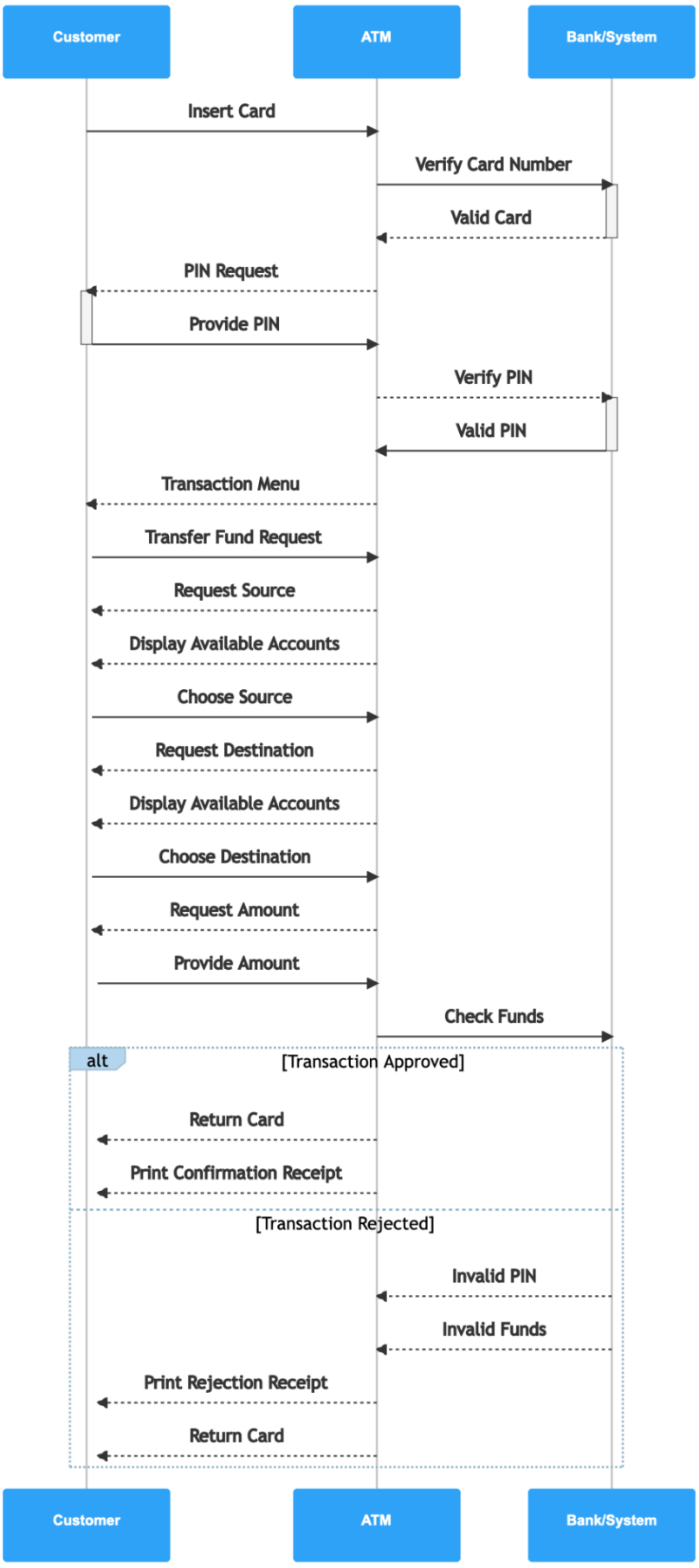
- Deposit Funds



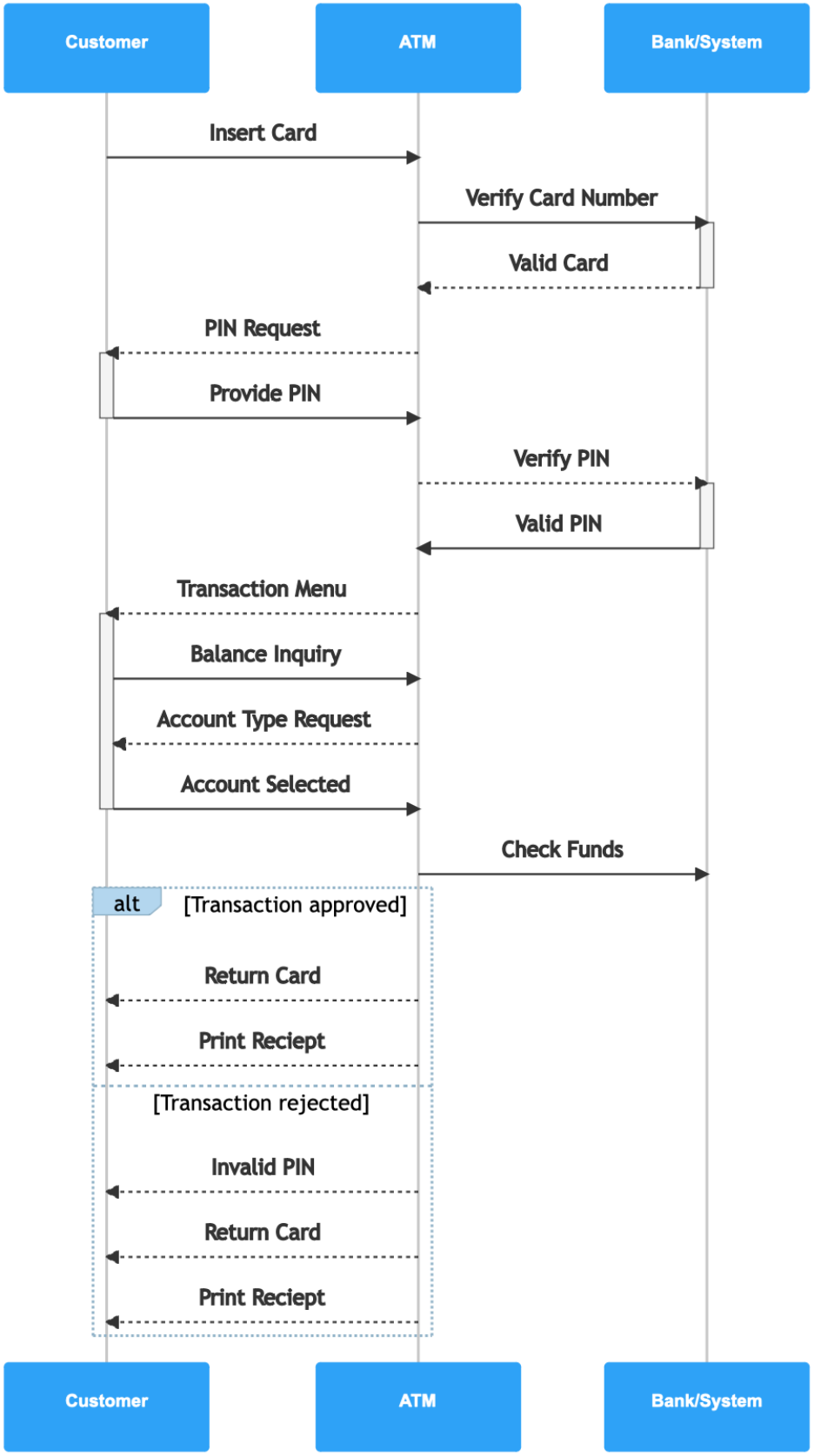
- Withdraw Funds



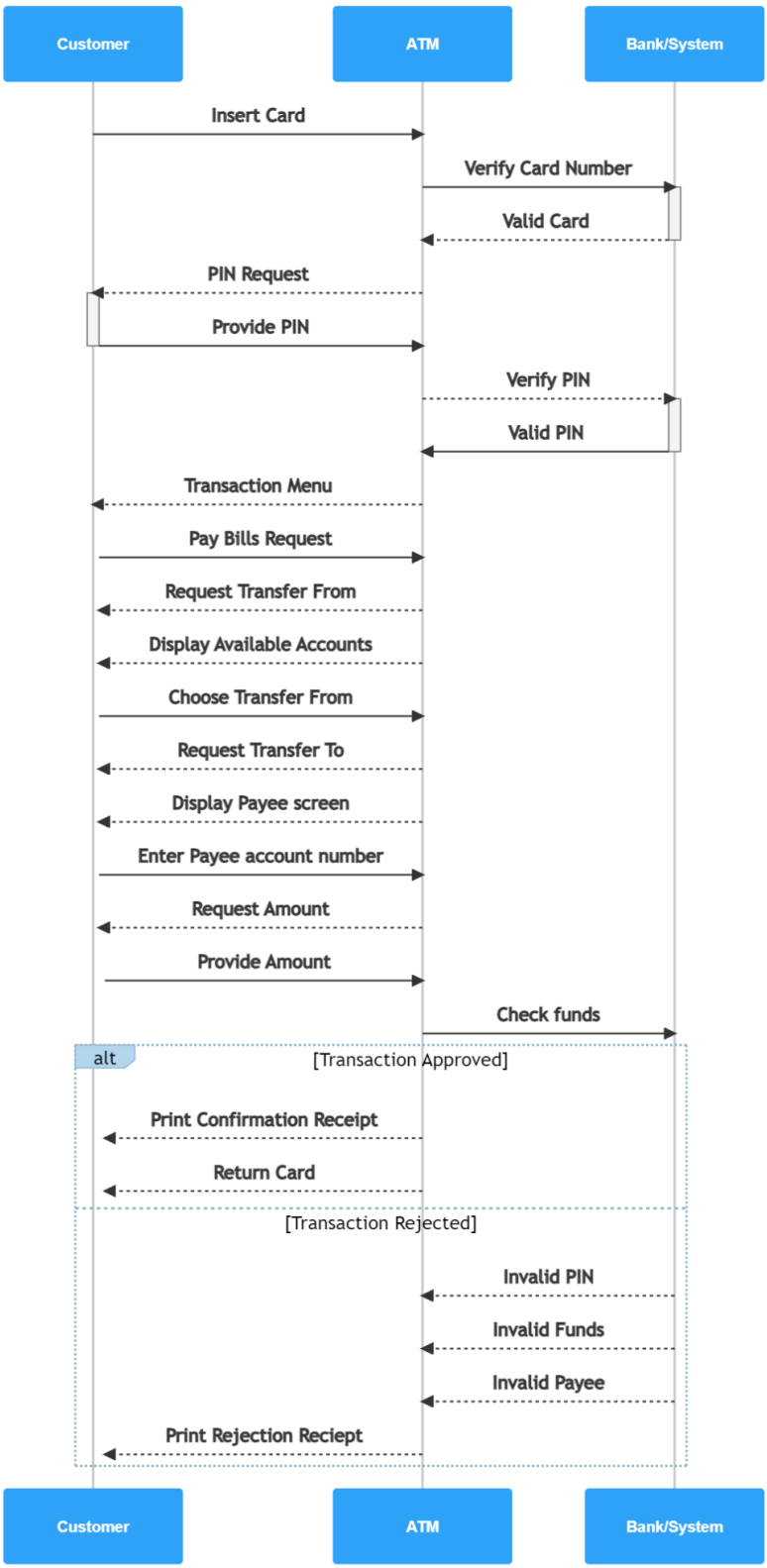
- Transfer Funds



- Balance Inquiry



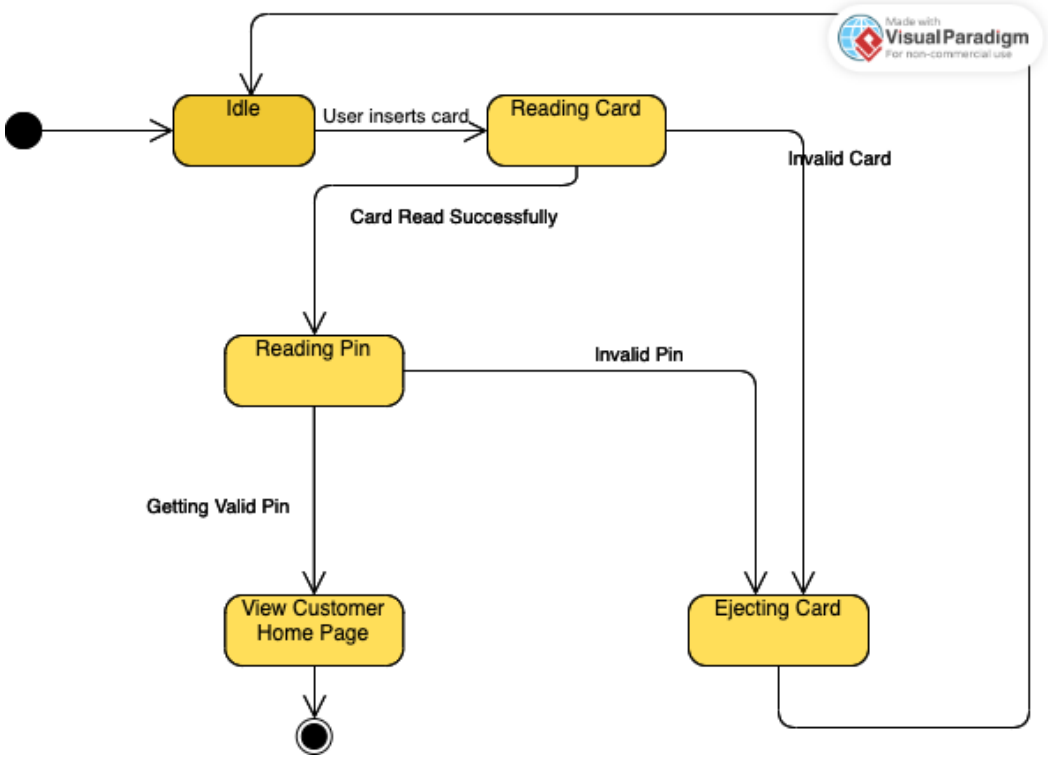
- Pay Bills


Customer
ATM
Bank/System
Insert Card
Verify Card Number
Valid Card
