Credit Card Processing

Aditya Krishna P 185001010

Armaan C Rao 185001023

Avinash Kartik 185001029

Problem Statement

More often than not, third-parties and physical visits to the bank are involved in the processing of credit card transactions. This software gets rid of all of the hassle by providing all the features online all while ensuring maximum security of the transactions combined with various other exciting features.

Features:

- Request for new card, termination of existing card, paying credit card bills, credit card transactions, modifying existing details.
- Admins can view client info, deny or approve card creation or termination, keep track of and modify credit limit, interest, billing cycle, minimum payment, balance, etc.
- Security by avoiding third-parties, encryption of data, authorization, avoiding physical storage of data.
- Admins can add and remove card plans, which can be viewed by users who can then apply.

Working:

- Users must enter and submit their details and identification through software when applying. After which it will be reviewed by admins who will get back to the user.
- The goal is to maximise the possible features that can be safely executed without the need for a user to visit a physical bank. The customer should be able to obtain and provide as much information as safely possible without the need of a physical meeting with a bank representative. This will help save time and money.
- Automated cross-checking of details provided by both users and admins, can help avoid issues in future and reduce load on bank employees.
- An attractive and easy to navigate UI will make it easy for users to perform any needed actions and will also attract more individuals to apply for a card.
- Loggins and the use of PINs for authentications at every important step ensure there is no fraud occurring. The ability to quickly terminate cards or suspend transactions can help during the event of losing one’s card.

Software Requirements Specifications

1. Introduction:

More often than not, third-parties and physical visits to the bank are involved in the processing of credit card transactions. This software gets rid of all of the hassle by providing all the features online all while ensuring maximum security of the transactions combined with various other exciting features.

- **Purpose**

The purpose of Software Requirements Specification (SRS) document is to describe the external behavior of Credit Card Preprocessing System. Requirements Specification defines and describes the operations, interfaces, performance, and quality assurance requirements of the Online Library System. The document also describes the nonfunctional requirements such as the user interfaces. It also describes the design constraints that are to be considered when the system is to be designed, and other factors necessary to provide a complete and comprehensive description of the requirements for the software. The Software Requirements Specification (SRS) captures the complete software requirements for the system, or a portion of the system. Requirements described in this document are derived from the Vision Document prepared for the Credit Card Preprocessing System.

- **Scope**

The Software Requirements Specification captures all the requirements in a single document. The Credit Card Preprocessing System that is to be developed provides the members of the Bank and employees of the Bank with credit card transaction history, helps modify credit card limit, keeps track of interest, billing cycle etc. The Credit Card Preprocessing System is supposed to have the following features:

a. Request for new card, termination of existing card, paying credit card bills, credit card transactions, modifying existing details.
b. Admins can view client info, deny or approve card creation or termination, keep track of and modify credit limit, interest, billing cycle, minimum payment, balance, etc.
c. Security by avoiding third-parties, encryption of data, authorization, avoiding physical storage of data.
d. Admins can add and remove card plans, which can be viewed by users who can then apply.

The features that are described in this document are used in the future phases of the software development cycle. The features described here meet the needs of all the

users. The success criteria for the system is based in the level up to which the features described in this document are implemented in the system.

- **Definitions, Acronyms and Abbreviations**

  CCPS - Credit Card Processing System

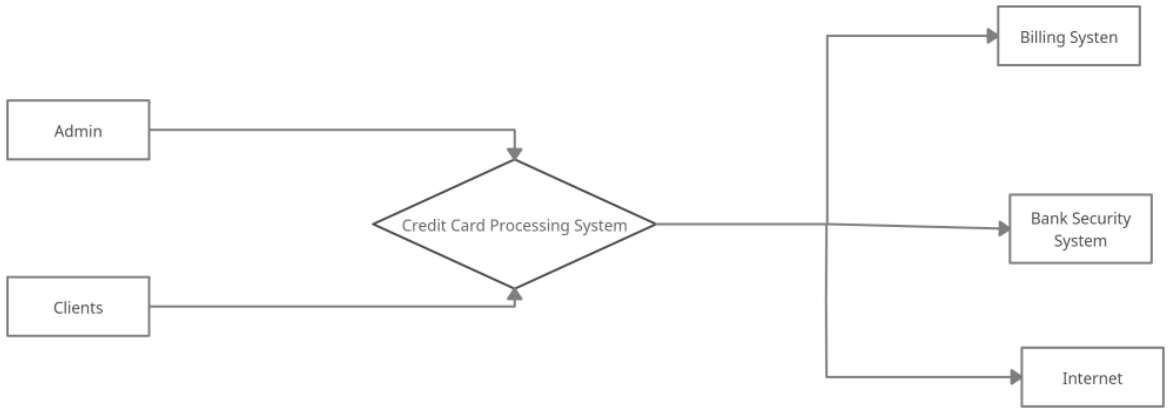
  OTP - One Time Password

- **Overview**

  The SRS will provide a detailed description of the CCPS. This document will provide the outline of the requirements, overview of the characteristics and constraints of the system.

2. Overall description:

Overview



- **Product Perspective**

  The Credit Card Processing software is a package to be used by Banking Systems to improve the efficiency of managing all the actions and transactions related to their credit cards. Library employees and Users. The software to be developed benefits greatly the admins and the customers of the bank willing to employ it. The software provides all the plans that are offered by the and suggests the appropriate one specific to one's back details. The software ensure that all the details are up-to date can so that the customers will be updated all the time and can make decisions based on the real time values. The complete overview of the system is as shown in the overview diagram below: The product to be developed has interactions with the users: Admin and Members who are the clients. The product has to interact with other systems like: Internet, Billing System, Bank security system and the Database of the bank.

- **Product Function**

  The Credit Card Preprocessing System provides online real time information about the members of the bank and their bank details and all the latest features introduced by the bank. The Product functions are more or less the same as described in the product perspective. The functions of the system include the system providing different type of services based on the type of users [Member/Librarian].

a. The clients should be provided with the real time information of the bank.
b. Provisions for clients to give or receive money to / from anyone else.
c. Provisions for clients to get a new credit card / cancel an existing one.
d. The clients are provided with various plans from which they can select what they want.
e. The admins can view the information of any client and modify them if needed.
f. The clients should be able to withdraw only as much as they have in their bank accounts and if their balance goes below the minimum amount, they should deposit how much ever required within a given period.
g. The system uses bank information to provide the facilities to the users.

- **User characteristics**

  The users of this software are the clients of the bank and the administrators of the bank who monitor the activities of the clients and makes all the clients are satisfied with the service. The members and the admins are assumed to have basic knowledge of the computers and Internet browsing. The administrators of the system to have more knowledge of the internals of the system and is able to rectify the small problems that may arise due to disk crashes, power failures and other catastrophes to maintain the system. The proper user interface, user's manual, online help and the guide to install and maintain the system must be sufficient to educate the users on how to use the system without any problems.

- **Constraints**

a. The information of all the clients must be stored in a database that is accessible by the Credit Card Processing System.
b. The bank security system must be compatible with the Internet applications.
c. The Credit Card Processing System is connected to the bank computer and is running all 24 hours a day.
d. The clients access the Credit Card Processing System from any computer that has Internet browsing capabilities and an Internet connection.
e. The billing system is connected to the Credit Card Processing System and the database used by the billing system must be compatible with the interface of the Credit Card Processing System.
f. The users must have their correct usernames and passwords to enter into the Credit Card Processing System.

- **Assumptions and dependencies**

a. The users have sufficient knowledge of computers.
b. The bank computers should have Internet connection and Internet server capabilities.
c. The users know the English language, as the user interface will be provided in English.
d. The product can access the bank database.

3. Specific Requirements:

This section describes in detail all the functional requirements.

- **Functionality**
    a. Logon Capabilities
        i. The system shall provide the users with logon capabilities.
    b. Checking credit card details
        i. Customers can get their credit card details including personal details, transaction details, credit limit, balance etc.
    c. Submit various requests to the system
        i. Customers can request for new card, termination of existing card, applying for a credit card plan, paying credit card bills, credit card transactions, modifying existing details.
    d. Administrative Capabilities
        i. Admins can view client info, deny or approve card creation or termination, keep track of and modify credit limit, interest, billing cycle, minimum payment, balance, add and remove card plans etc.
    e. Security
        i. The system provides security by avoiding third-parties, encryption of data, authorization, avoiding physical storage of data

- **Usability**
    a. The software can be used to process credit card payments on e-commerce websites.
    b. The software can be used in many apps like Paytm and Gpay.
    c. The system is user friendly and self-explanatory.

- **Reliability**

    The system has to be very reliable due to the importance of data and the damages incorrect or incomplete data can do.

    a. Availability
        i. The system is available 100% for the user and is used 24 hrs. a day and 365 days a year. The system shall be operational 24 hours a day and 7 days a week.
    b. Mean Time Between Failures (MTBF)

i. The system will be developed in such a way that it may never fail, as it involves an individual's hard-earned money.

c. Mean Time to Repair (MTTR)
   i. Even if the system fails, the system will be recovered back up within an hour or less.

d. 3.3.4 Maximum Bugs or Defect Rate
   i. Not specified.

- **Performance**

a. Response Time
   i. The CCP should load the payment screen immediately so as to facilitate fast payments. The information is refreshed every two minutes. The system shall respond to the member in not less than two seconds from the time of the request submittal. The system shall be allowed to take more time when doing large processing jobs.

b. Administrator Response
   i. The system shall take as less time as possible to provide service to the administrator.

c. Throughput
   i. The number of transactions is directly dependent on the number of users, the users may be the admin, clients of the bank.

d. Capacity
   i. The system is capable of handling 250 users at a time.

e. Resource Utilization
   i. The resources are modified according the user requirements and also according to the books requested by the users.

- **Supportability**
  a. Internet Protocols
     i. The system shall be comply with the TCP/IP protocol standards and shall be designed accordingly.
  b. Maintenance
     i. The maintenance of the system shall be done as per the maintenance contract.
  c. Standards
     i. The coding standards and naming conventions will be as per the Indian standards.

- **Design Constraints**
  a. Software Language Used
     i. The languages that shall be used for coding the Credit Card Processing System are Python, HTML, CSS, Javascript.
  b. Development Tools
     i. Will make use of the available Java Development Tool kits for working with Java Beans and Java Server Pages. Also will make use of the online

references available for developing programs in ASP, HTML and the two scripting languages, JavaScript and VBScript.

c. Class Libraries
    i. Will make use of the existing Java libraries available for JSP and Servlets. Also we need to develop some new libraries for the web-based application. Also will develop new programs using ASP and scripting languages.

- **On-line User Documentation and Help System Requirements**

  Online help is provided for each of the feature available with the Credit Card Processing System. All the applications provide an online help system to assist the user. The nature of these systems is unique to application development as they combine aspects of programming (hyperlinks, etc) with aspects of technical writing (organization, presentation). Online help is provided for each and every feature provided by the system. The User Manual describes the use of the system to admin and clients. It describes the use of the system on mobile systems. The user manual should be available as a hard copy and also as online help. An installation document will be provided that includes the installation instructions and configuration guidelines, which is important to a full solution offering. Also, a Read Me file is typically included as a standard component. The Read Me includes a "What's New with This Release" section, and a discussion of compatibility issues with earlier releases. Most users also appreciate documentation defining any known bugs and workarounds in the Read Me file.

