B.M.S. COLLEGE OF ENGINEERING BENGALURU
Autonomous Institute, Affiliated to VTU

Lab Record

Object Oriented Analysis and Design

Submitted in partial fulfillment for the 6th Semester Laboratory

Bachelor of Technology
in
Computer Science and Engineering

Submitted by:

TONY J MATHEW

1BM17CS119

Department of Computer Science and Engineering
B.M.S. College of Engineering
Bull Temple Road, Basavanagudi, Bangalore 560 019
Jan-May 2019

B.M.S. COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Object-Oriented Analysis and Design(16CS6DCOOM) laboratory has been carried out by TONY J MATHEW (1BM17CS119) during the 6th Semester Jan-May-2019.

Signature of the Faculty Incharge:

NAME OF THE FACULTY: REKHA G.S

Department of Computer Science and Engineering
B.M.S. College of Engineering, Bangalore

Table of Contents

1. College Information System

Problem Statement:

To design a College Information System to simplify the process of course registration. The system will allow students to register their courses online. The information is directly relayed to the college database and eliminates unnecessary paperwork, travelling and time consumption.

SRS:

Purpose:

The College Management System is an online system. The system is developed to facilitate more streamlined course registration process. The system allows students to register their courses online. The information is directly relayed to the college database and reduces paperwork, travelling and time consumption. Details of courses such as syllabus, credits and prescribed learning material can also be checked.

Requirements:

Functional Requirements:

- Provide Student Login:
  Students can login and register for courses within prescribed time limits.
- Provide Course Information:
  Course Details can be checked before and after registration. Data such as syllabus, credits and learning material are available for viewing.
- Provide Admin Login:
  Admin login is required to update available courses as well as course details.
- GUI:
  An interface is developed for above mentioned functional requirements.

Non-Functional Requirements:

- Cost:
  The software should be cheap and within budget.
- Time:
  The software product should be completed within deadlines
- Security:
  The data should be secure

Class Diagram:

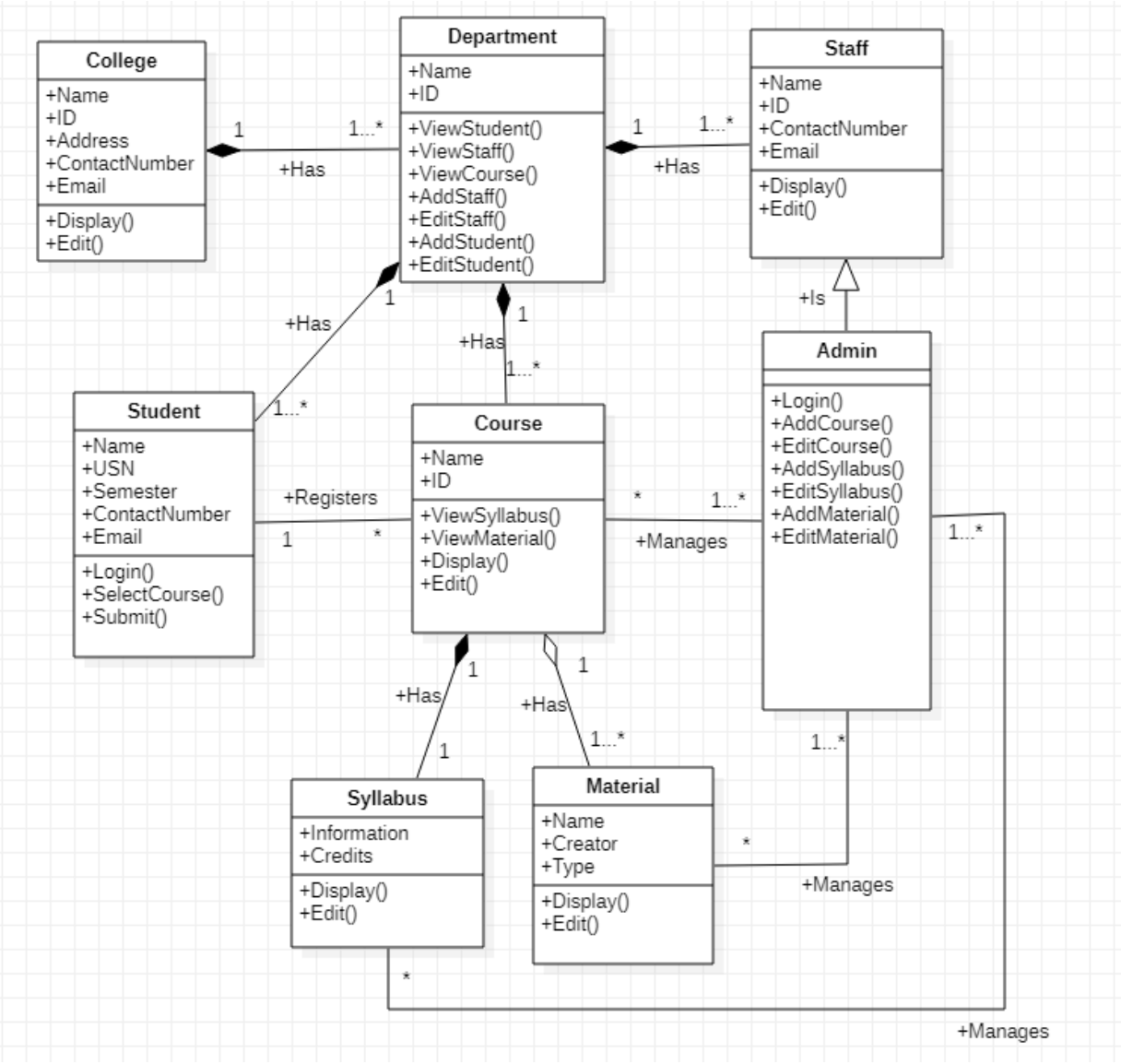

Fig 1.1

- Admin: Manages the available courses.
- Staff: Contains details of all staff working for the departments.
- Departments: Contains information about departments in the college and manages staff.
- College: Contains information about college.
- Student: Contains information about student and allows them to register for courses.
- Course: Contains information about courses.
- Syllabus: Contains information about syllabus of a course.
- Material Contains information about material of a course.

State Diagram:

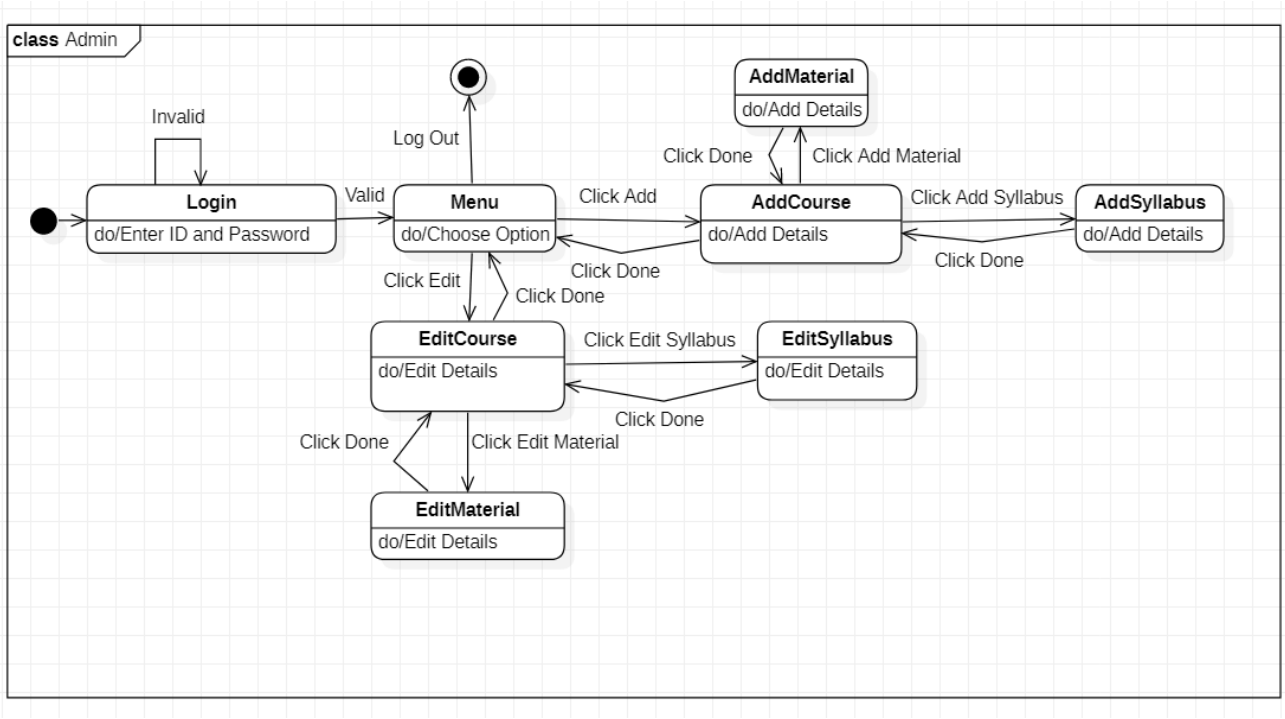

Fig 1.2

The state diagram shows the course management options available to the admin.