PIN Request
Provide PIN
Verify PIN
Valid PIN
Transaction Menu
Pay Bills Request
Request Transfer From
Display Available Accounts
Choose Transfer From
Request Transfer To
Display Payee screen
Enter Payee account number
Request Amount
Provide Amount
Check funds
alt
[Transaction Approved]
Print Confirmation Receipt
Return Card
[Transaction Rejected]
Invalid PIN
Invalid Funds
Invalid Payee
Print Rejection Reciept
Customer
ATM
Bank/System

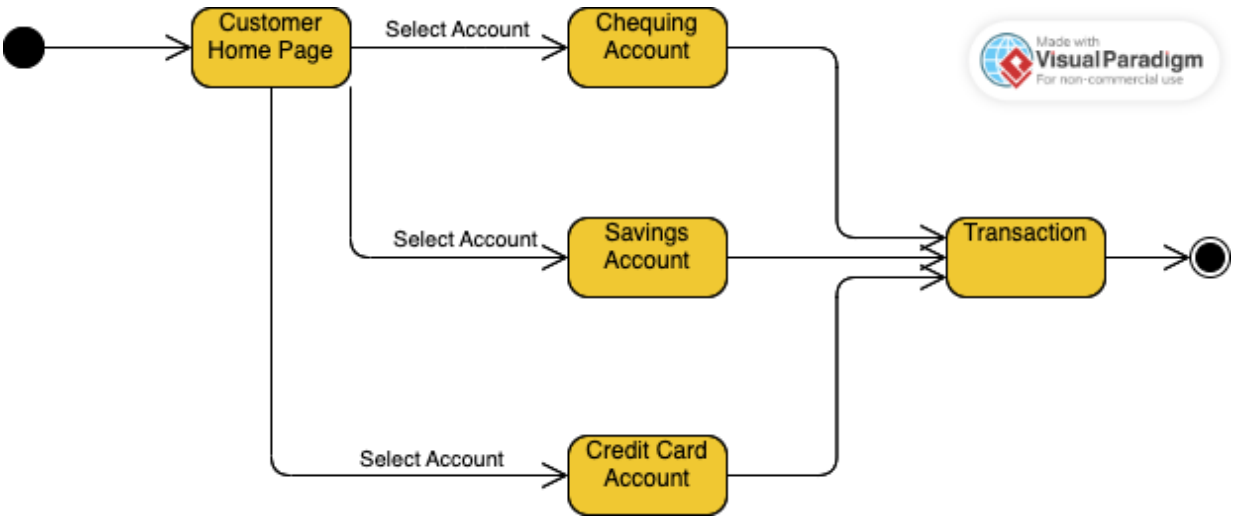
Statechart Diagrams

The purpose of a statechart diagram is to illustrate the dynamic behaviour of an individual object of the system. It is a Dynamic Model.