- **Purchased Components**

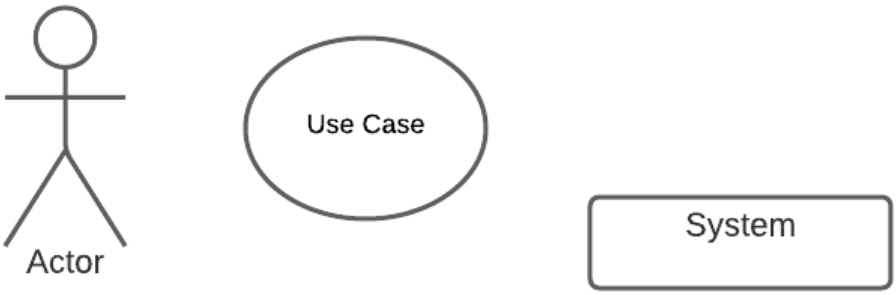
  The System Administrator will need to purchase the license for IIS Server. Mostly it is available with Windows Environment. So, the system need not purchase any licensing products.

Identify Use Cases and develop the Use Case model

Aim:

To identify the use cases and develop the use case model for the credit card processing system.

Notations:



Identification of Actors:

1. Client
2. Admin

Identification of Scenarios:

Success scenarios:

1. Successfully logged in as client or admin
2. Successfully created account
3. Credit card applied for and application confirmed as valid
4. Successfully paid balance owed
5. Successfully changed card plan
6. Transactions through chosen card suspended
7. Chosen card terminated
8. Account deleted

Failure scenarios:

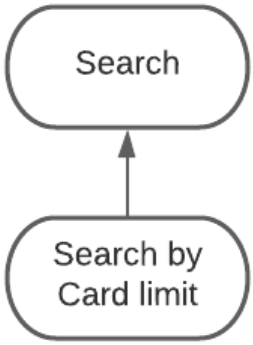
1. Log in attempt failed due to incorrect username and/or password
2. Could not create account due to email already taken or password strength insufficient
3. Credit card applied for and application detected as invalid
4. Attempt to pay balance failed
5. Could not terminate card due to existing payment due
6. Could not delete account due to existing payment due

Relating Use cases:

1. **Generalization:**

Generalization refers to the relationship between two use cases, where one is the child of the other. The child inherits the structure, behaviour, and relationships of the parent. The child can also add to or override any behaviour

of the parent. The child can be substituted in place of the parent in any place it appears.



To show generalization, we make an arrow from the child point to the parent.

2. **Association (include, extend):**

Includes: An include relationship between use cases means that the base use case explicitly incorporates the behaviour of another use case at a location specified in the base.

Representation:

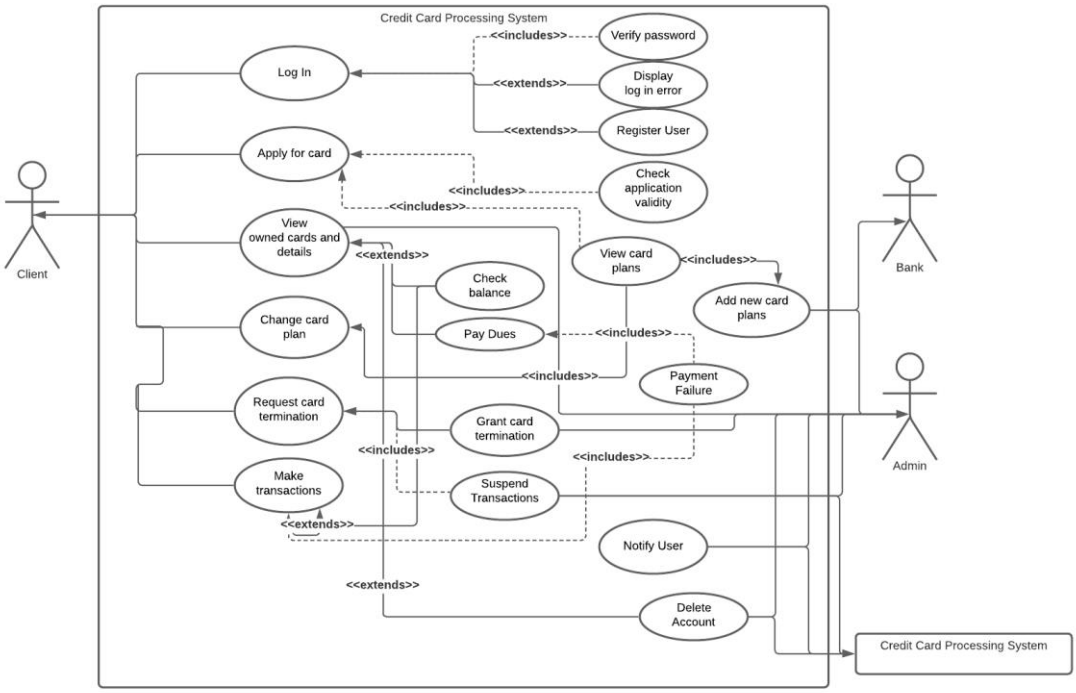
Extends: An extends relationship between use cases means that the base use case implicitly incorporates the behaviour of another use case at a location specified indirectly by the extending use case.

Representation:

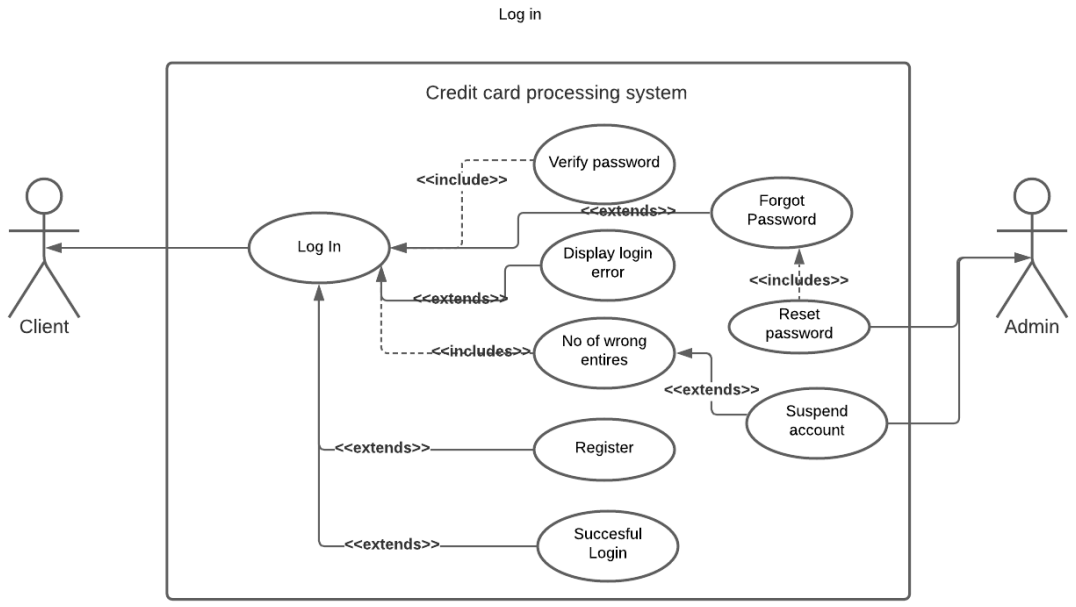
with their respective notations

Use Case Diagrams:

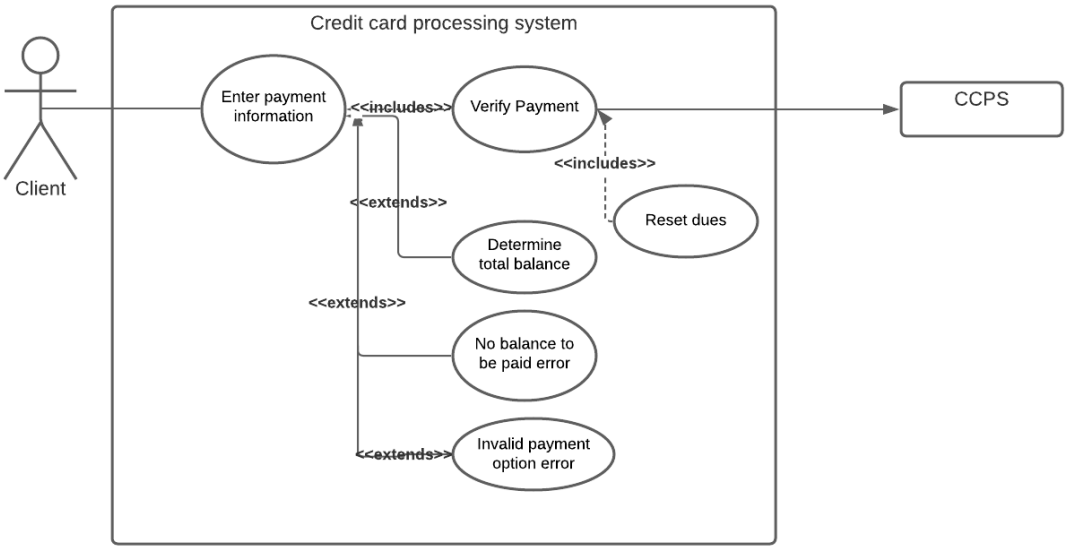
1. **Main Success Scenario:**



2. Alternate Scenarios:

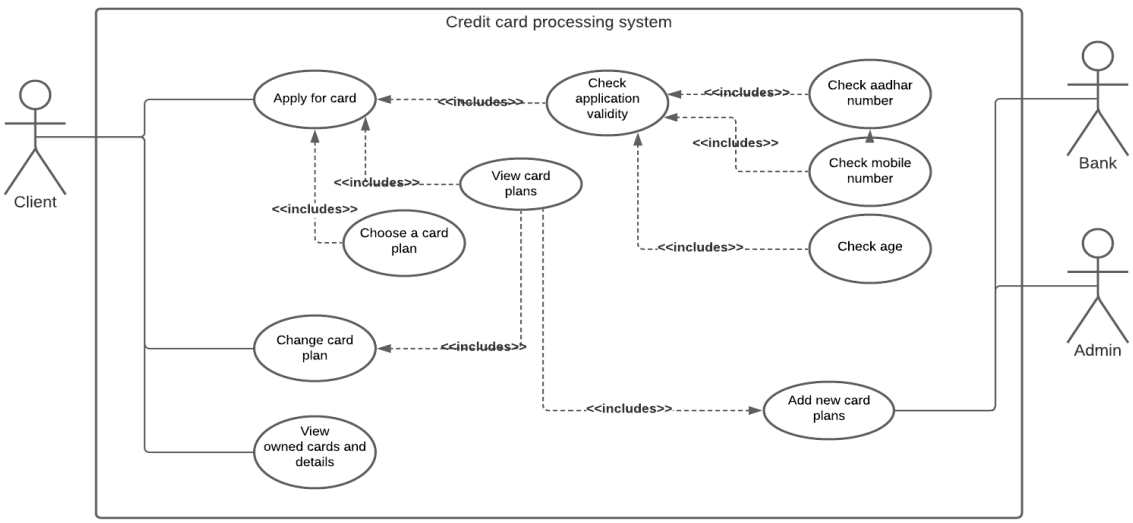


Pay Dues:

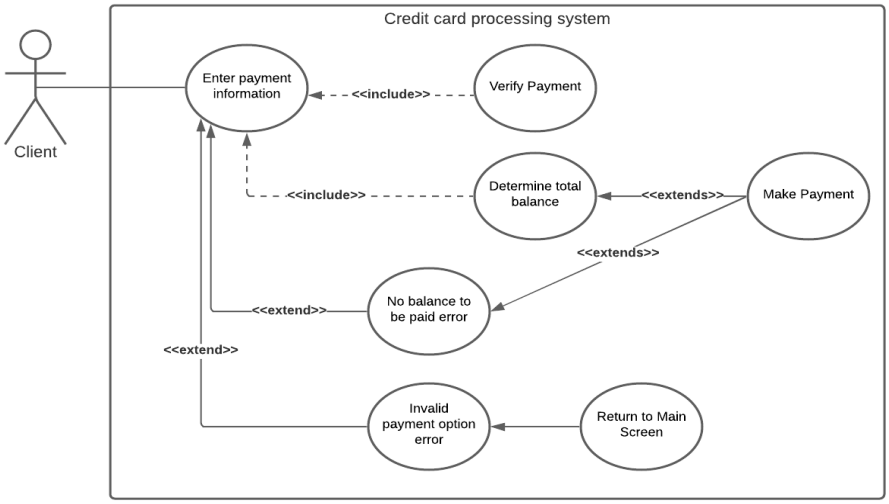


3. Subfunctions:

Card Plans



Make Transactions



Fully Dressed Use Case Descriptions:

Use Case Name:

Scope: Credit card management system

Level: User goal

Primary Actor: Customer

Stakeholders and Interests: Banks, Corporations and Customers (General Public)

Preconditions: Users have the required information for applying for a card on hand.

Success Guarantee: All of users' information is correct.

Main success scenario:

1. Customer enters log in page
2. Customer chooses to create account
3. System verifies email and password
4. After successful log in they are redirected to home page
5. Customer views list of card plans
6. Customer chooses card plan to apply for
7. Customer enter details and applies for a card
8. Customer gets confirmation for card
9. Customer transfers money from card to another account
10. Customer receives acknowledgement of transaction
11. Customer logs out

Alternate Scenario:

1. Customer enters log in page
2. A. Customer enters email and password

1. System verifies info as invalid

2. System Displays login error

3. Customer continues to enter invalid log in details 5 times

After 5^{th} try customer account is suspended

B. Customer chooses to create account

1. Customer enters their email and chosen password

2. Password strength is too low

3. System displays error message

C. Customer chooses forgot password

1. Customer enters phone number.

2. Password reset link is sent to customer via sms.

3. Customer enters new password twice, once for confirmation.

4. Password is reset.

Subfunction:

1. Customer enters payment domain.
2. Customer enters card details and payment information
    a. Payment details are verified
    b. Bank details are checked
    c. If sufficient balance is present, payment is valid
    d. If insufficient balance, insufficient balance error is displayed
3. Return to home screen once complete.

Special Requirements:

1. Digital Device with Internet Connectivity
2. Recovery system when database fails

Frequency of Occurrence: Rarely

Documentation:

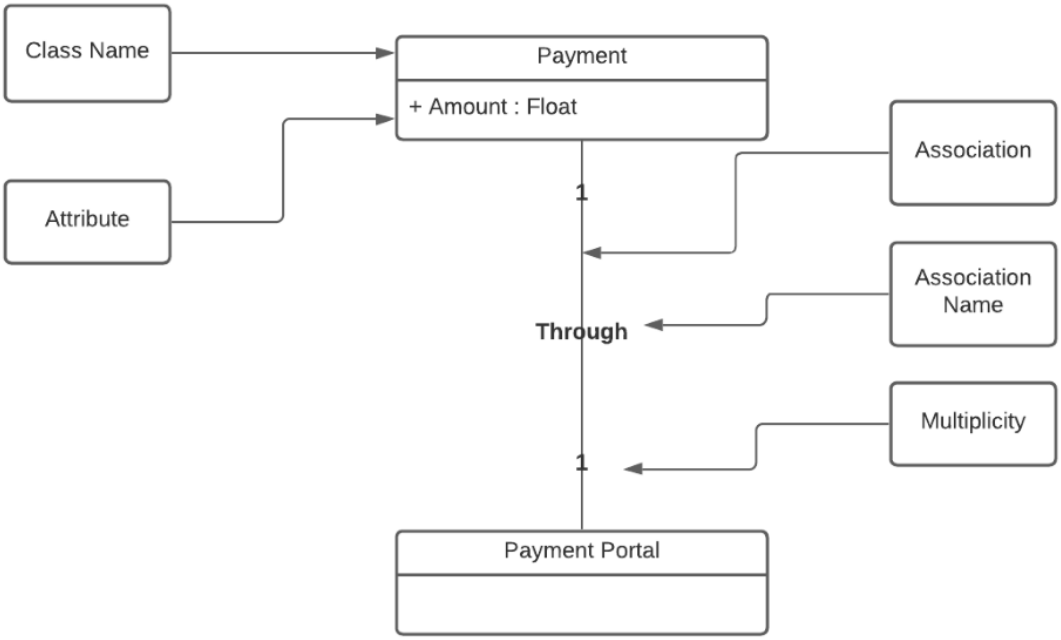
Developing the use case models helped us get a deeper understanding as to what goes on in the development of a software application, and helped us understand our project better, and how to go about implementing the features and functionality.

Ex – 4

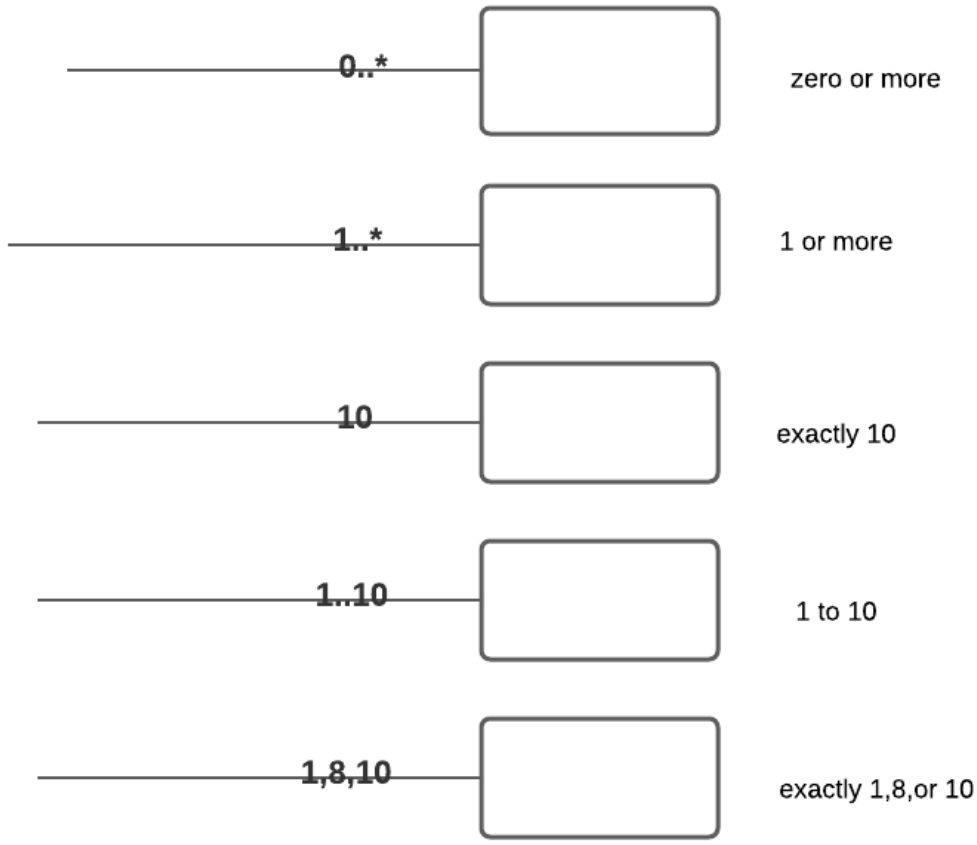
Map a problem to an UML domain model

Aim: Creation of an UML domain model for a problem domain.

UML Notation For Domain Model:

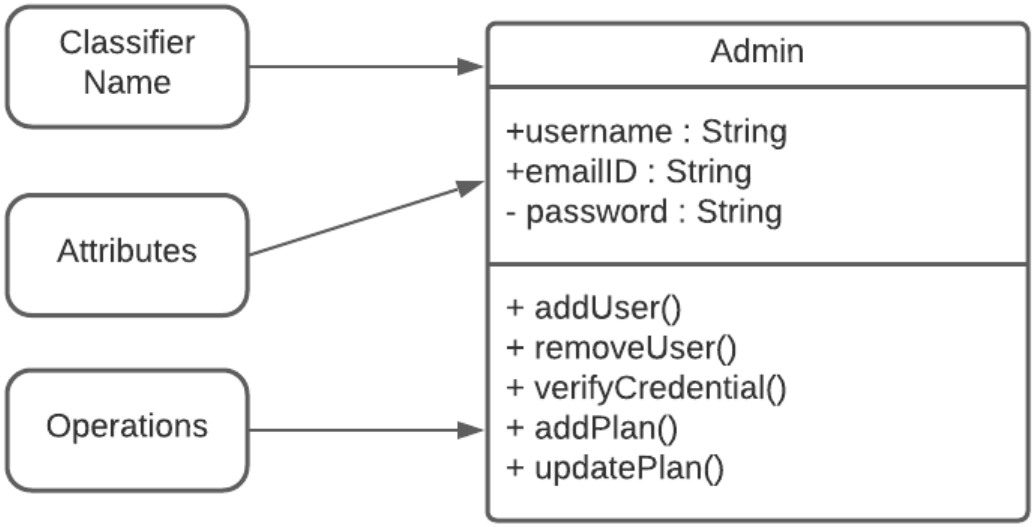


Multiplicities:

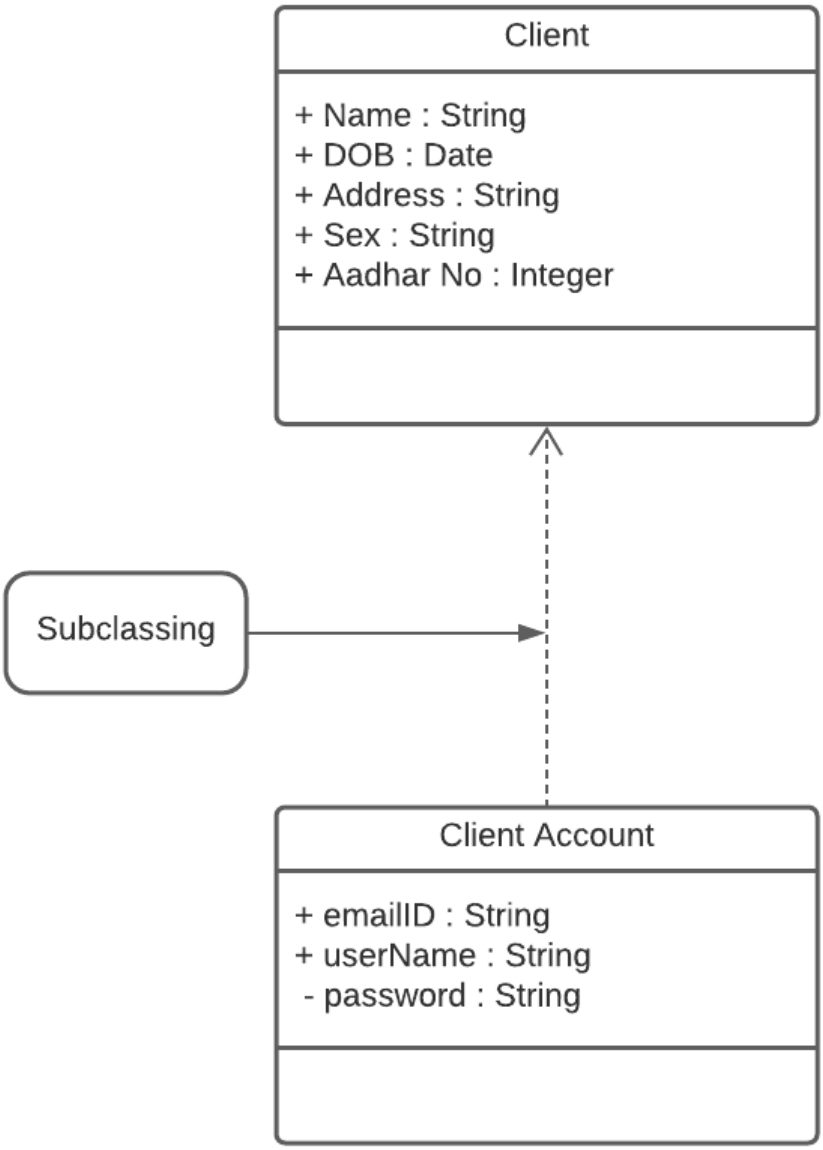


UML Notations are Denoted as Follows:

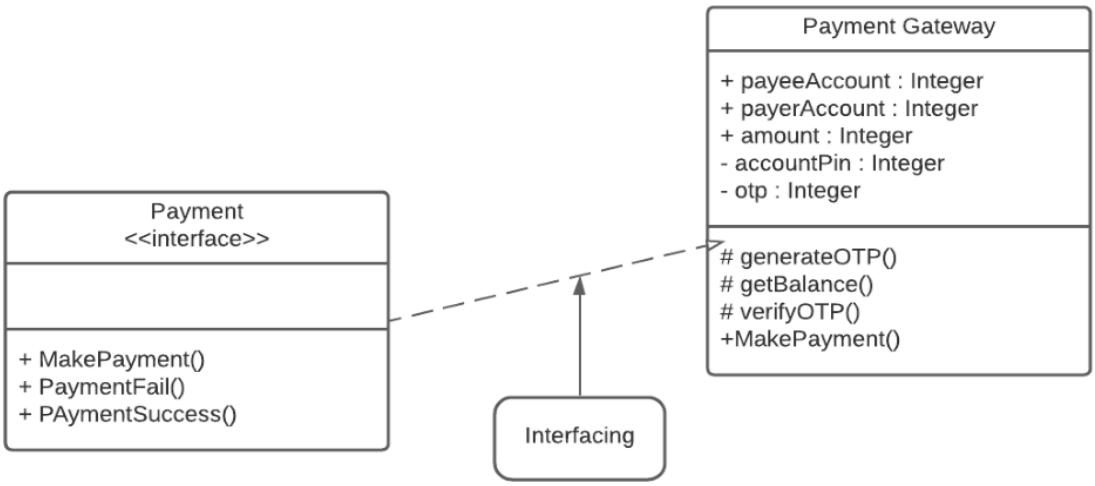
There are usually 3 parts of a class -



Subclassing is done as follows -



Interfaces are shown with the keyword <<interface>>, and denoted as follows :



Main success scenario:

1. **Client** enters log in page.
2. Client chooses to create **account**.
3. **Credit Card Processing System** verifies email and password and stores the details into the **client database**.
4. After successful log in they are redirected to home page.
5. Client views list of **card** plans.
6. Client chooses card plan to apply for.
7. Client enter details and applies for a card.
8. **Admin** approves the card.
9. Client gets confirmation for card.
10. Client **transfers** money from card to another account via the **payment** gateway.
11. Client receives acknowledgement of transaction after verification from the **bank database**.
12. Client logs out.

The noun phrases identified from the main success scenario are:

Client	Client Account	CCPS
Client Database	Card	Transfer
Payment Portal	Bank Database	Admin

Identification of associations:

1. **Association category list:**
    a. CCPS, Client Account, User Database, Bank Database are connected by Association.
    b. Card and Bank Database are connected by composition.
    c. Payment Interface and Payment Gateway are connected by Composition.
    d. Card and Payment interface are connected by Composition.
    e. Card and Admin are connected by Aggregation.
    f. Client Account and Client are connected by Aggregation.

2. **Definitions of associations and their notations:**
    a. **Association:** Association relationship is a structural relationship in which different objects are linked within the system. It exhibits a binary relationship between the objects representing an activity.

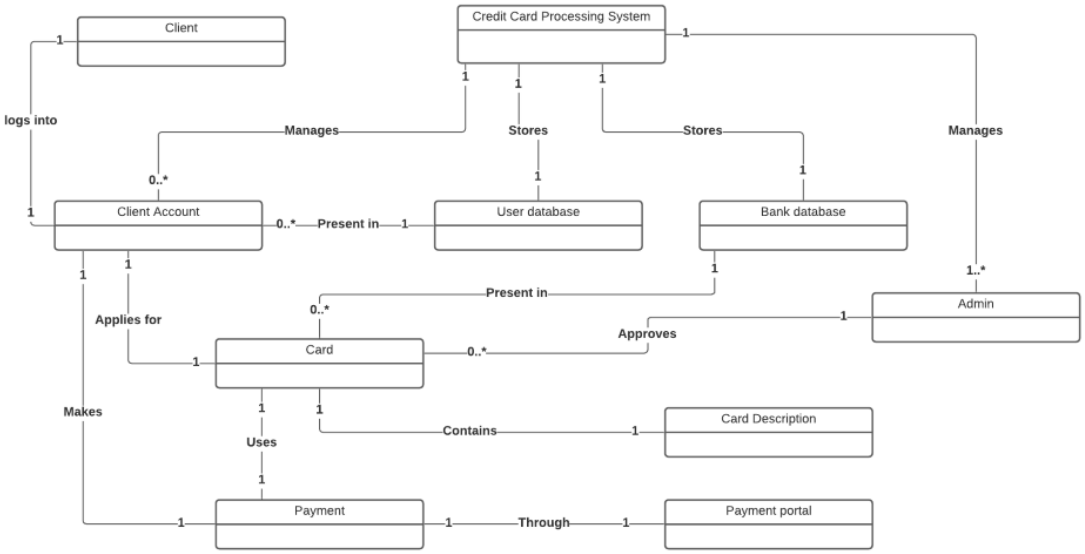
    b. **Aggregation:** Aggregation is a subset of association, is a collection of different things. It represents a relationship. It is more specific than an association. It describes a part-whole or part-of relationship. It is a binary association, i.e., it only involves two classes. It is a kind of relationship in which the child is independent of its parent.

    c. **Composition:** The composition is a part of aggregation, and it portrays the whole-part relationship. It depicts dependency between a composite (parent) and its parts (children), which means that if the composite is discarded, so will its parts get deleted. It exists between similar objects.

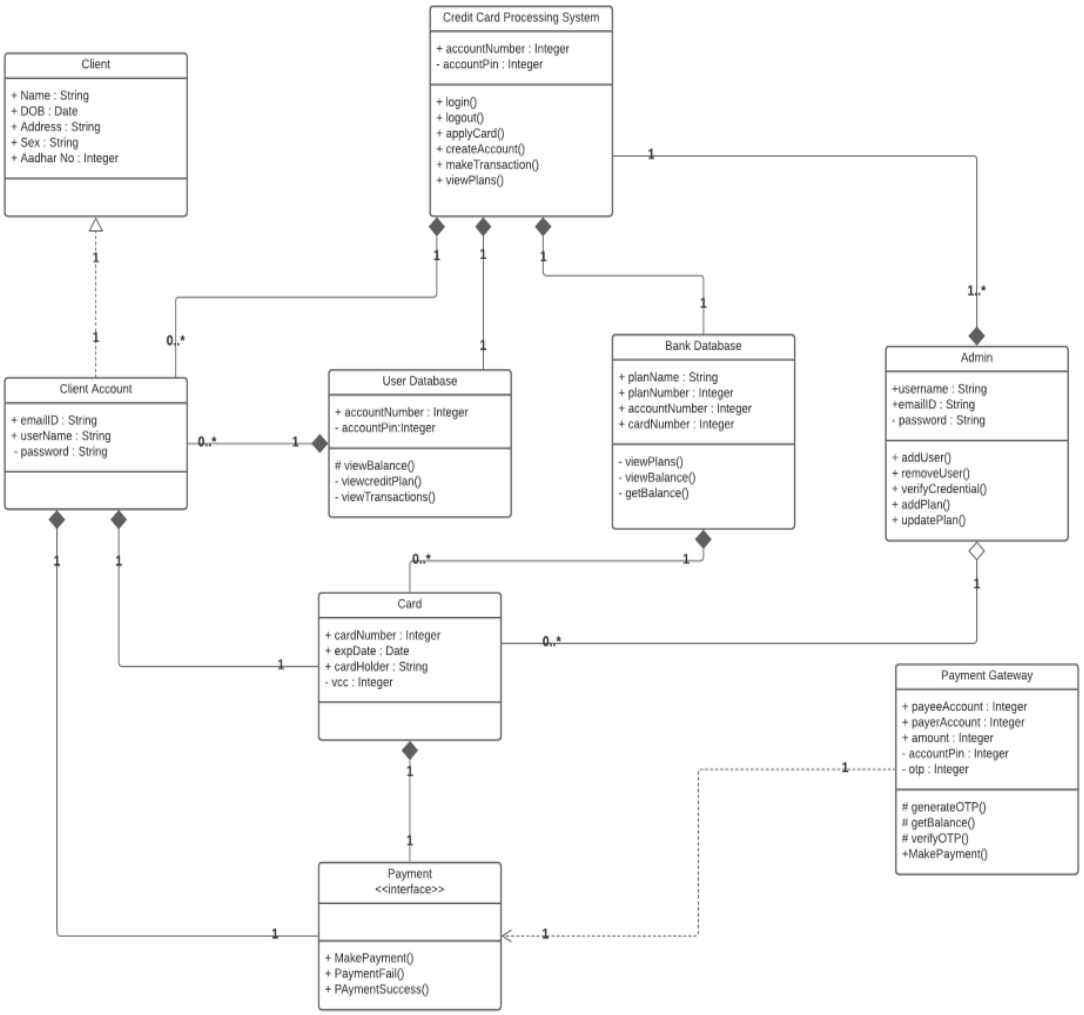
c. Multiplicity based associations:

1. Client and client account are connected in a 1:1 relation.
1. Credit card processing system and client account are connected in a 1:0..* relation.
2. Credit card processing system and bank database are connected in a 1:1 relation.
3. Credit card processing system and user database are connected in a 1:1 relation.
4. Credit card processing system and admin are connected in a 1:1..* relation.
5. Client account and card are connected in a 1:1 relation.
6. Client account and payment are connected in a 1:1 relation.
7. Bank database and card are connected in a 1:0..* relation.
8. Admin and card are connected in a 1:0..* relation.
9. Card and payment are connected in a 1:1 relation.
10. Card and card description are connected in a 1:1 relation.
11. Payment and payment portal are connected in a 1:1 relation.