First, they need to login. If valid, they can access the menu. From the menu, they can add courses or edit existing courses along with syllabus and material.

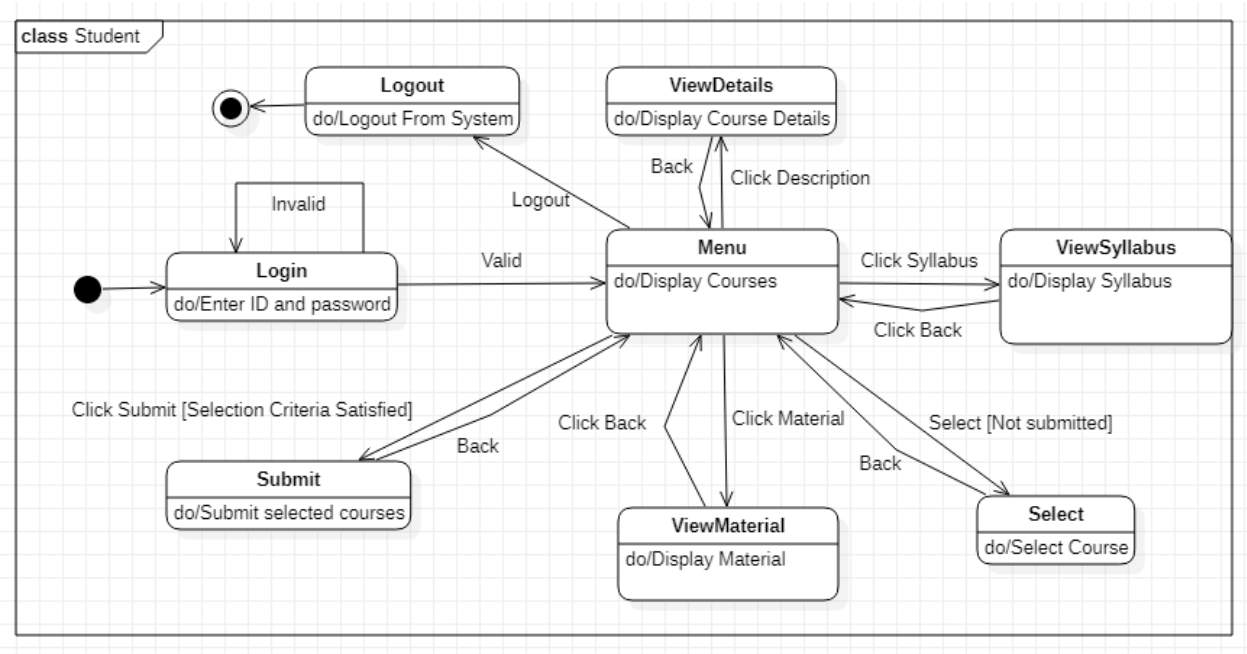

Fig 1.3

The state diagram shows the course registration options available to student.

First, they need to login. If valid, they can access the menu. They can view course details, syllabus and materials. They can select courses and submit their selections.

Use Case Diagrams:

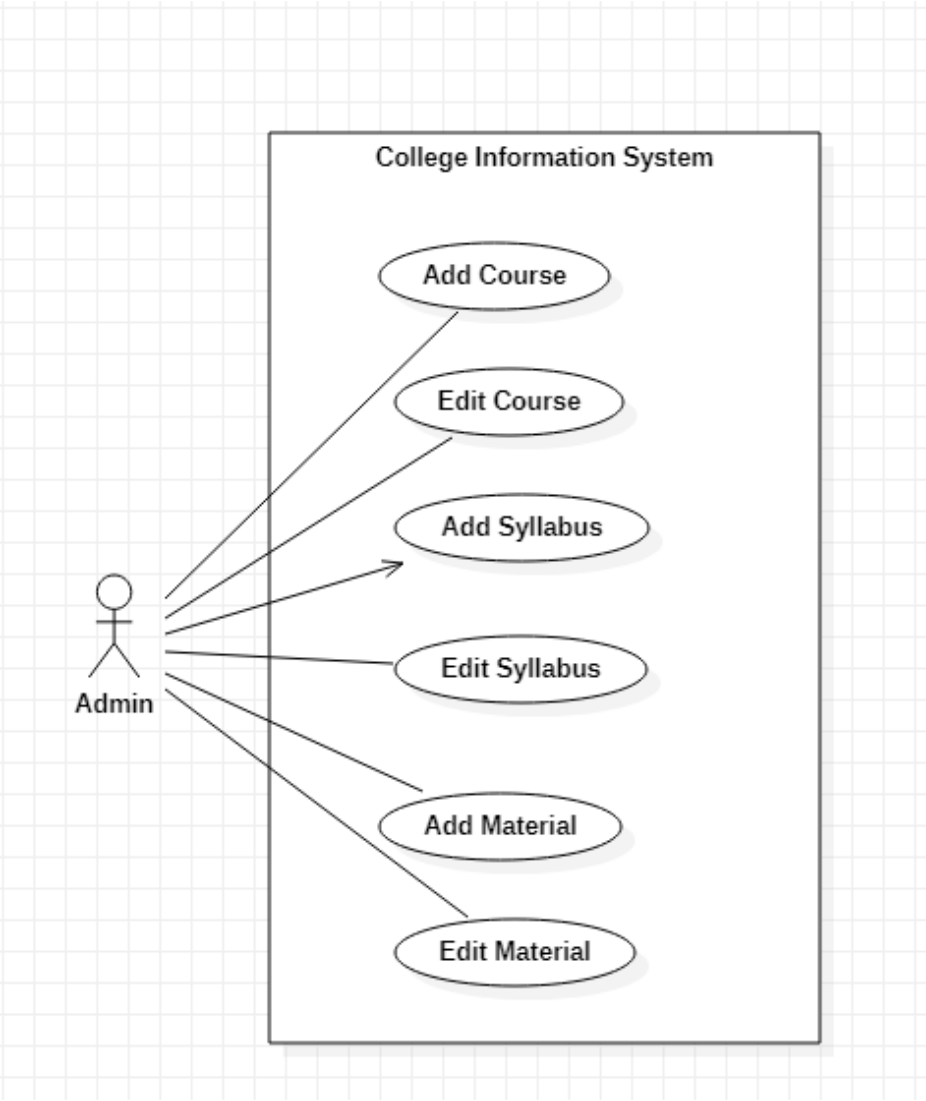

Fig 1.4

There is one actor:

- **Admin:** Person who manages the available courses

There are six use cases:

- **Add Course:** Add courses that can be chosen
- **Edit Course:** Edit courses that can be chosen
- **Add Syllabus:** Add syllabus to course
- **Edit Syllabus:** Edit syllabus of course
- **Add Material:** Add material to course
- **Edit Material:** Edit material of course

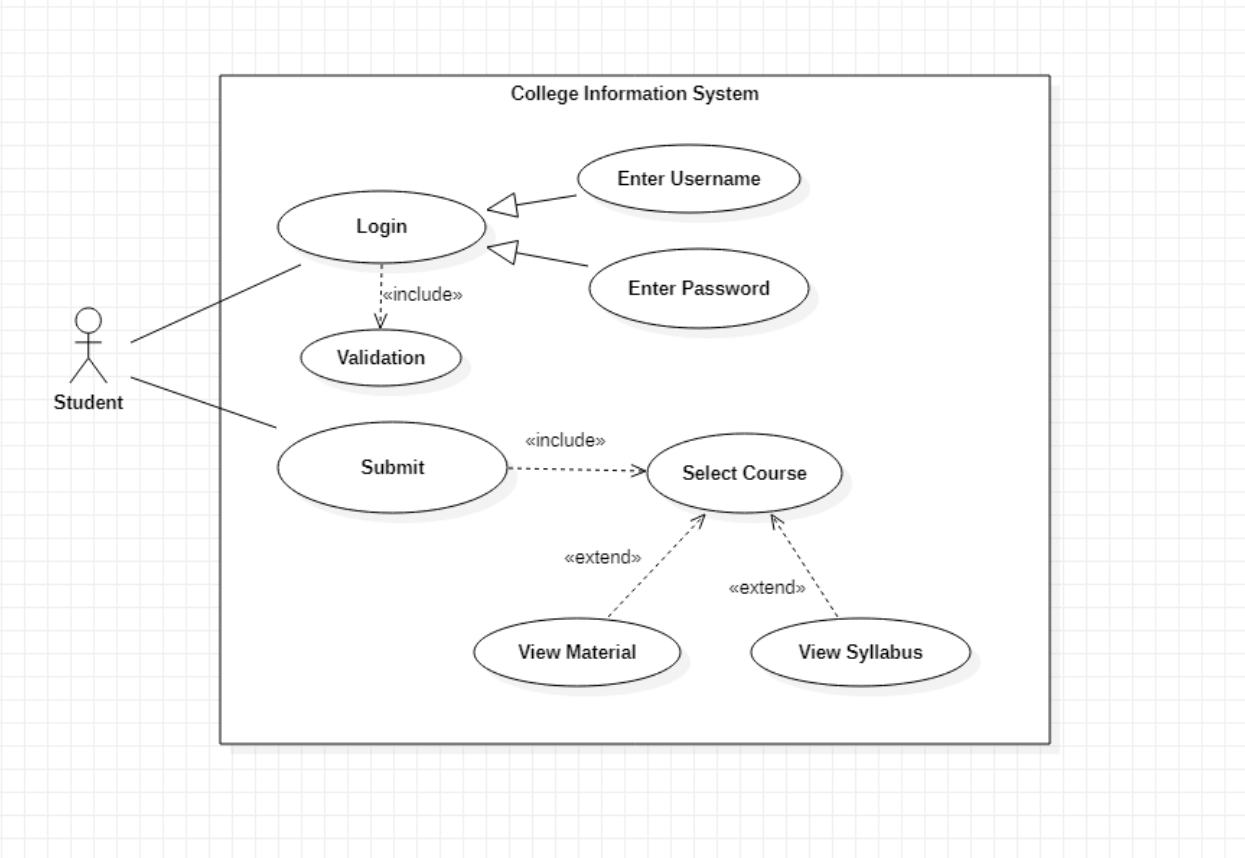

Fig 1.5

There is one actor:

- **Student:** Person who registers for courses

There are eight use cases:

- **Login:** Login to course registration website
- **Enter Username:** Enter username to login
- **Enter Password:** Enter password to login
- **Validation:** Validate login credentials and allow login
- **Submit:** Submit selected courses to college database
- **Select Course:** Select course from available courses
- **View Syllabus:** View syllabus of course
- **View Material:** View material of course

Sequence Diagrams:

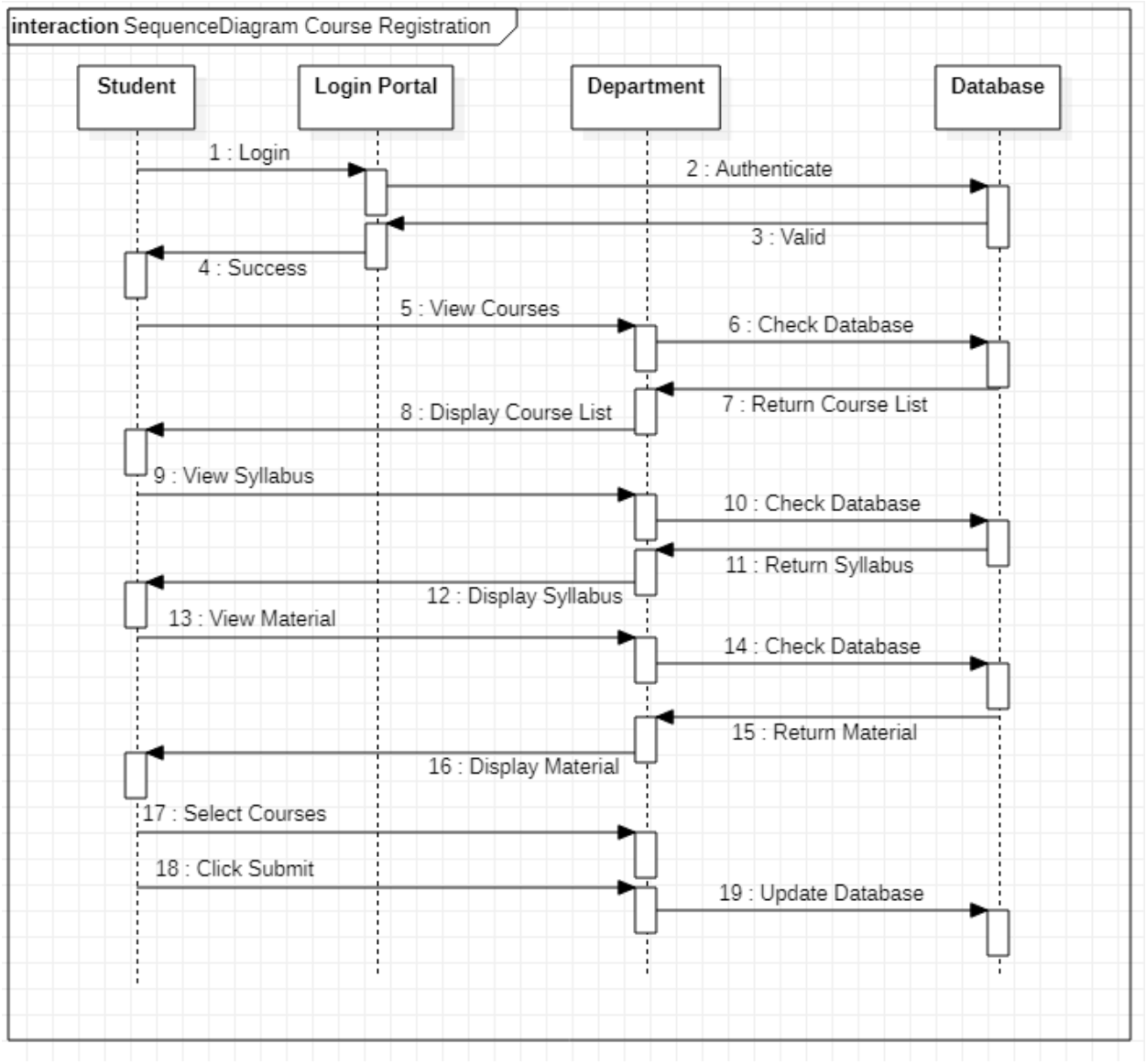

Fig 1.6

The sequence diagram shows the interaction of student with system regarding course registration process.

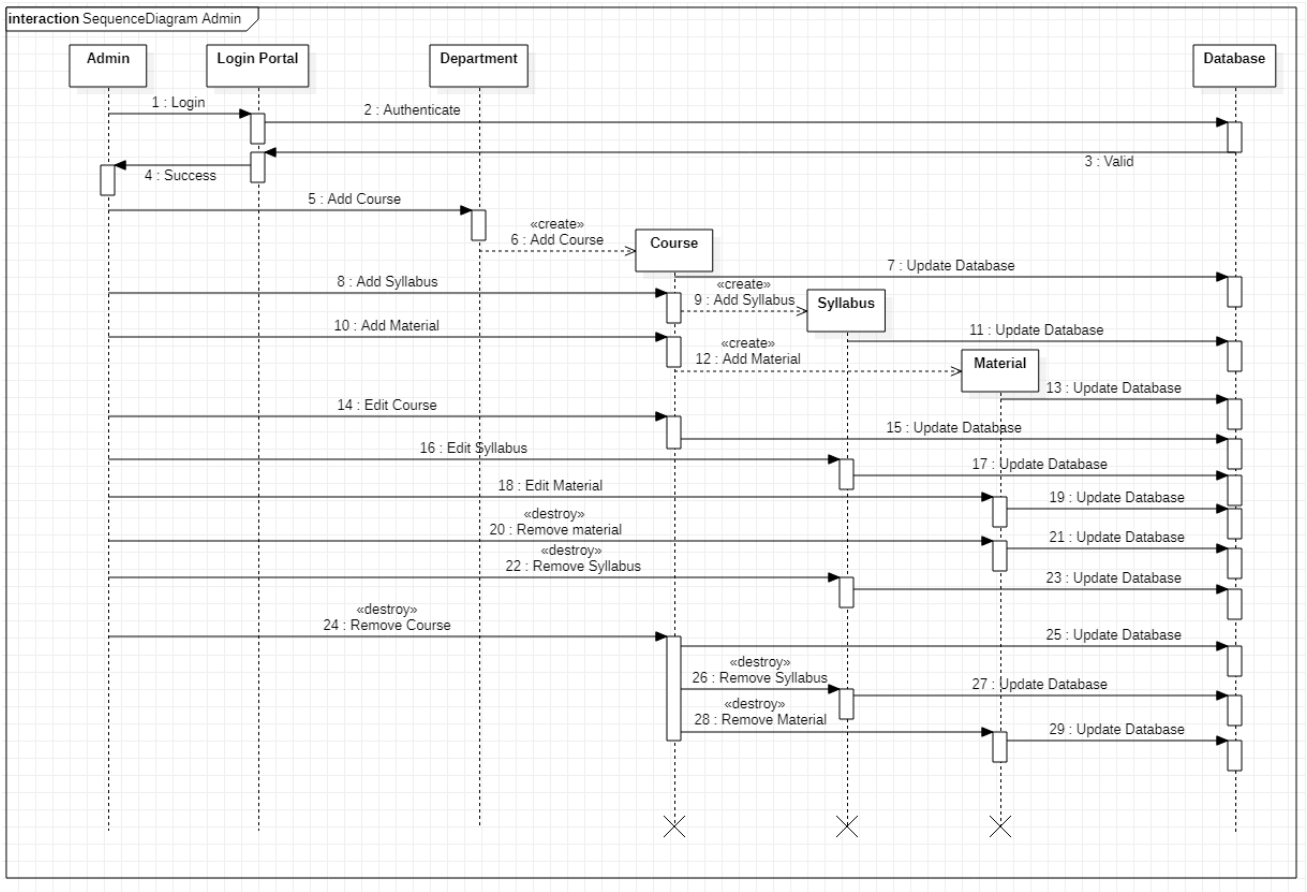

Fig 1.7

The sequence diagram shows the interaction of Admin with system regarding management of available courses, syllabus and materials.