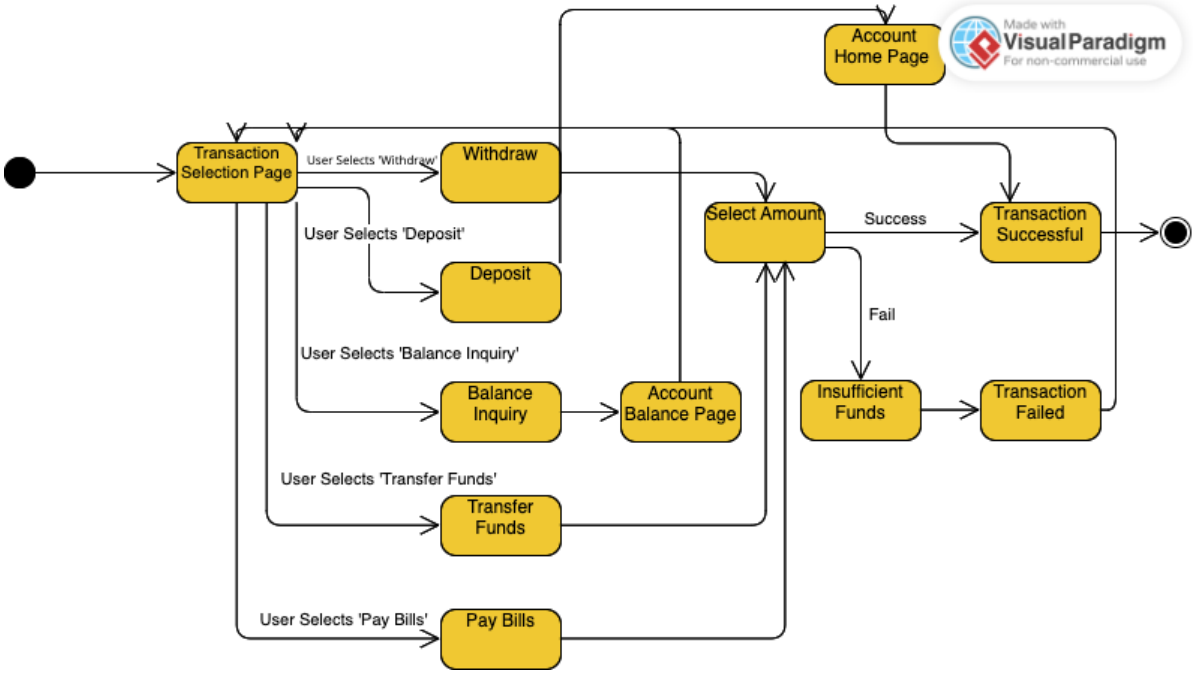
- Account Verification State



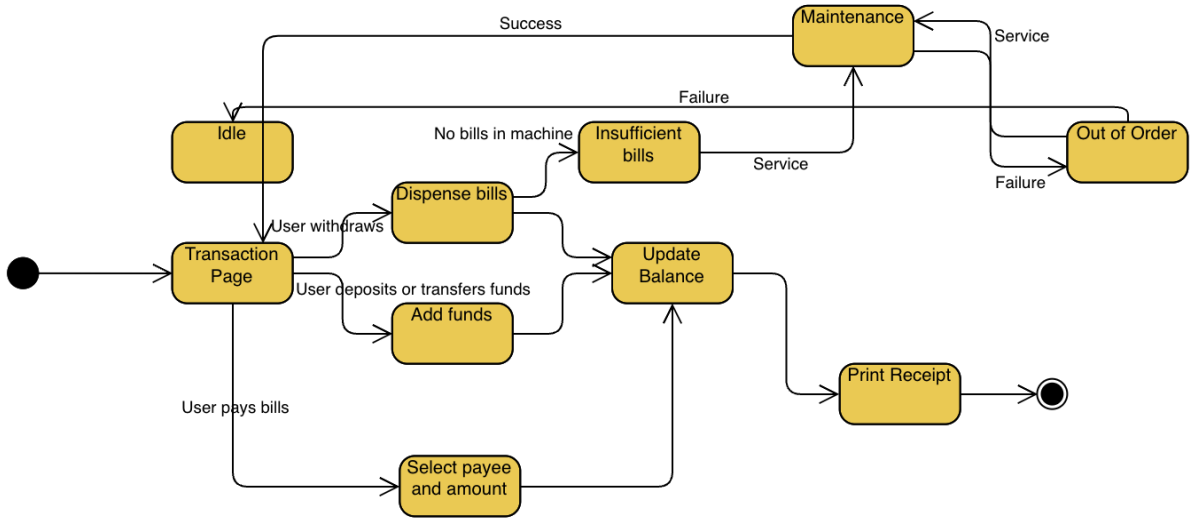
- Account Selection State



- Transaction Selection State



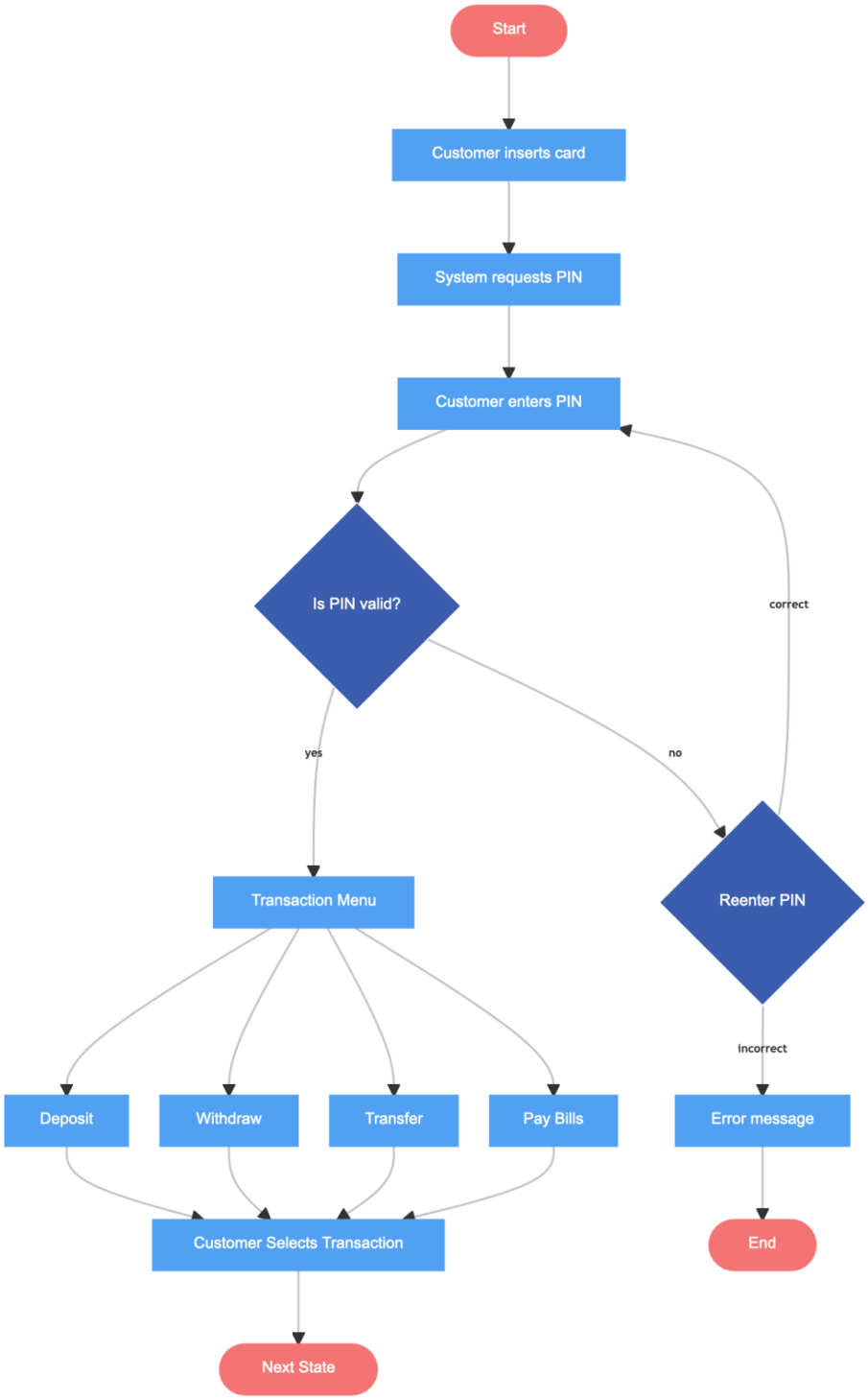