Domain Model Diagram:



Class Diagram:



Documentation:

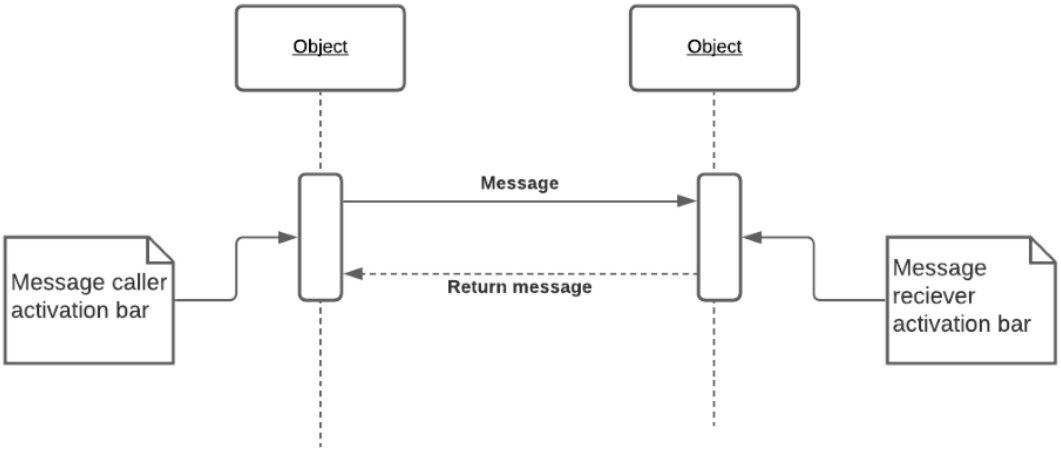
From this exercise we learned how to make UML domain models. The UML domain modelling being structural it emphasizes on organization of the system which helps in visualizing the structure of the CCP System. We also learned how to make UML class diagrams and their importance. UML Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. It also helps in mapping the UML domain model directly to an Object-Oriented Programming language which aids in implementation.

EX - 5

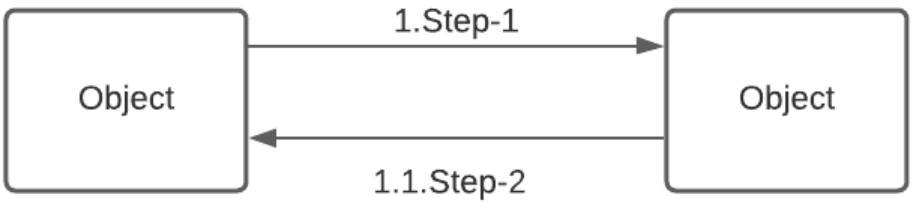
Aim: To draw sequence and communication diagrams for the credit card processing system.

Notations:

1) Sequence Diagram:

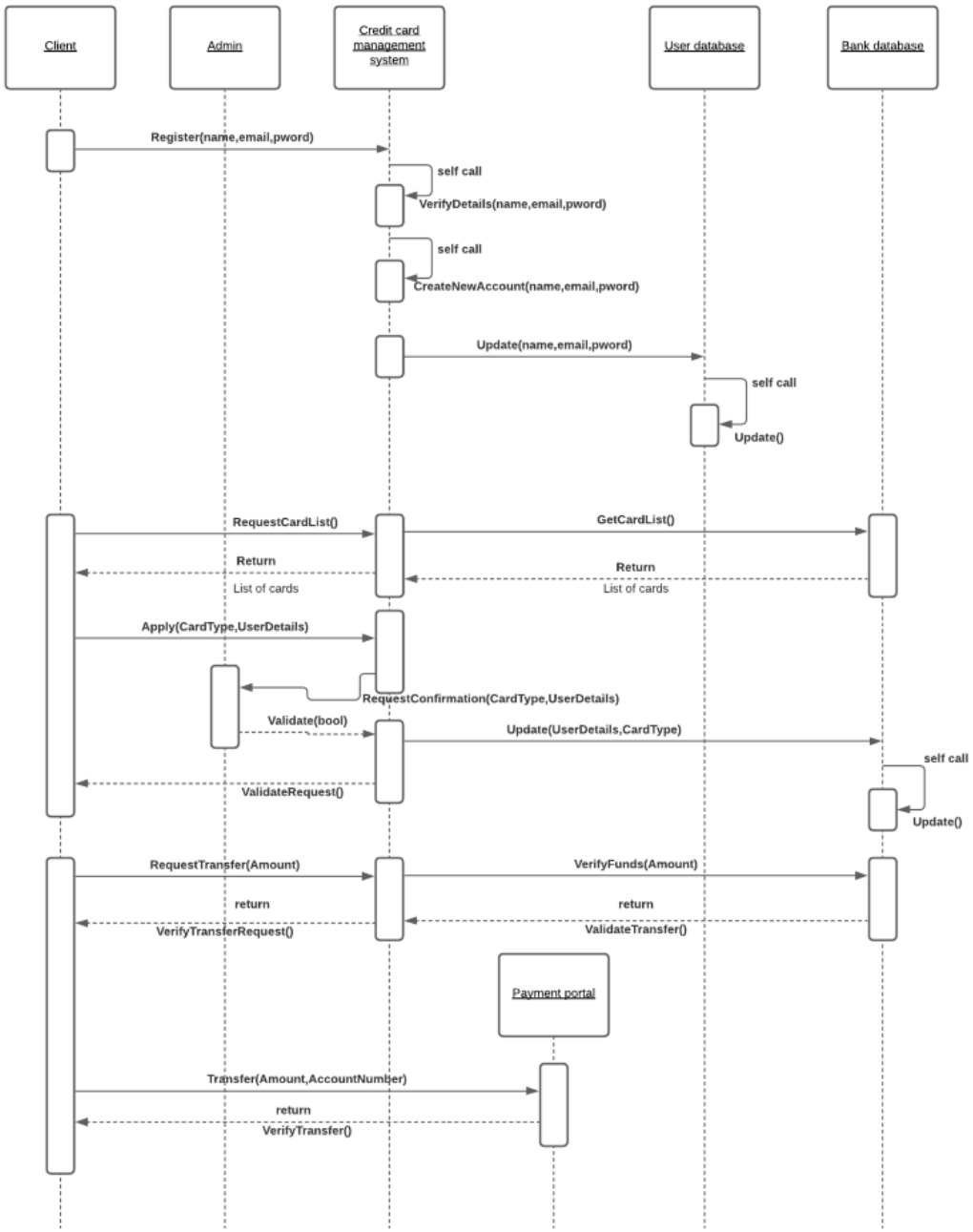


2) Communication Diagram:

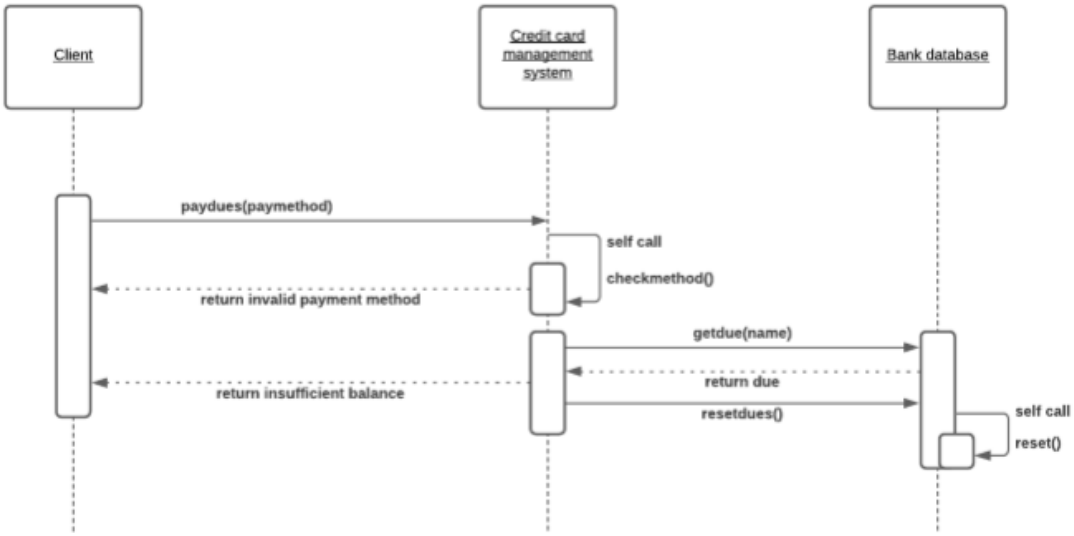


Sequence Diagrams:

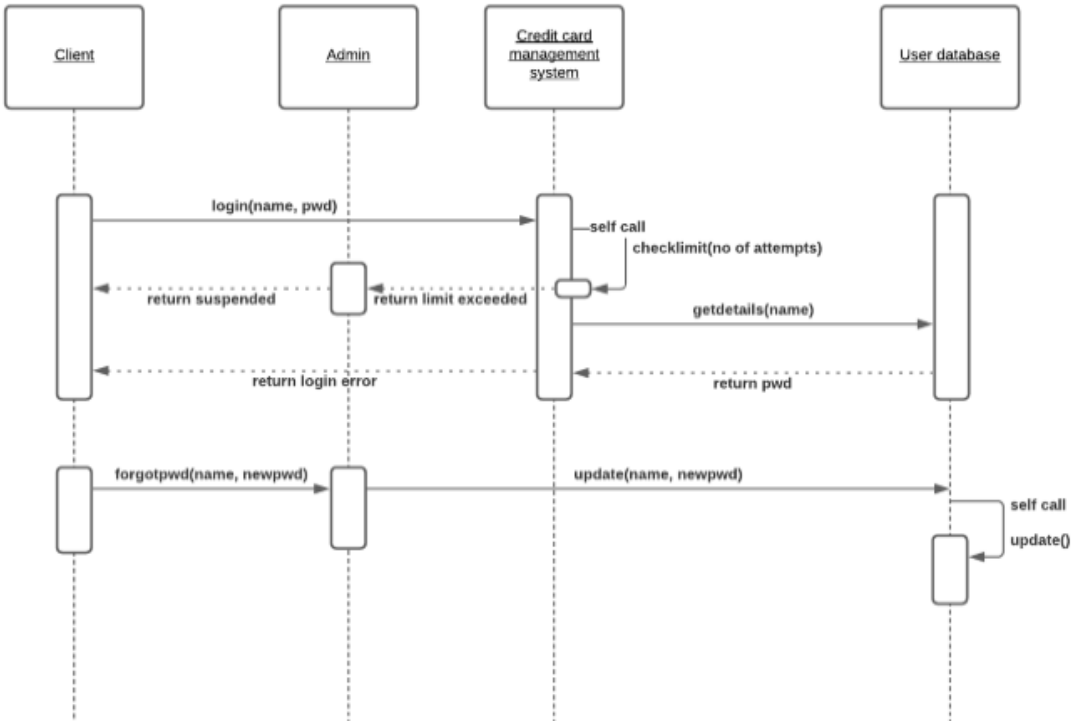
1) Main Success Scenario:



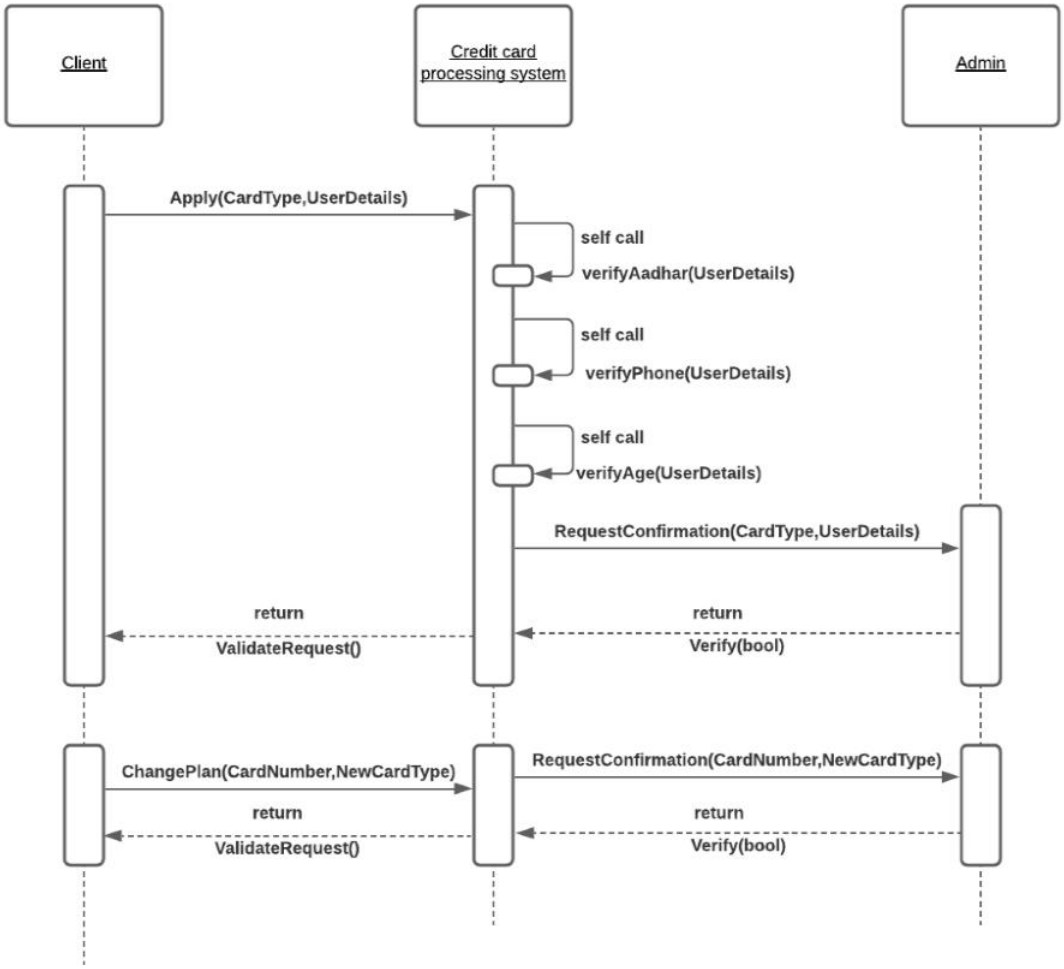
2a) Alternate Scenario: Pay Dues



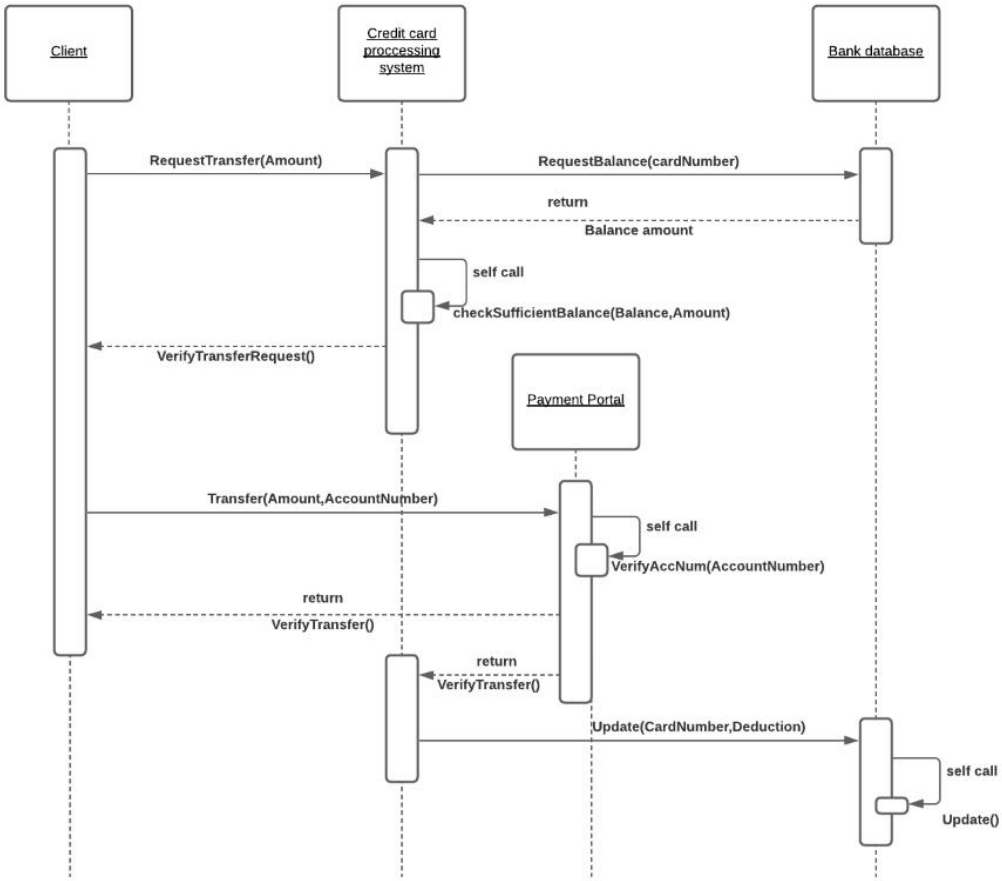
2b) Alternate Scenario: Log in



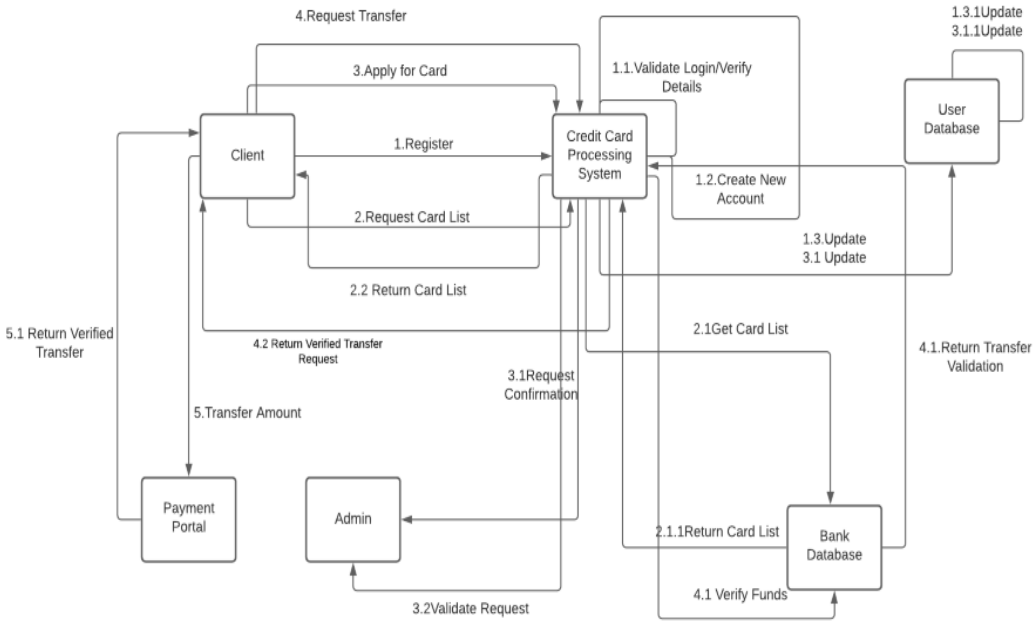
3a) Subfunction: Apply/Modify card



3b) Subfunction: Transfer to another account



Communication Diagram:



<u>Documentation:</u>

From the exercise we learnt how to create sequence and communication diagrams, and we learnt how to convert a sequence diagram to a communication diagram, and learnt the importance of both the diagrams in the software development process, as with the wo diagrams, we were able to get a better understanding of our project, which will make it easier for us to implement it in code.

EX - 6

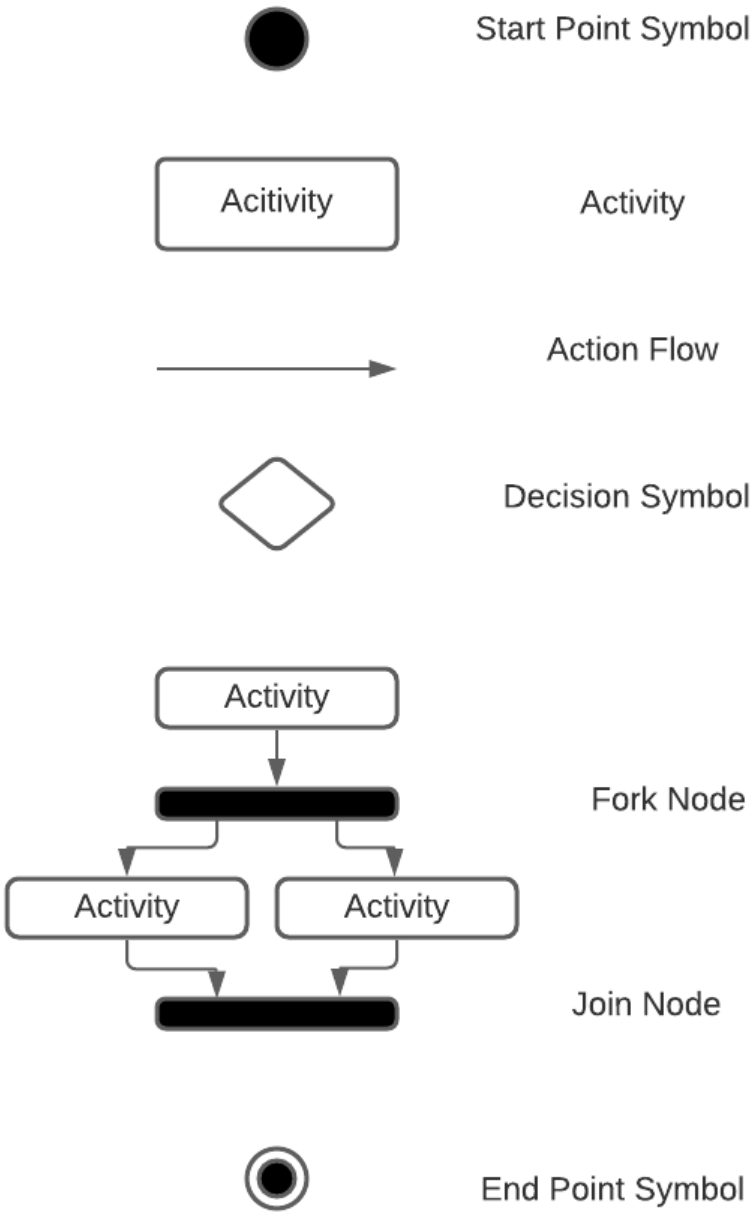
STATE MACHINE AND ACTIVITY DIAGRAM

Aim: To design a state machine and activity diagram for CCPS.