Activity Diagrams:

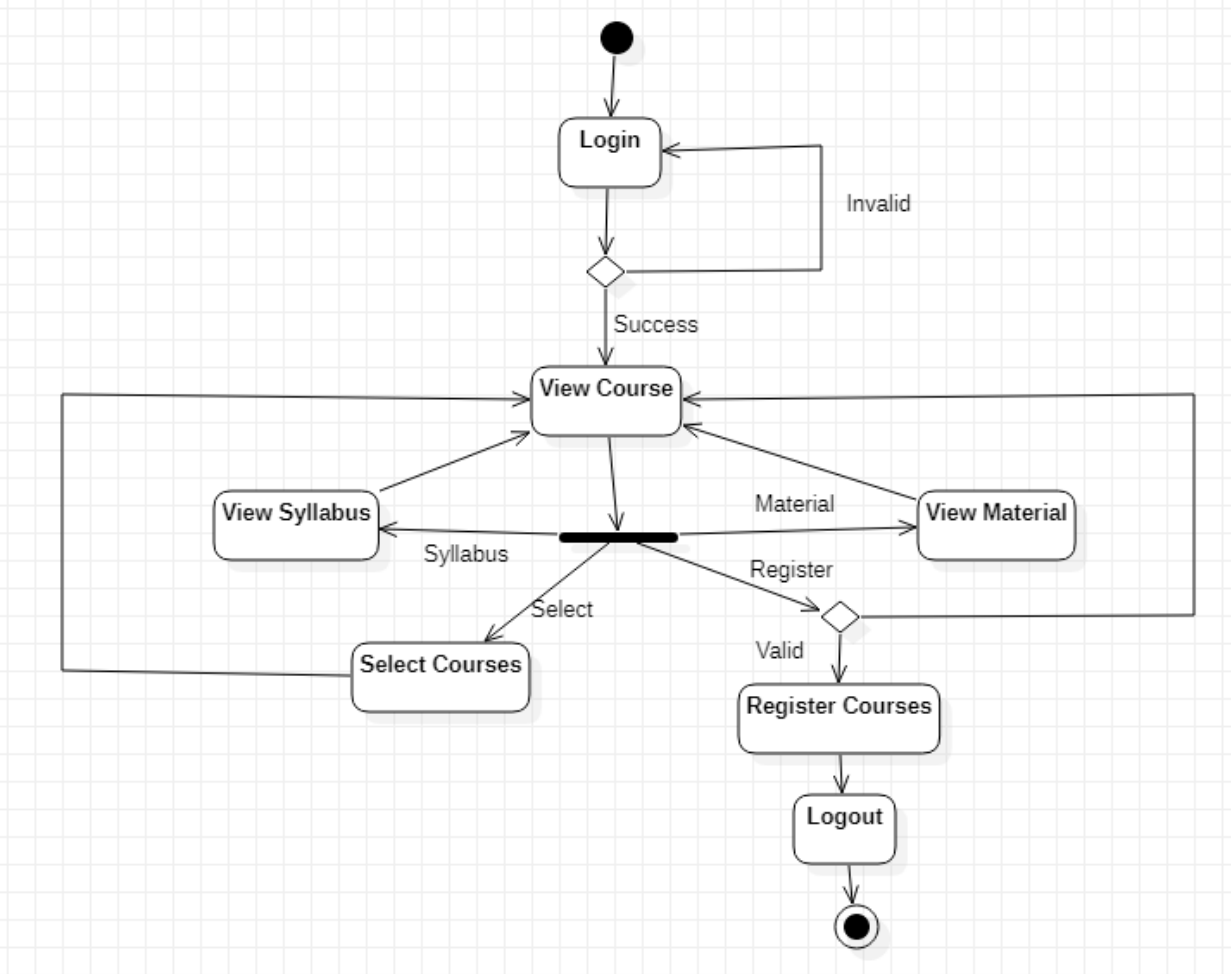

Fig 1.8

The activity diagram shows the activities involved in course registration performed by user.

First they need to login, then they can view courses, syllabus and material.

They can select course and then register them.

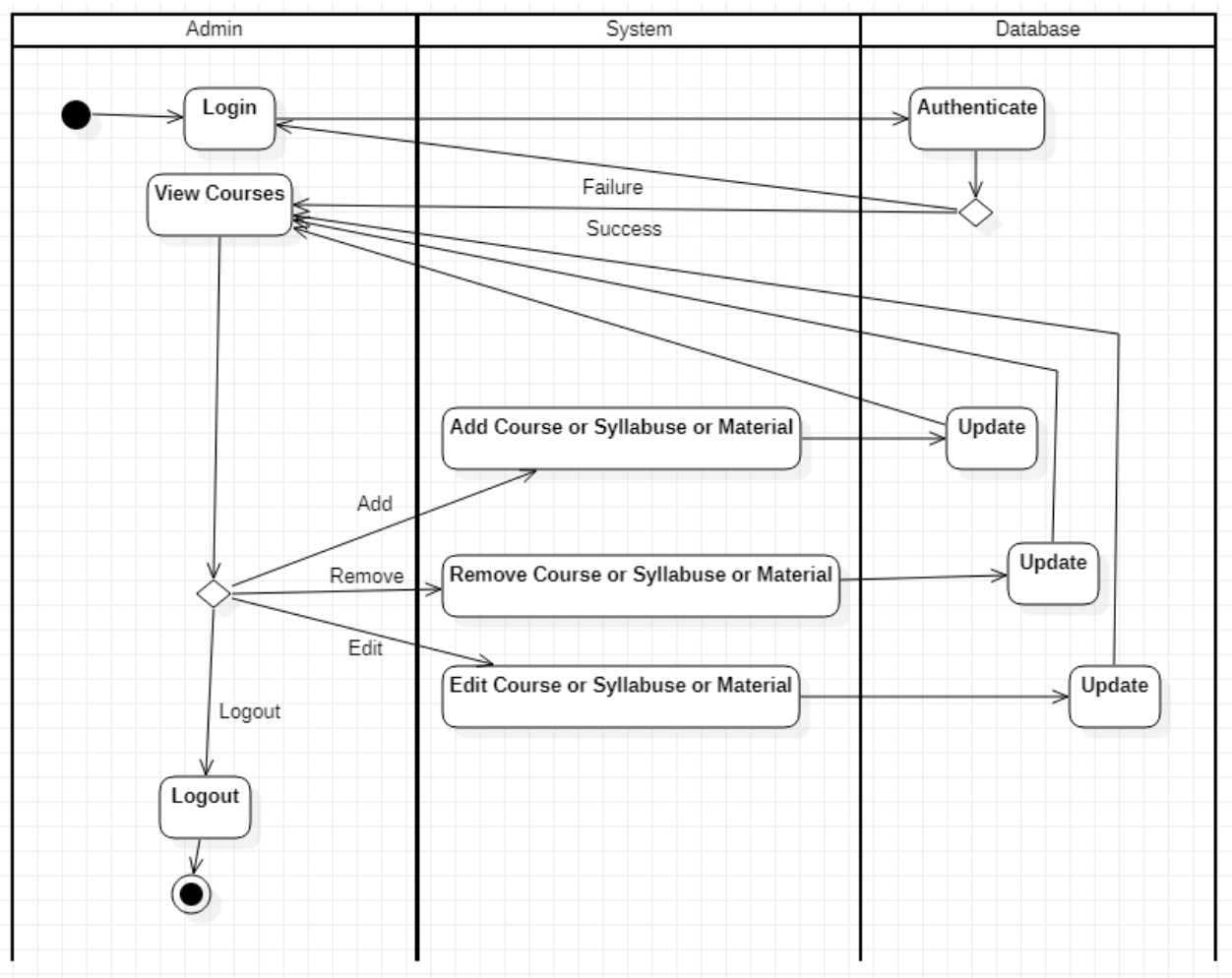

Fig 1.9

The advanced activity diagram shows the activities involved using swim lanes.

The swim lanes provide information about the persons responsible for the activities in the system.

The respective persons activities are shown under their swim lanes. In the diagram shown, admin, system and database are the swim lanes who perform the activities shown in the diagram.

2. Hostel Management System

Problem Statement:

To design a Hostel Management System to manage various activities related to hostel services. The system will allow students as well as college administrators to interact with hostel services in a streamlined manner with user-friendly GUI. The information is directly stored in the college database and eliminates unnecessary paperwork, travelling and time consumption.

SRS:

Purpose:

The Hostel Management System is an online system. The system is developed to facilitate more streamlined hostel management process. The system allows students to request a room and pay hostel fees online. Administrators can get information regarding students staying at hostel as well as collect fees, set canteen menu and security patrol routes. The information is directly stored in the college database and reduces paperwork, travelling and time consumption.

Requirements:

Functional Requirements:

- Allow Student to request hostel room:
  Students can login and request hostel room.
- Provide Option to Pay Fees:
  Hostel fees can be payed online through system.
- Provide Admin Login:
  Admin login is required to approve student requests as well as manage hostel services such as canteen menu and security patrol routs
- GUI:
  An interface is developed for above mentioned functional requirements.