- Transaction Processing State



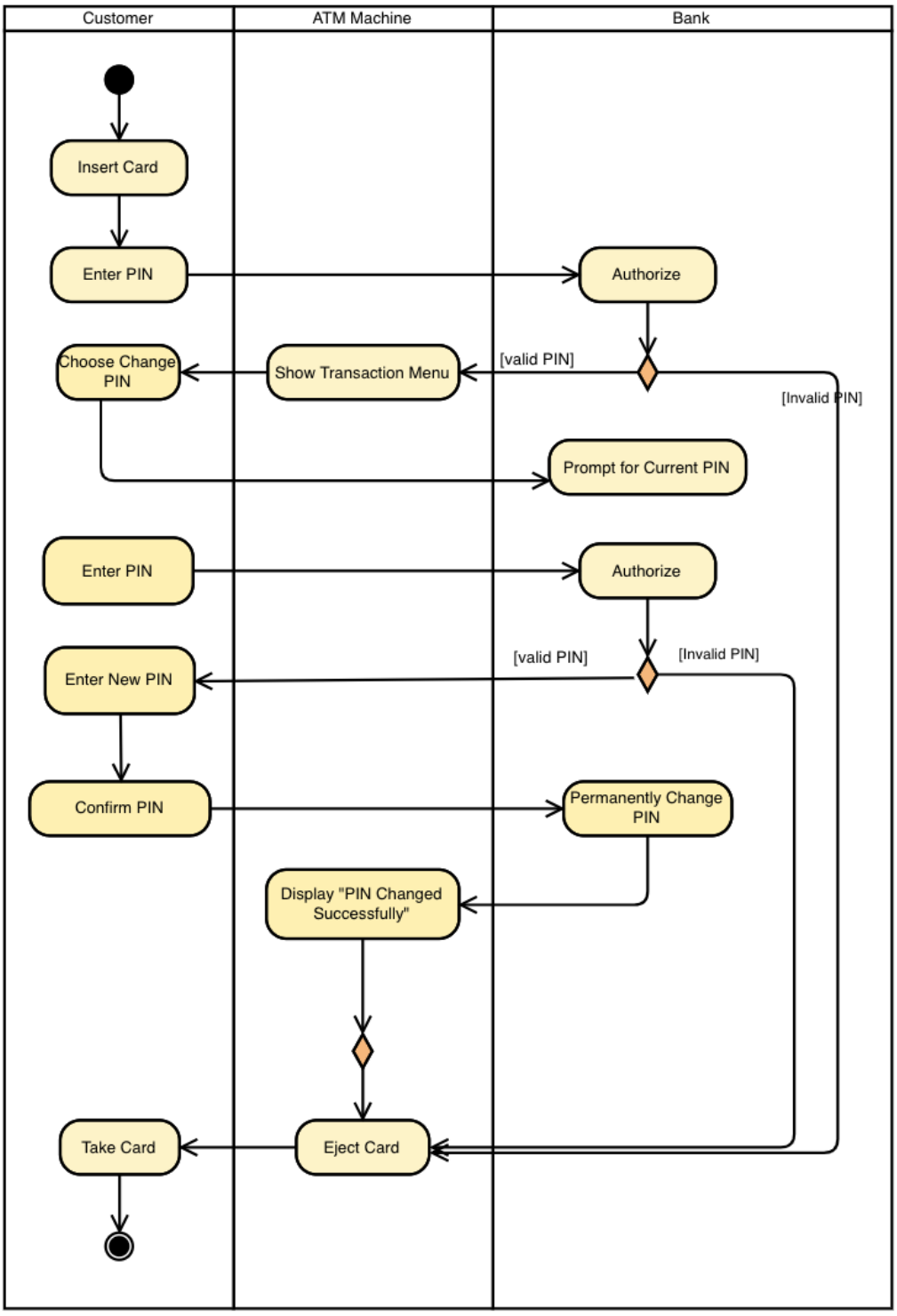
Activity Diagrams

The purpose of an activity diagram is to illustrate the dynamic behaviour of the system. It describes the workflow in particular.