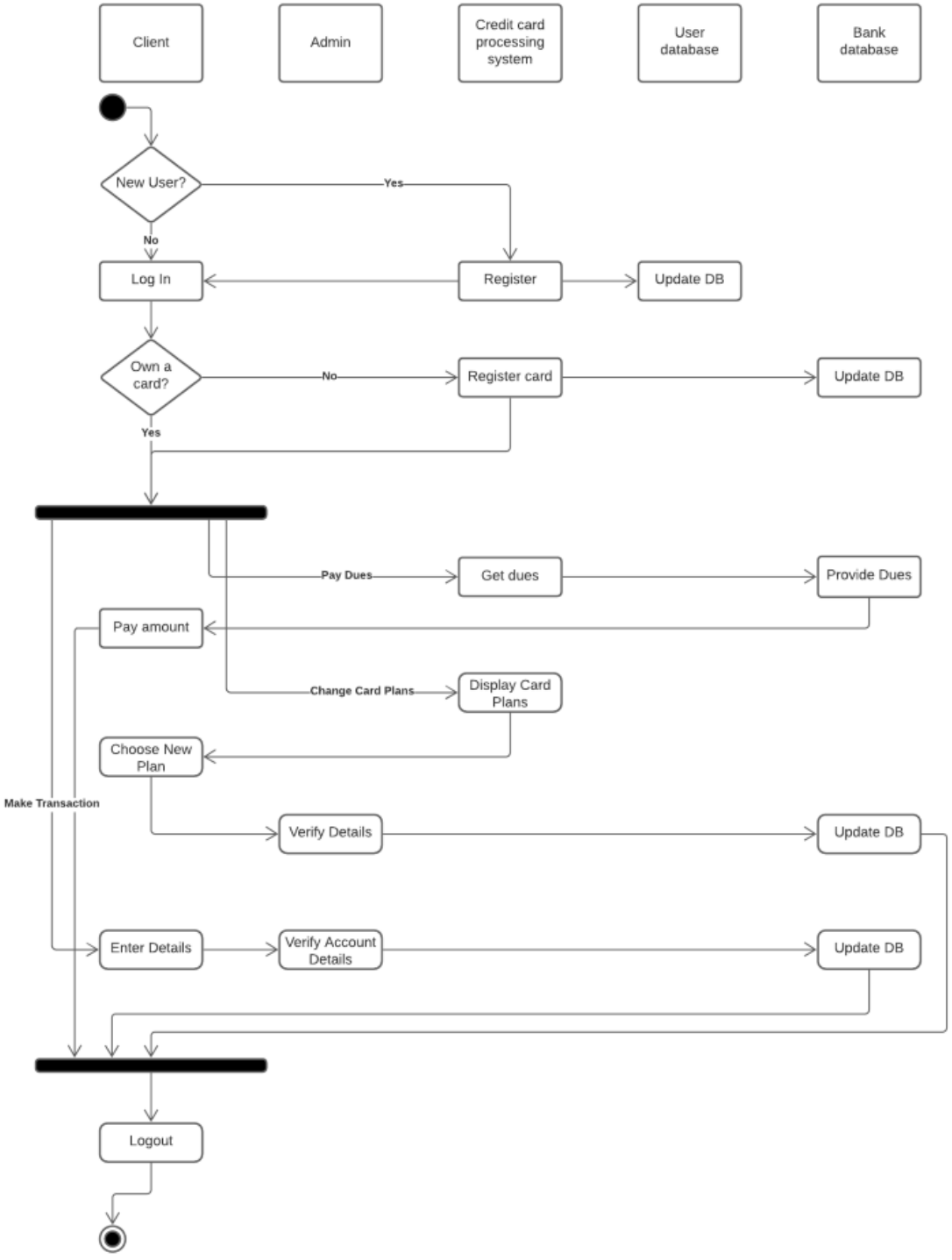
Identification of States:

1. Log In page
2. Admin
3. Create Account
4. Home Page
5. Apply
6. View Cards
7. View Plans
8. Transaction
9. Terminate
10. Change Plan

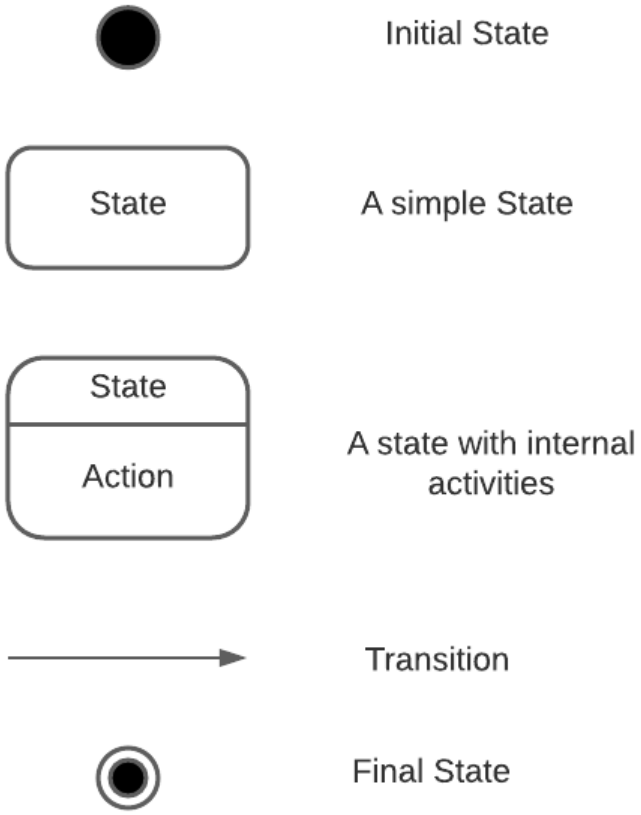
UML notations for activity diagram:



Activity Diagram:



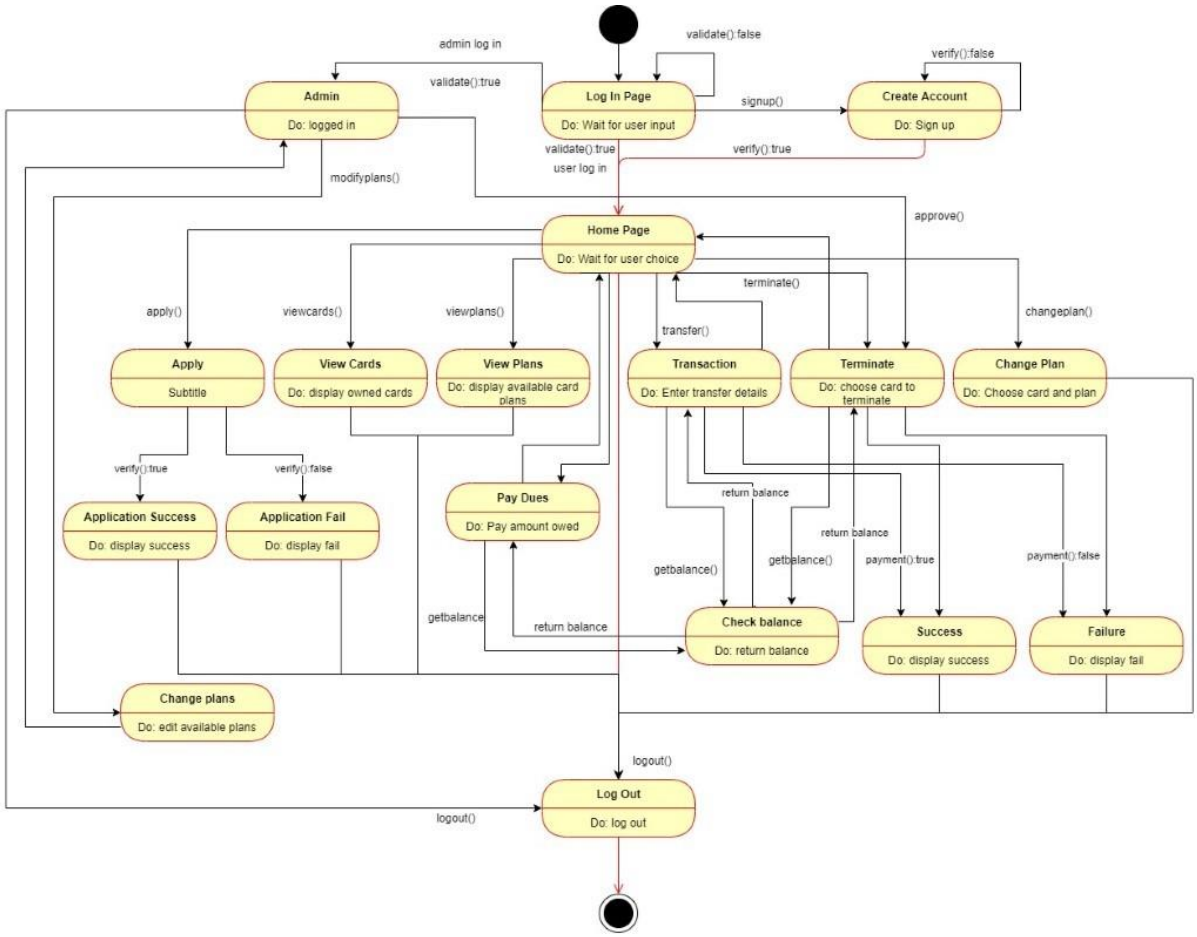
UML notations for the State Machine Diagram:



Identification of states:

- Log in page
- Create account
- Admin
- Home page
- Apply
- View cards
- View plans
- Transaction
- Terminate
- Change plan
- Application successful
- Application fails
- Pay dues
- Check balance
- Success
- Failure
- Change plans
- Log out

State Machine Diagram:



Documentation:

The State Machine diagram defines the dynamic behaviour of an individual object and the activity diagram describes the workflow behaviour and process flows in the CCPS. This model helps in implementation of the entire system.