Non-Functional Requirements:

- Cost:
  The software should be cheap and within budget.
- Time:
  The software product should be completed within deadlines
- Security:
  The data should be secure

Class Diagram:

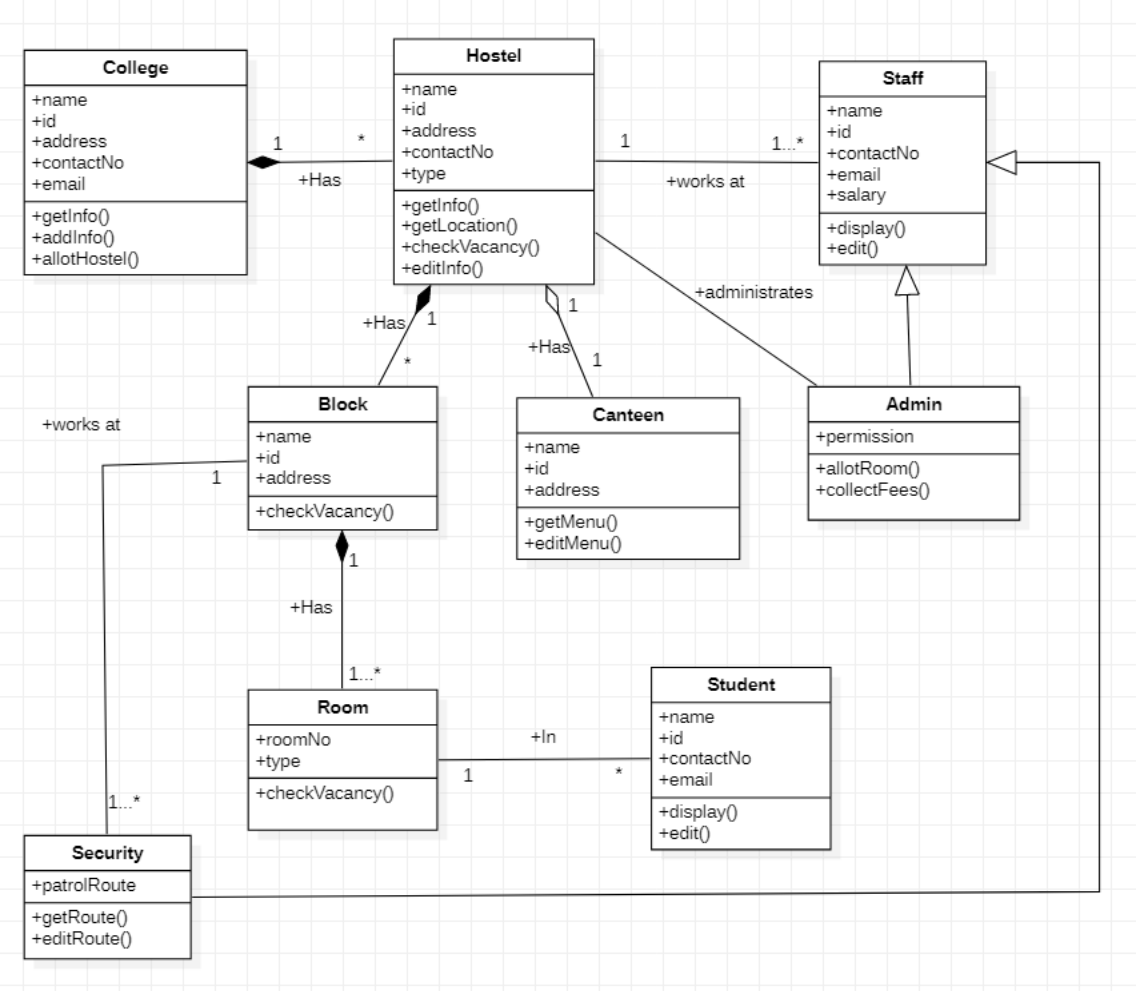

Fig 2.1

- College: Contains information regarding college and helps allot hostel.
- Hostel: Contains information about hostel and checks vacancy of rooms.
- Staff: Contains information about staff working at hostel.
- Block: Contains information about block and vacancy.
- Room: Contains information about room and vacancy.
- Student: Contains information about student staying in room
- Canteen: Contains information about canteen and allows viewing and editing of menu.
- Security: Contains information about security patrols
- Admin: Approves allotment of rooms and collects fees.

State Diagram:

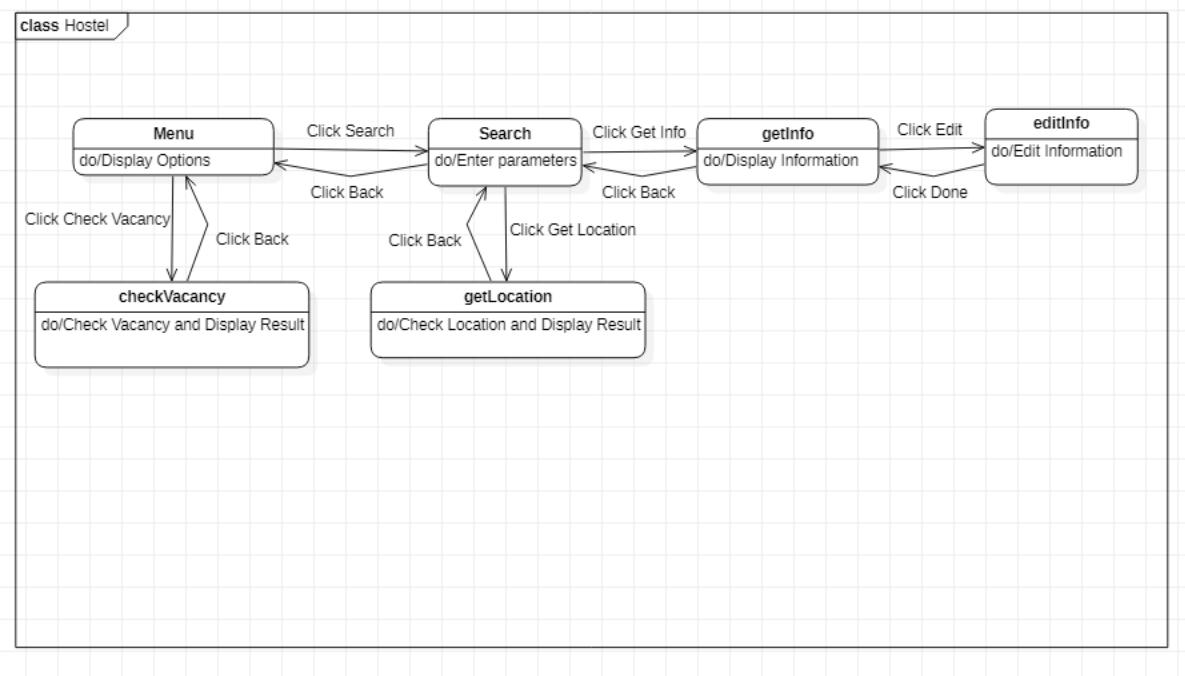

Fig 2.2

The state diagram shows the functionality of hostel class.
From the hostel menu, you can search for student information and location. You can also check if rooms are available.

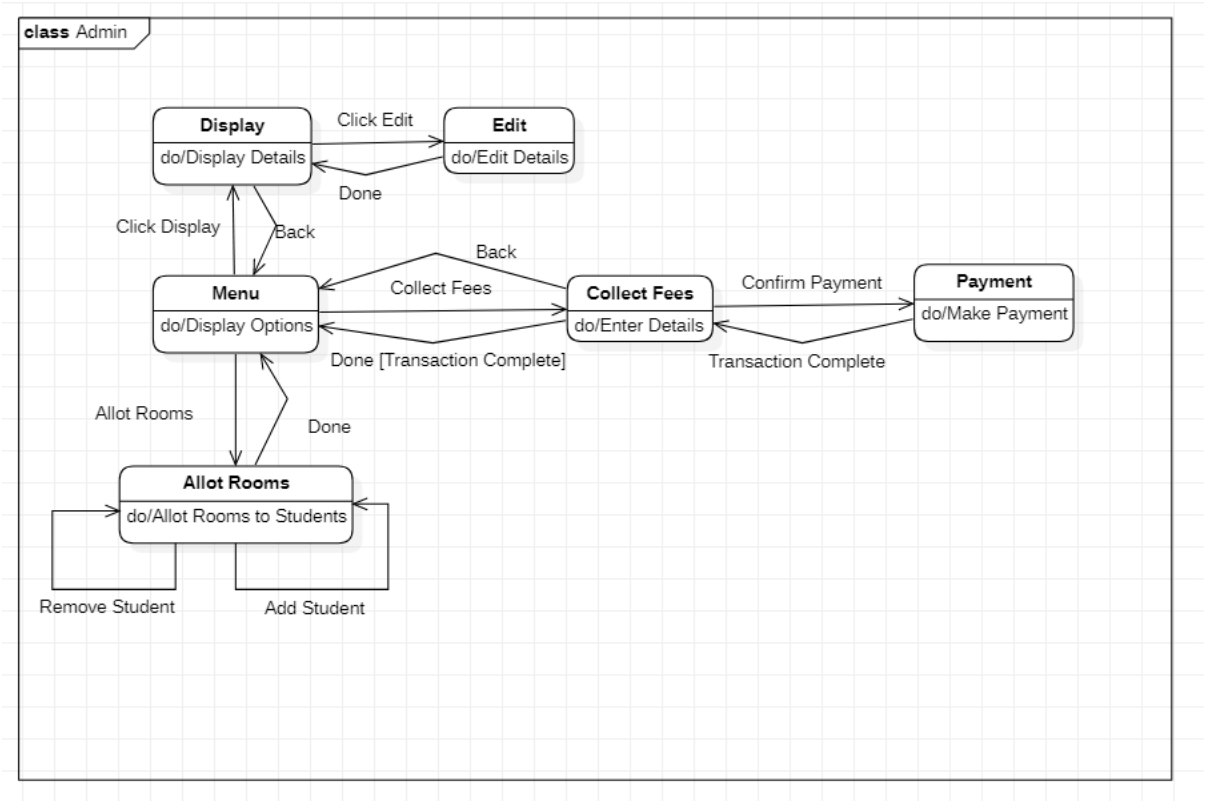

Fig 2.3

The state diagram shows the options available to admin.
From the menu. They can view and edit details, collect fees and confirm payment or allot rooms to students.