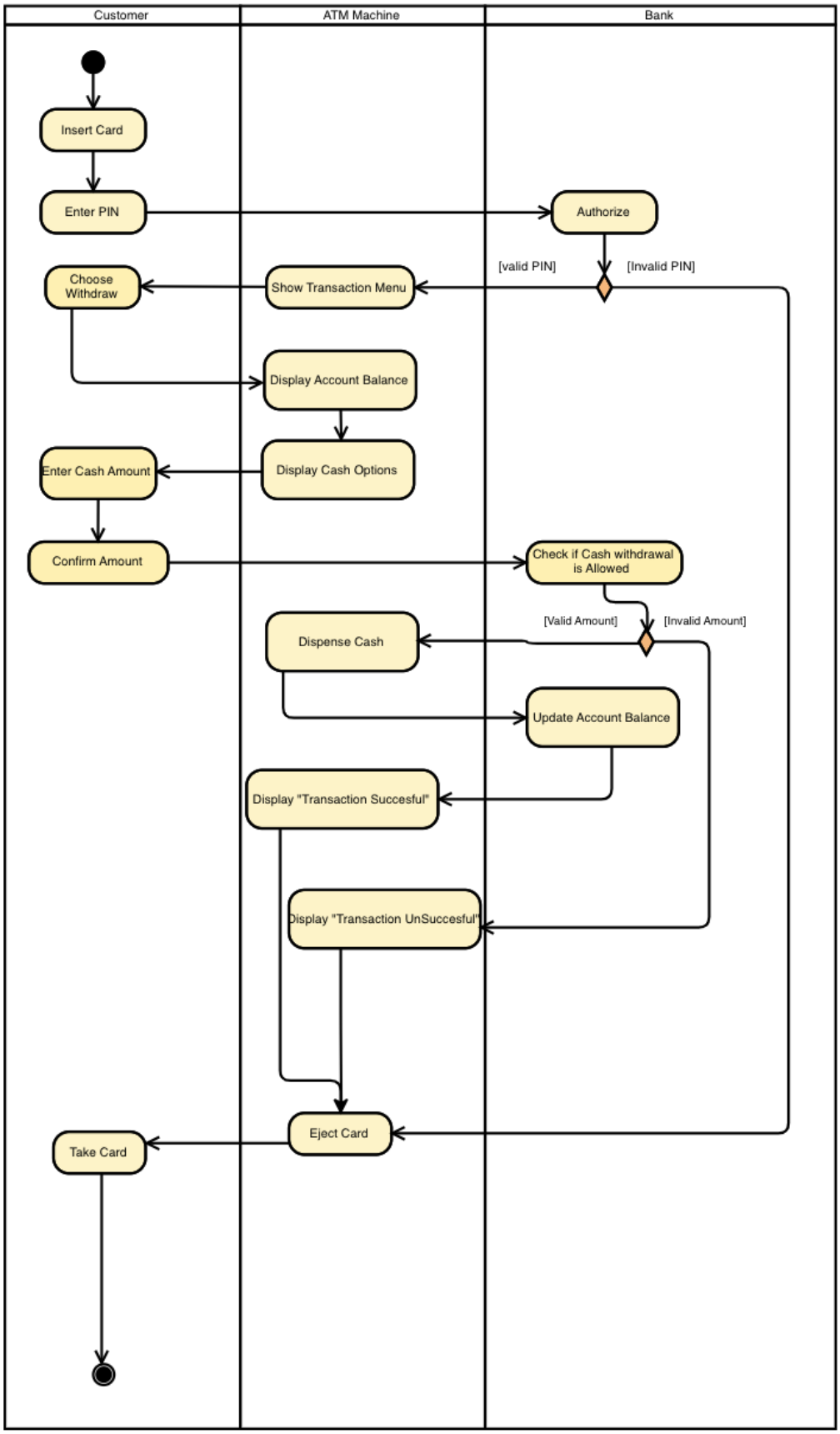
- Credential Check



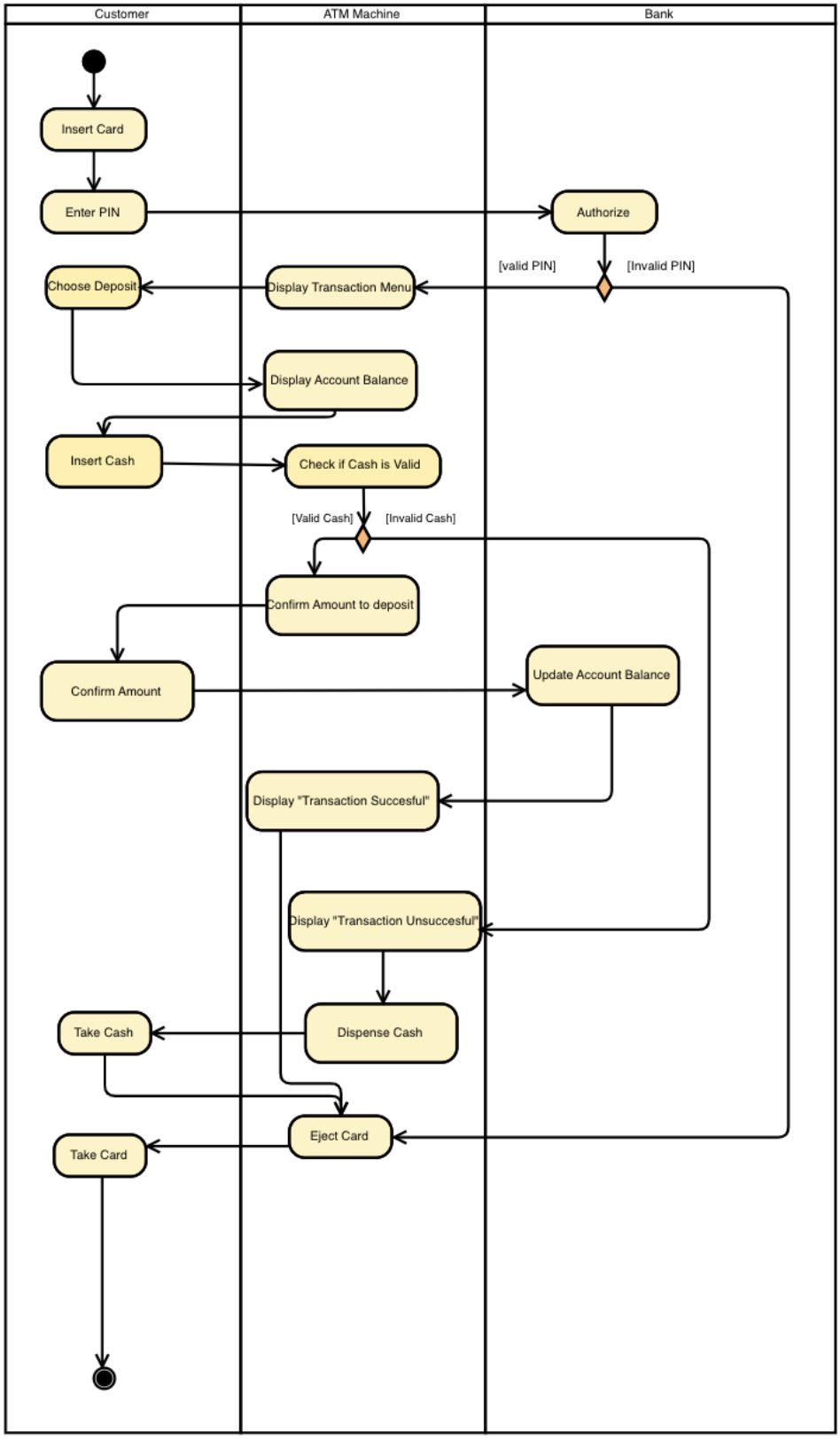
- Change PIN


Customer
ATM Machine
Bank
Insert Card
Enter PIN
Authorize
Choose Change PIN
Show Transaction Menu
[valid PIN]
[Invalid PIN]
Prompt for Current PIN
Enter PIN
Authorize
[valid PIN]
[Invalid PIN]
Enter New PIN
Confirm PIN
Permanently Change PIN
Display "PIN Changed Successfully"
Take Card
Eject Card