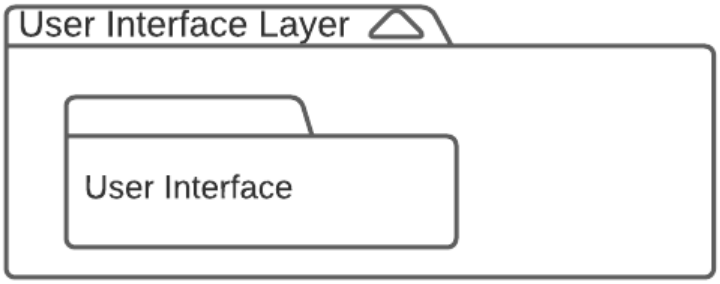
EX - 7

PACKAGE DIAGRAM

Aim: To draw a package diagram for the CCPS.

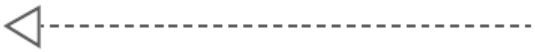
UML notations for package diagram:

1. **Package**



2. **Model**

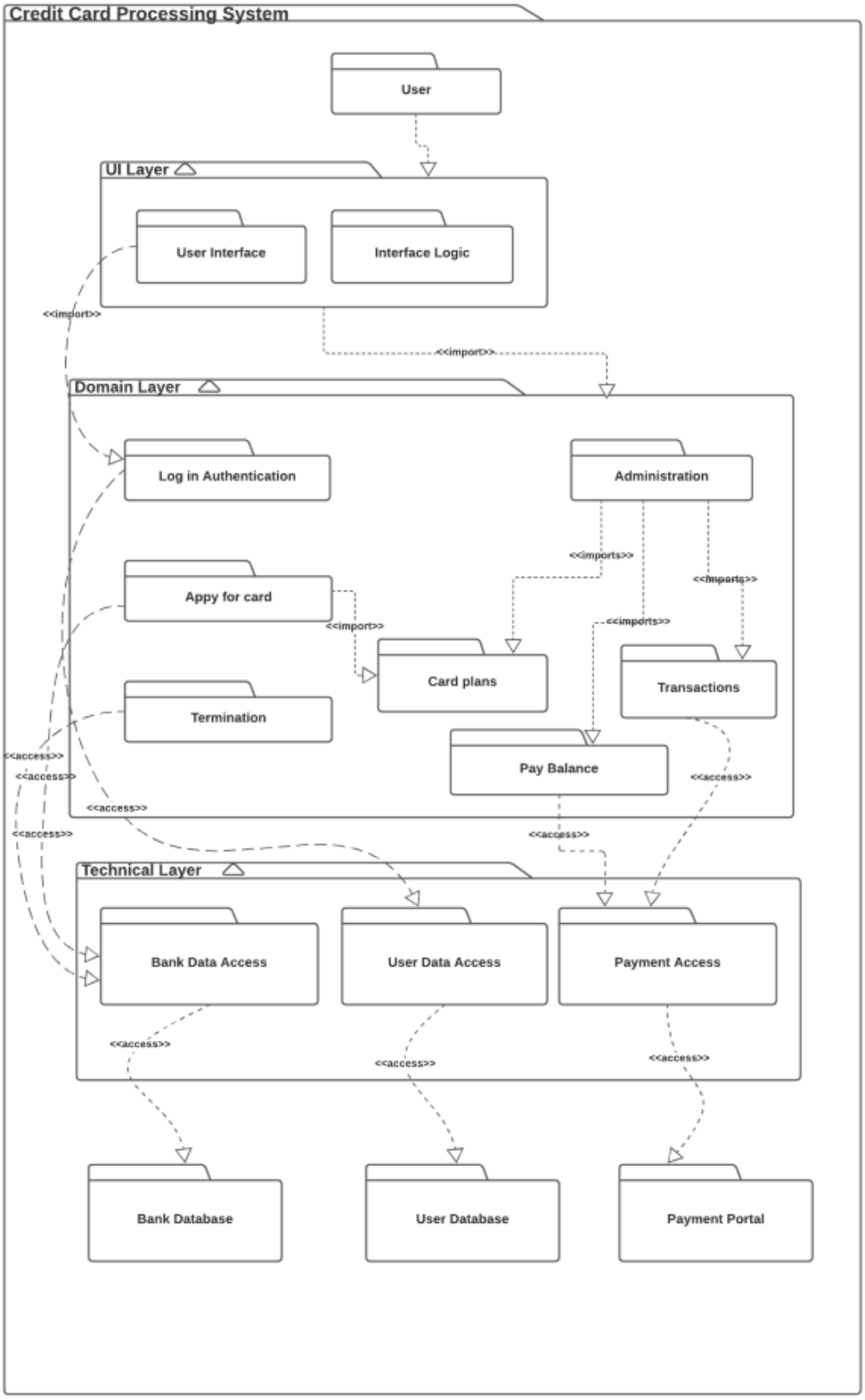
3. **Dependency**



4. **Import**

5. **Access**

Package Diagram:


Credit Card Processing System
User
UI Layer
User Interface
Interface Logic
<<import>>
<<import>>
Domain Layer
Log in Authentication
Administration
<<imports>>
<<imports>>
Appy for card
<<import>>
<<imports>>
Card plans
Transactions
Termination
Pay Balance
<<access>>
<<access>>
<<access>>
<<access>>
<<access>>
<<access>>
Technical Layer
Bank Data Access
User Data Access
Payment Access
<<access>>
<<access>>
<<access>>
Bank Database
User Database
Payment Portal

Documentation:

The package diagram depicts the dependencies between the packages of the CCPS. It also shows both the structures and the dependencies between sub-systems or modules. The model helps in implementation of the entire system as it breaks down the dependencies to a simpler level.

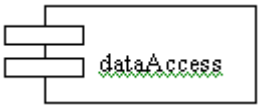
EX - 8

Component and Deployment Diagram : Credit Card Processing System

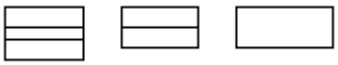
Aim: To understand and design the Component and Deployment Diagram for the Credit Card Processing System.

UML Notations For Component Diagram:

1. Component

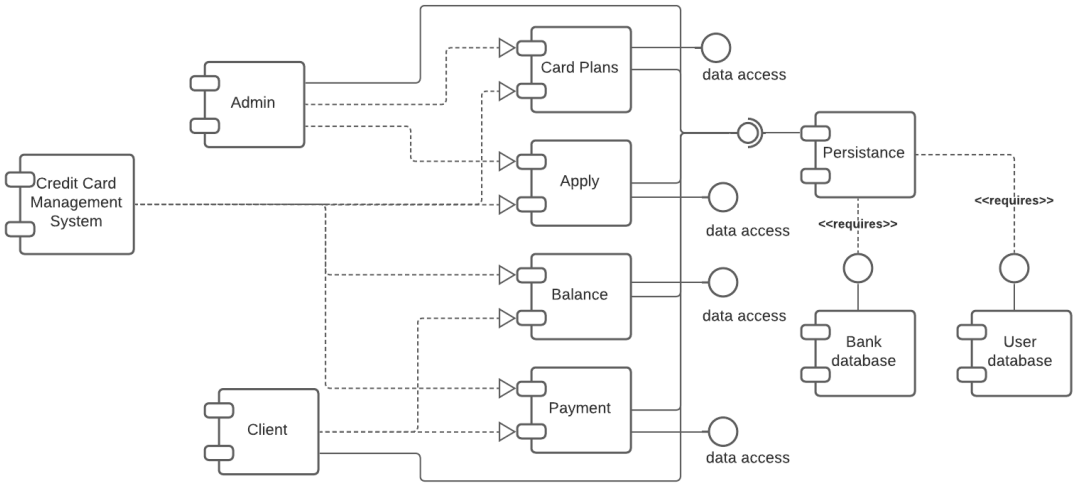


2. Class/Interface/Object



3. Relation/Association

Component Diagram :



List of Identified Nodes from Component Diagram:

1. Credit card management system
2. Admin
3. Client
4. Card plans

5. Apply
6. Balance
7. Payment
8. Persistence
9. Bank database
10. User database

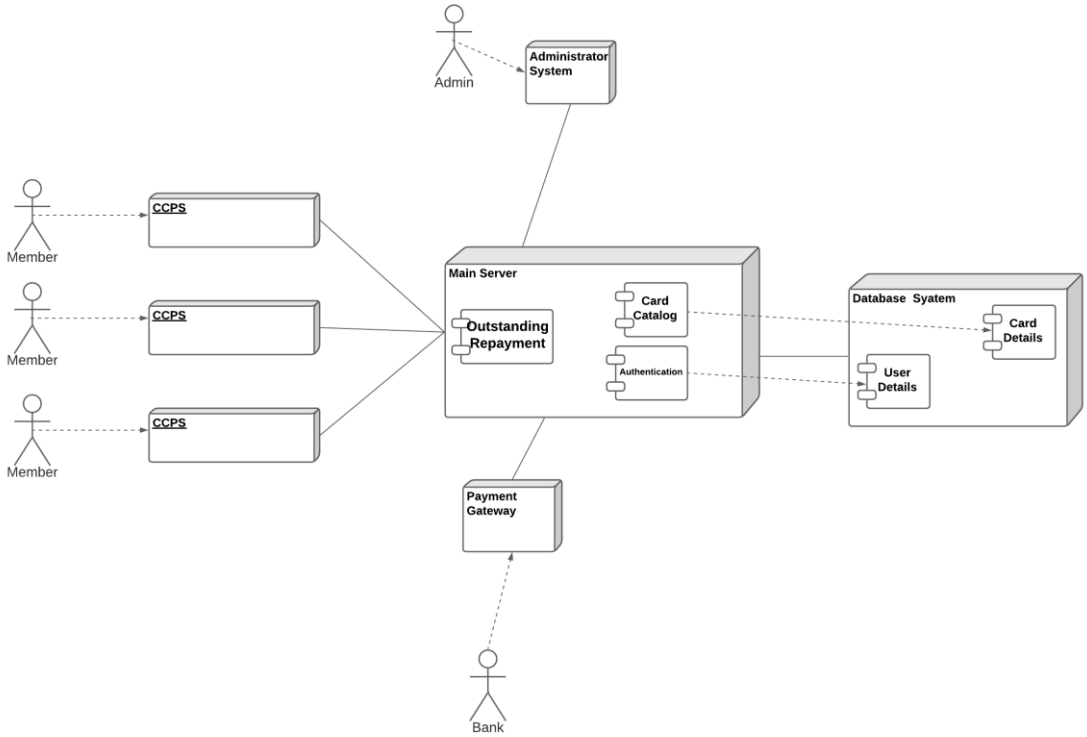
UML Notations For Deployment Diagram:

1.Node:

2.Communication Path:

3.Entity:

Deployment Diagram:



List of identified nodes from Deployment Diagram:

1.CCPS

2.Main Sever

3.Payment Gateway

4.Administrator System

5.Database System

Documentation:

From this exercise we learn how to efficiently categorize which component

can access what data making it easier during implementation. The deployment

diagram also provides a clear representation of the flow of processes that take place within the CCPS.