Use Case Diagrams:

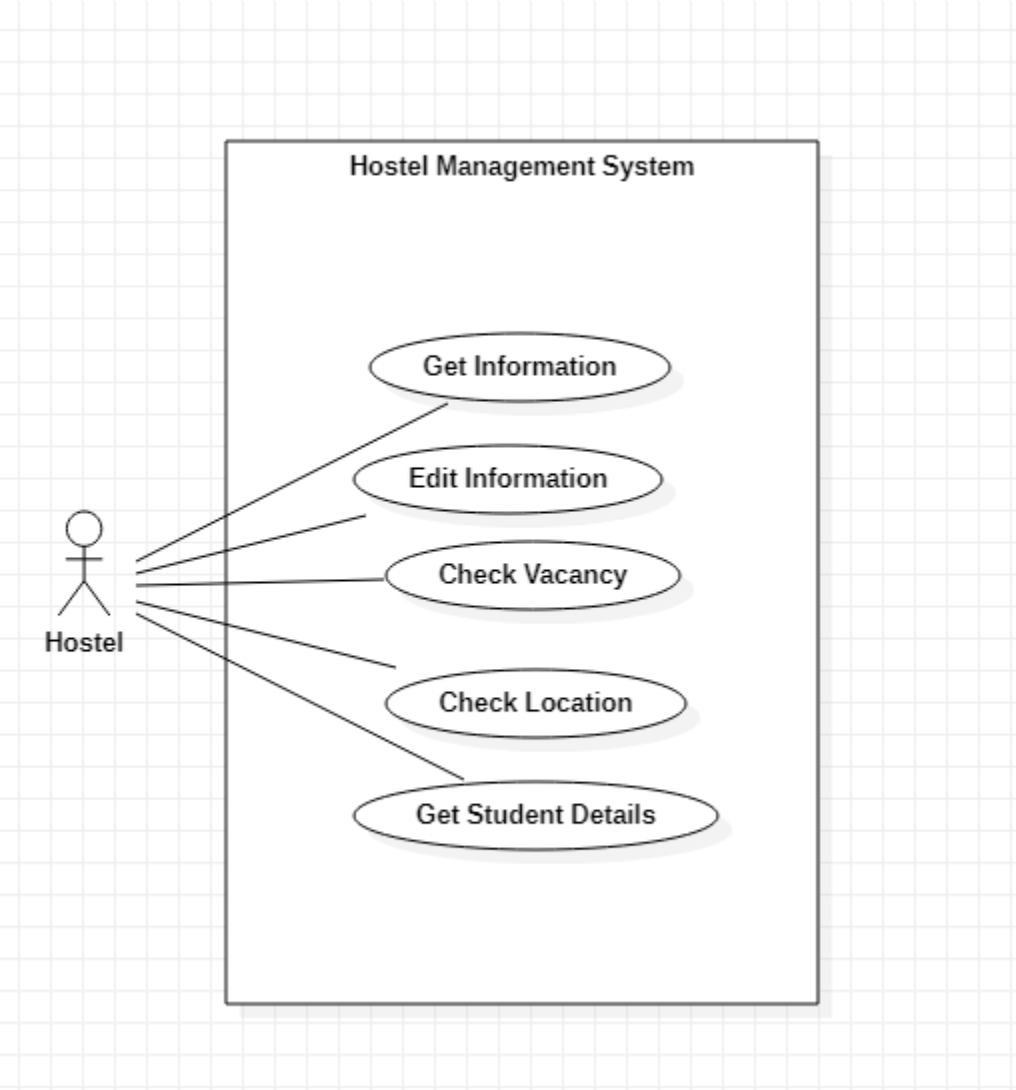

Fig 2.4

There is one actor:

- **Hostel:** The area where students are allowed to live

There are five use cases:

- **Get Information:** Get information about hostel
- **Edit Information:** Edit information about hostel
- **Check Vacancy:** Check for vacant rooms
- **Check Location:** Query location of vacant rooms
- **Get Student Details:** Get information about students residing in hostel

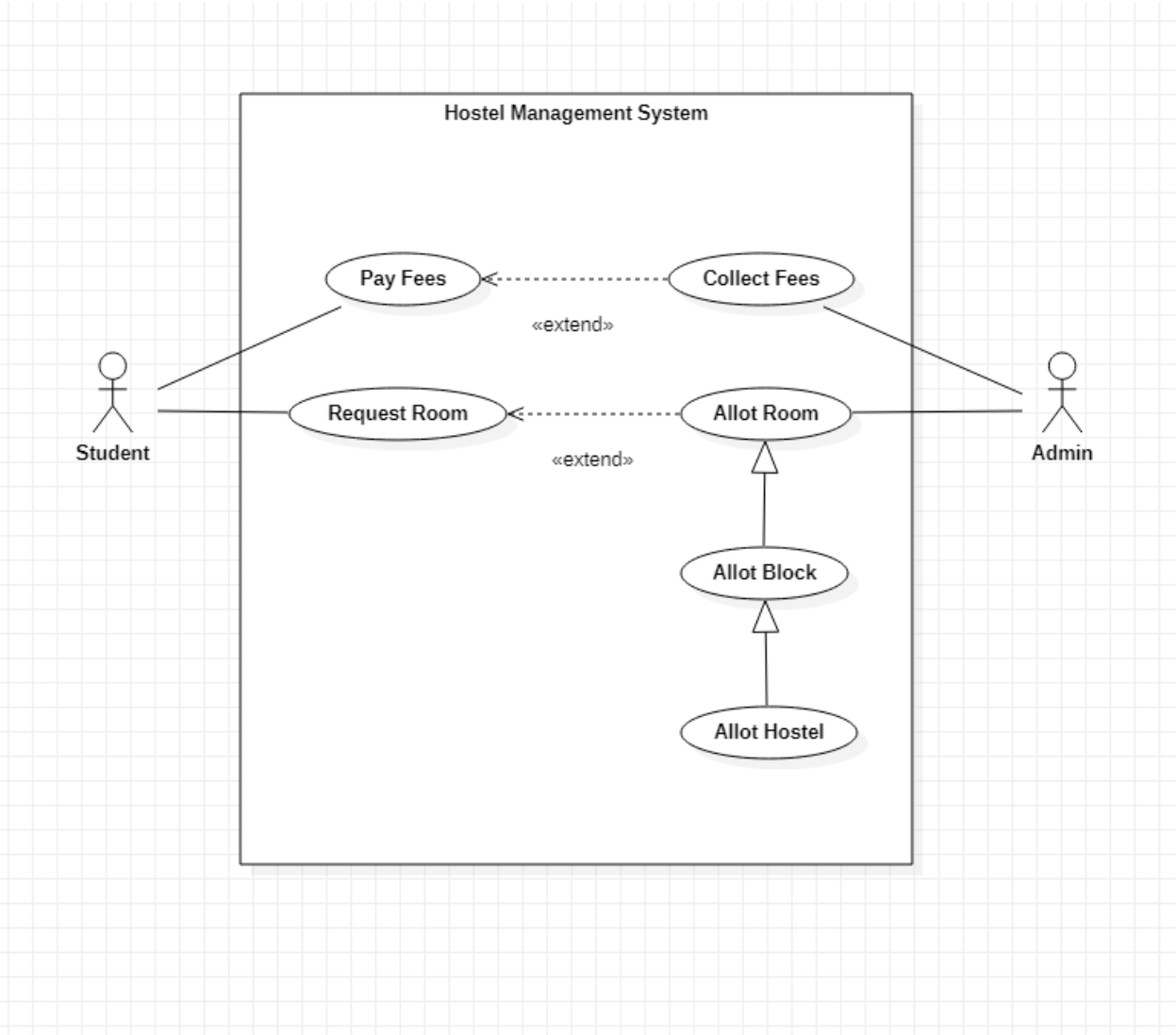

Fig 2.5

There are two actors:

- **Student:** Person who requests a room
- **Admin:** Person who allots rooms and collects fees

There are six use cases:

- **Pay Fees:** Pay fees for staying at hostel
- **Request Room:** Request hostel room
- **Collect Fees:** Collect fees for staying at hostel
- **Allot Room:** Allot room for student
- **Allot Block:** Allot block for student
- **Allot Hostel:** Allot hostel for student

Sequence Diagrams:

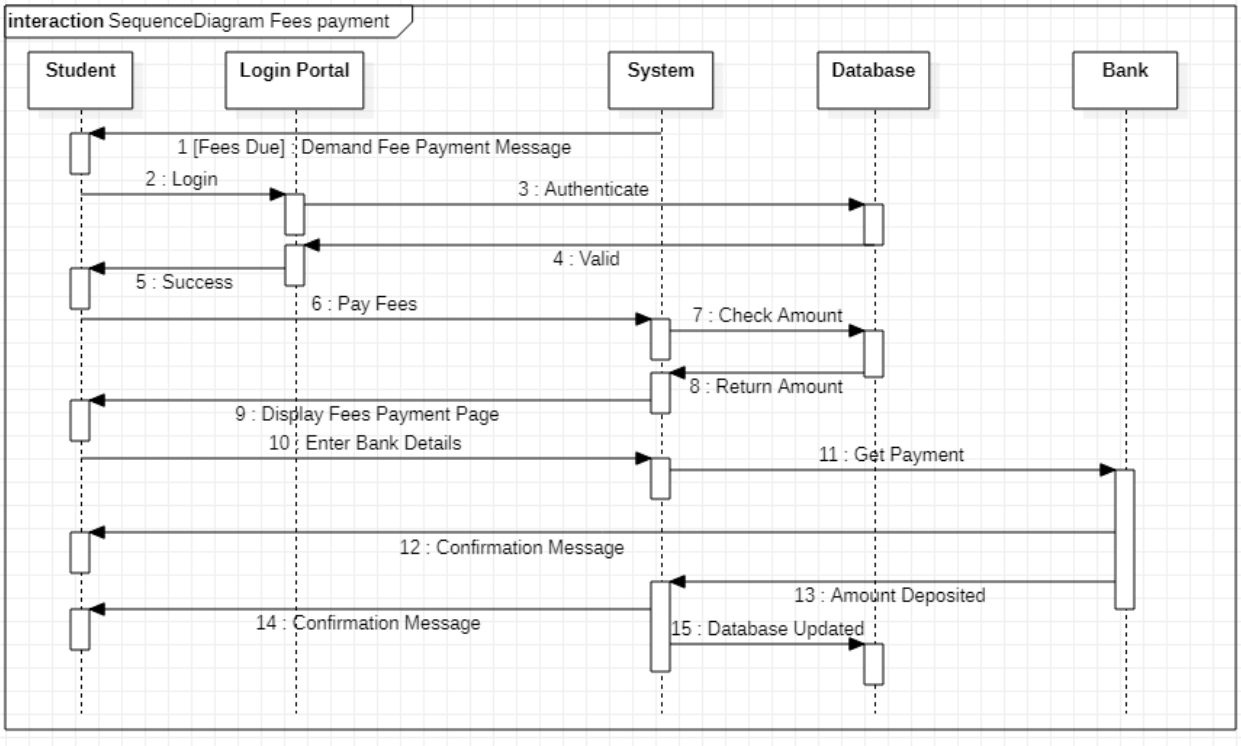

Fig 2.6

The sequence diagram shows the interaction of student with system regarding payment of hostel fees.

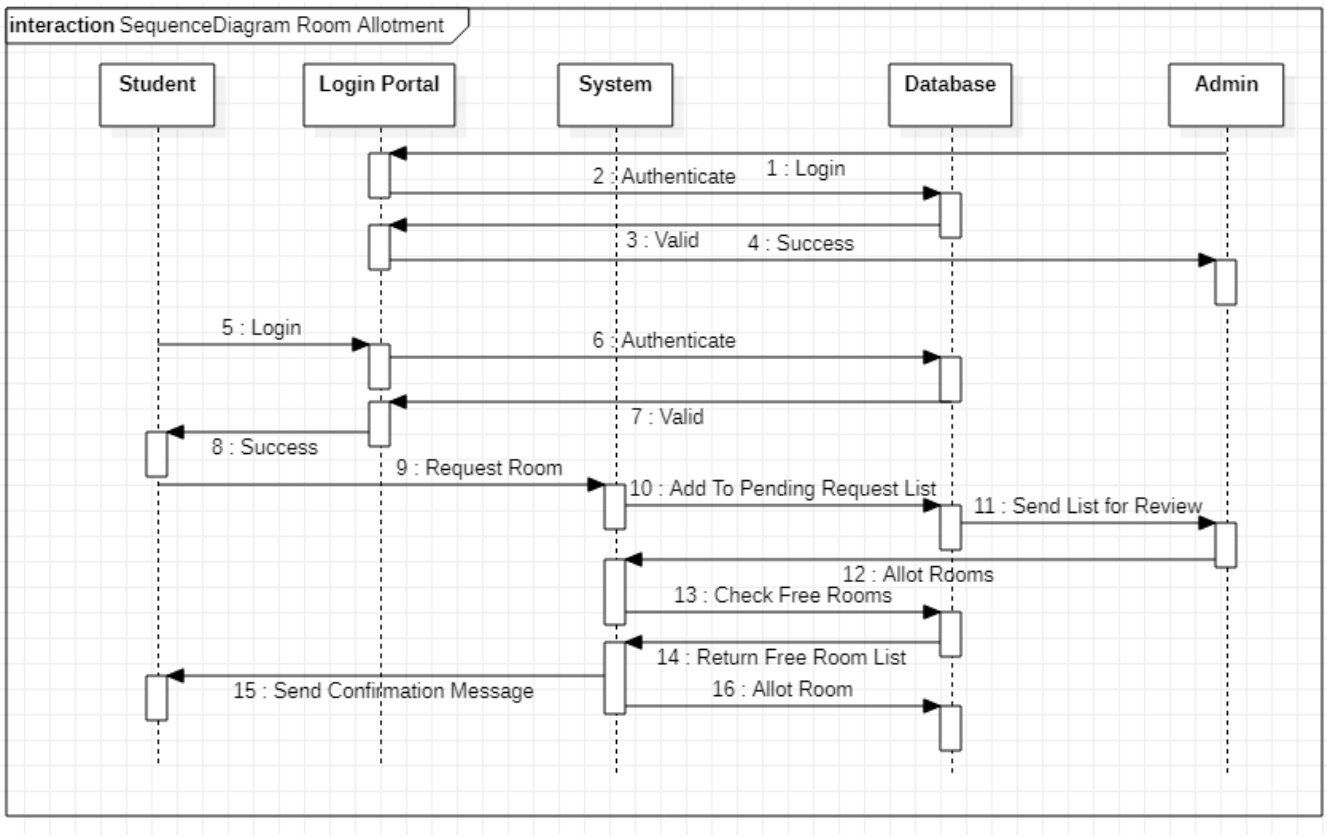

Fig 2.7

The sequence diagram shows the interaction of student and admin with system regarding allotment of rooms at hostel.

Activity Diagrams:

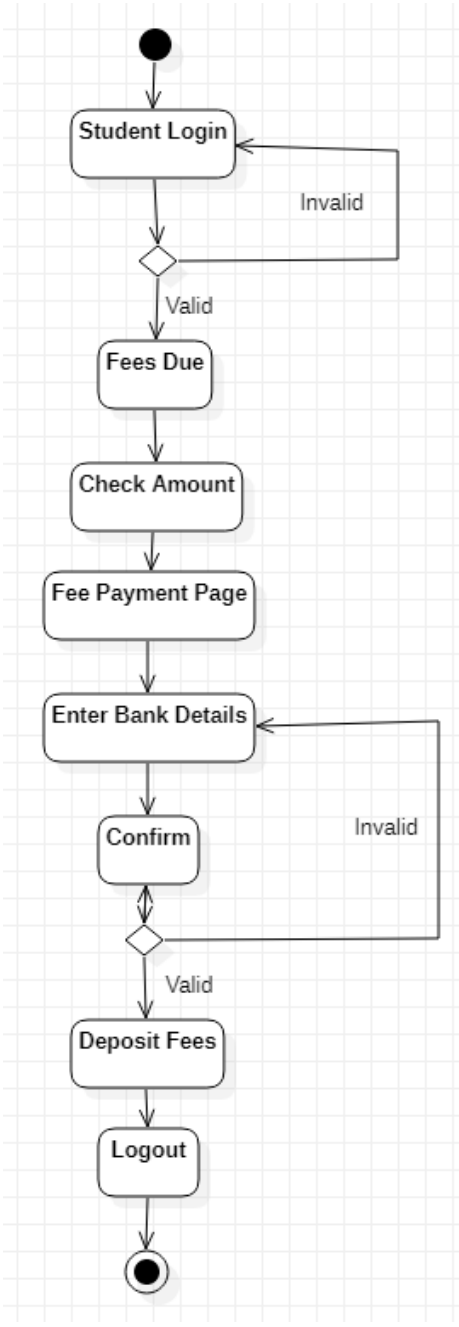

Fig 2.8

The activity diagram shows the activities involved in fees payment performed by student.

First they need to login, then they can view fees, go to payment page, enter bank details, validate and deposit fees.