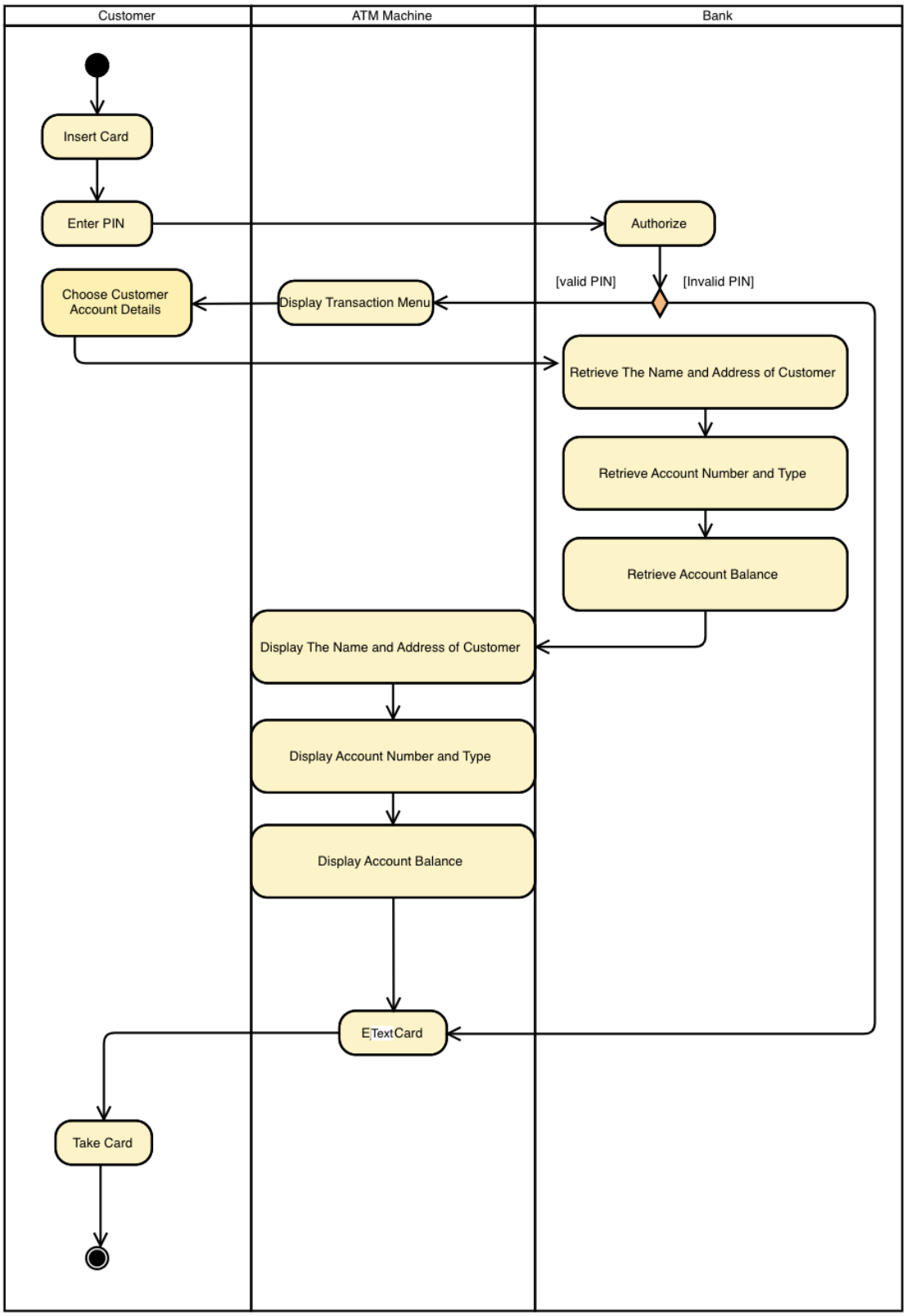
- Withdraw Amount



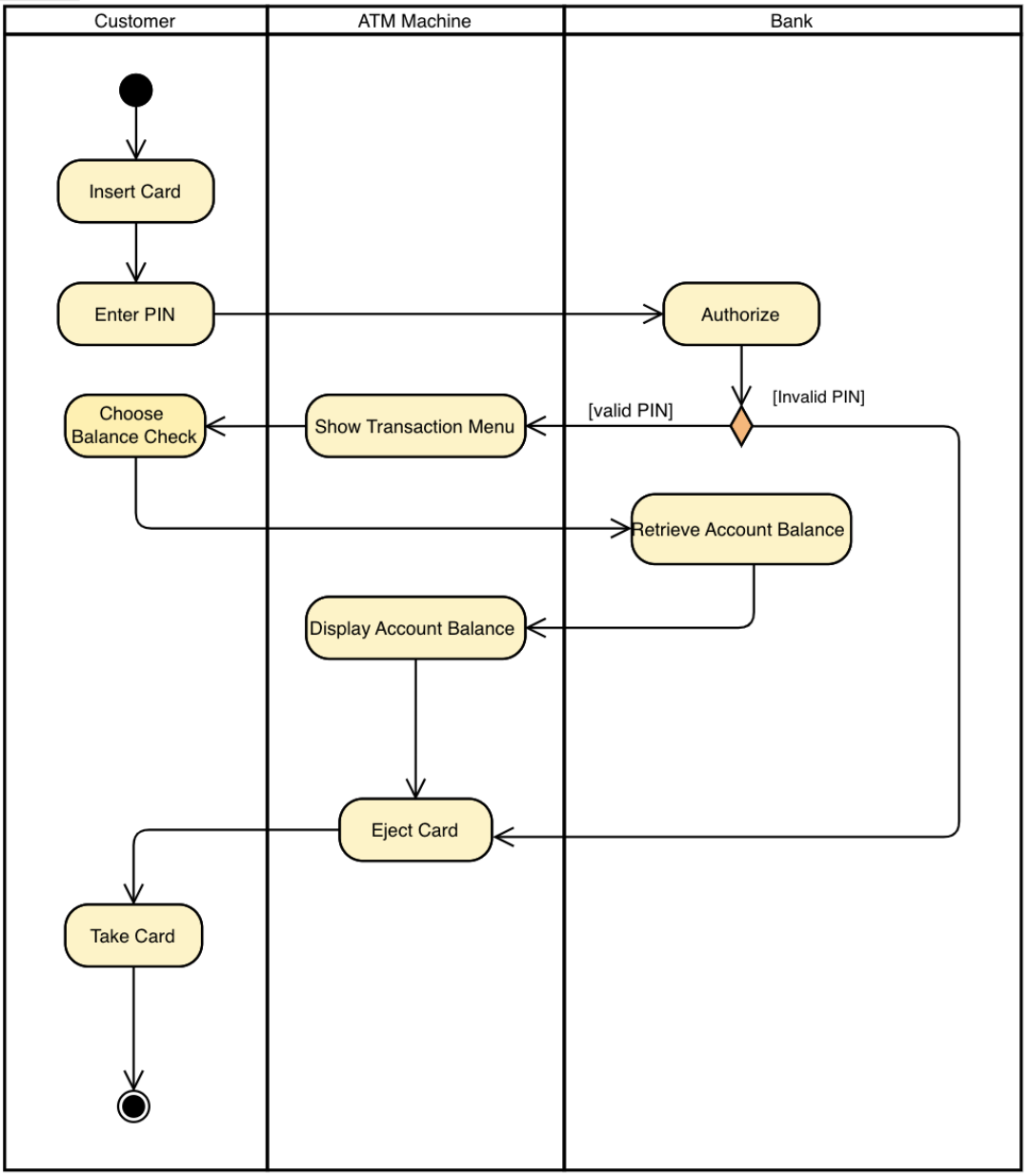
- Deposit Amount



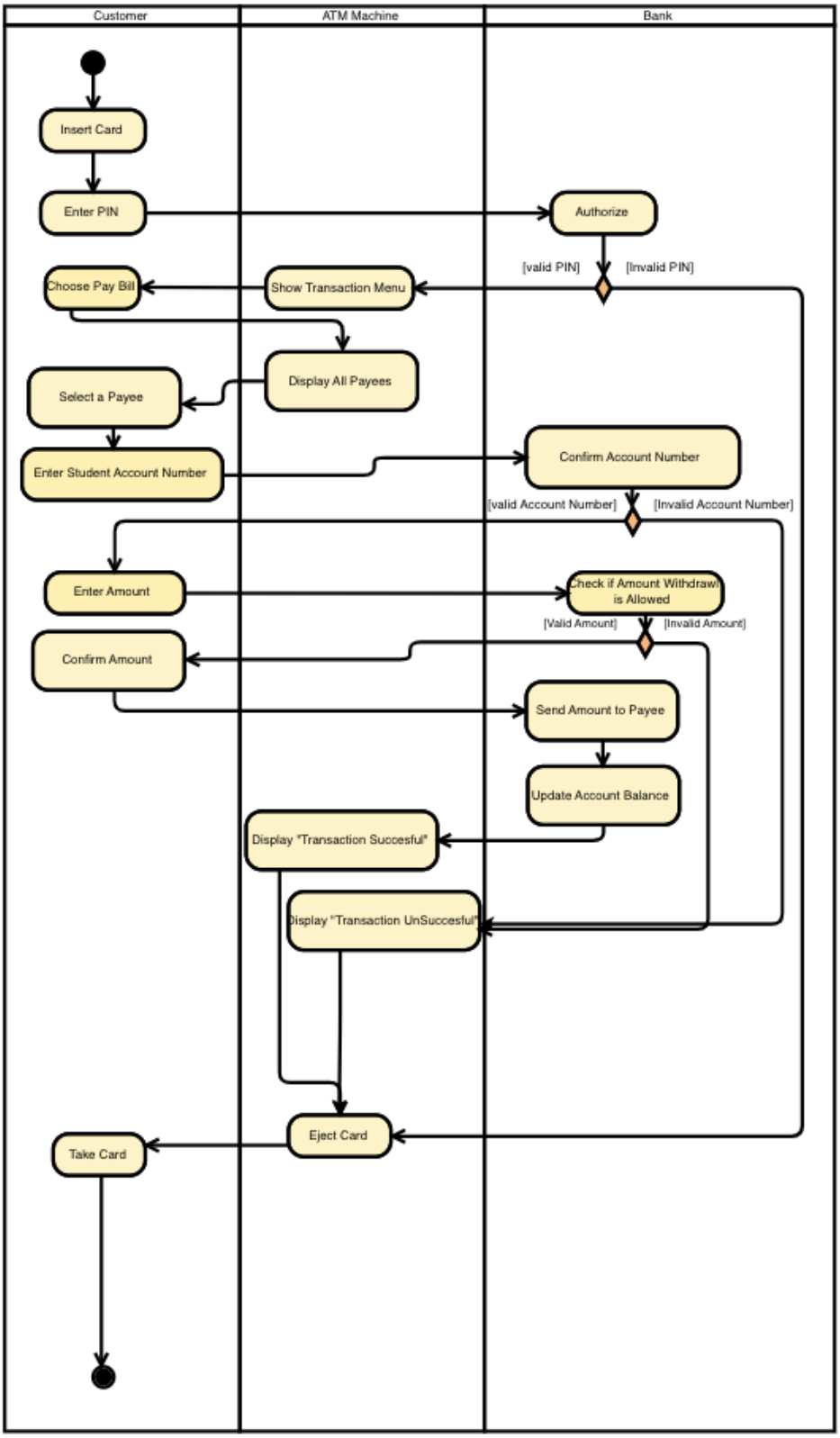
- Customer Account Details



- Balance Check



- Pay Bills


Customer
ATM Machine
Bank
Insert Card
Enter PIN
Authorize
[valid PIN]
[Invalid PIN]
Choose Pay Bill
Show Transaction Menu
Display All Payees
Select a Payee
Enter Student Account Number
Confirm Account Number
[valid Account Number]
[Invalid Account Number]
Enter Amount
Check if Amount Withdrawl is Allowed
[Valid Amount]
[Invalid Amount]
Confirm Amount
Send Amount to Payee
Update Account Balance
Display "Transaction Succesful"
Display "Transaction UnSuccesful"
Eject Card
Take Card

SYSTEM OBJECTS

System objects refer to the elements or components that make up a project's overall structure and define its purpose and scope. These objects can be physical or virtual, tangible or intangible, and include hardware, software, documentation, procedures and protocols. They are instances of system classes.

The system objects implemented in this project are -

- Customer
- ATM Card
- Transactions
- Machine (ATM)
- Account
- Bank/System
- Maintenance

OBJECT DESIGN

Object Design serves as the basis of implementation steps where data structures and algorithms are chosen. It consists of 4 Activities -

1. Reuse of Code, Interfaces, Classes
    - Inheritance
    - Design Patterns
    - Components and solution objects
2. Interface Specification
    - Describes each class interface
3. Object Model Restructuring
4. Object Model Optimization

We shall adopt the Strategy Design Pattern for our ATM banking system. It is a behavioural pattern that allows interchangeable algorithms to be selected at runtime. In the case of an ATM banking system, it can encapsulate different authentication or withdrawal algorithms in separate classes, allowing them to be easily interchanged at runtime. This approach can provide greater flexibility and extensibility to the system, as new algorithms can be added without changing the core code.

Name	Strategy Design Pattern
Synopsis	The Strategy pattern defines a set of algorithms that can be selected at runtime. It separates the selection of the algorithm from the implementation of the algorithm. This pattern allows for dynamic behaviour that can be easily changed without affecting the rest of the system.

Context	In an ATM banking system, there are different types of accounts, and each account has a different limit on the amount of money that can be withdrawn at a time. Additionally, different banks have different policies for charging fees for ATM transactions. The ATM banking system must be able to handle these differences in a flexible way.
Problem Description	The ATM system must ensure error-free money handling and execute the crucial functional requirements, leading to faster, more reliable, and convenient transactions. We may encounter technical challenges, security breaches, bill unavailability, transaction failures, and accessibility discrepancies. The primary aspect is that the ATM banking system must be able to handle multiple withdrawal strategies based on account type and bank policies. At runtime, the system should be able to dynamically determine the most suitable approach without harming the rest of the system.
Forces	The ATM banking system must be adaptable and effective in undertaking different account types and bank regulations. The system should be simple to operate and updates to the withdrawal strategy shouldn't have a significant impact on the rest of the system.
Solution	This issue can be resolved using the Strategy Pattern. As objects, the system can specify a set of withdrawal strategies, each of which implements a common interface. The system could then dynamically determine the right approach during runtime. This structure separates the algorithm selection from the algorithm implementation, allowing for flexibility and easy maintenance. Finally, it enables flexibility in dealing with various withdrawal techniques based on account type and bank rules. The design separates algorithm selection from algorithm implementation, allowing for easy repair and dynamic behaviour.
Consequences	Avoiding implementing the Strategy Design pattern would not allow us to add new transaction options in the future without modifying the Transaction class.
Example	A WithdrawalStrategy interface with the function withdraw(amount: float): void can be built by the system. It may then specify several withdrawal strategies as interface implementations. Based on the account type and banking policies, the ATM banking system can then dynamically identify the right withdrawal tactic at runtime.