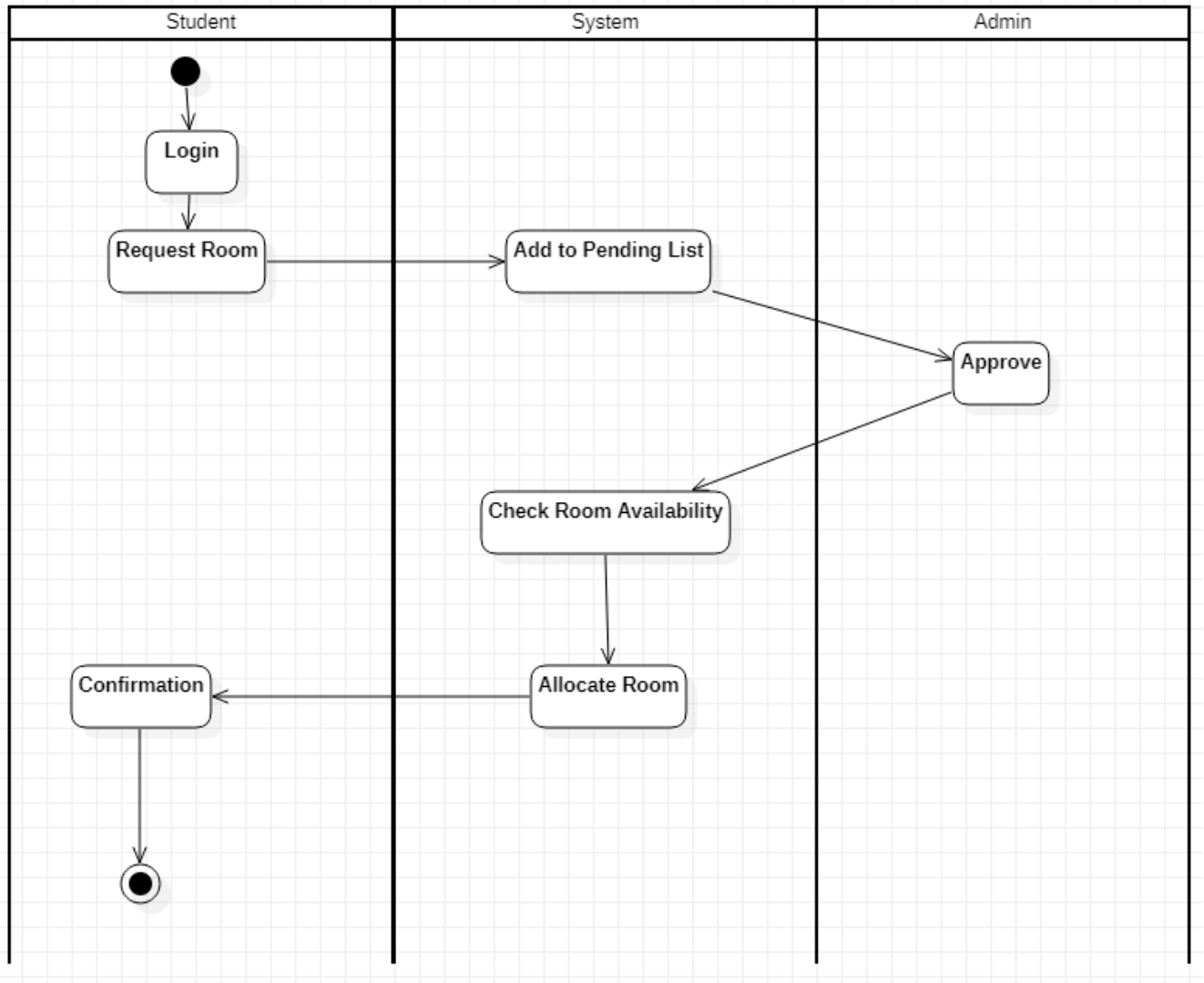

Fig 2.9

The advanced activity diagram shows the activities involved using swim lanes.

The swim lanes provide information about the persons responsible for the activities in the system.

The respective persons activities are shown under their swim lanes. In the diagram shown, student, system and admin are the swim lanes who perform the activities shown in the diagram.

3. Stock Maintenance System

Problem Statement:

To design a Stock Maintenance System to store information regarding stock of products and conduct transactions. The system will allow administrators to maintain stock of products by streamlining process of purchasing from suppliers and selling to customers. The information regarding stock is easily accessible and allows for ease of use.

SRS:

Purpose:

The Stock Maintenance System is an online system. The system is developed to provide detailed information regarding transactions related to stock maintenance and ease the process of carrying out such transaction. The system allows administrators to maintain stock of products by streamlining process of purchasing from suppliers and selling to customers.

Requirements:

Functional Requirements:

- Provide Admin Login:
  Admin login is required to manage stock and carry out transactions.
- Provide Stock Details:
  Details of product in stock such as quantity, quality and price can be viewed
- Maintain Transaction History:
  All details of transactions carried out with suppliers and customers should be stored in database
- GUI:
  An interface is developed for above mentioned functional requirements.

Non-Functional Requirements:

- Cost:
  The software should be cheap and within budget.
- Time:
  The software product should be completed within deadlines
- Security:
  The data should be secure

Class Diagram:

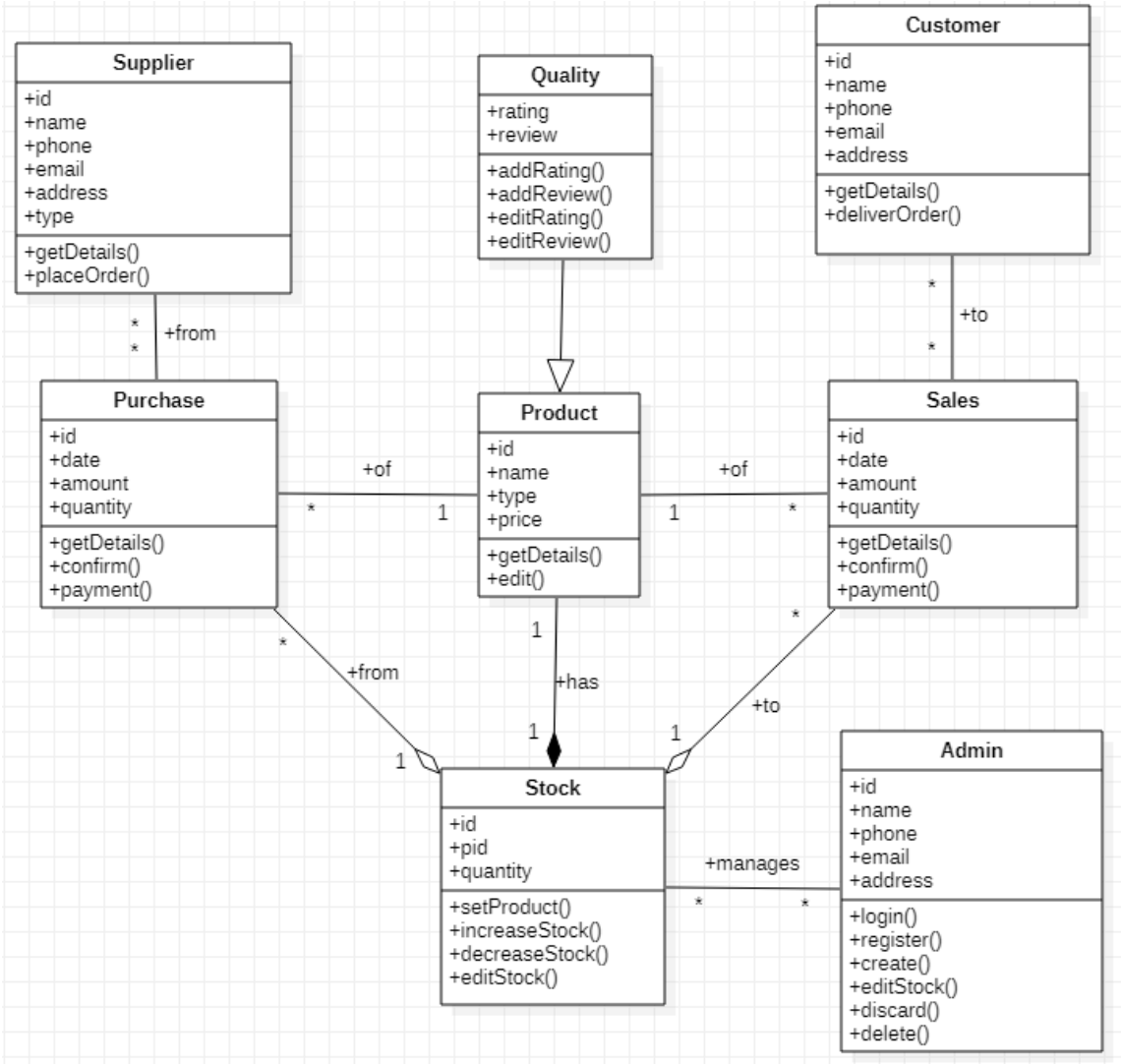

Fig 3.1

- Supplier: Contains information about supplier and handles placing orders.
- Purchases: Contains information about purchases and handles payment for orders.
- Customer: Contains information about customer and handles delivery of orders.
- Sales: Contains information about sales and receives payment from customer.
- Product: Contains information about product.
- Quality: Allows rating and review of product.
- Stock: Contains information about stock.
- Admin: Manages stock.

State Diagram:

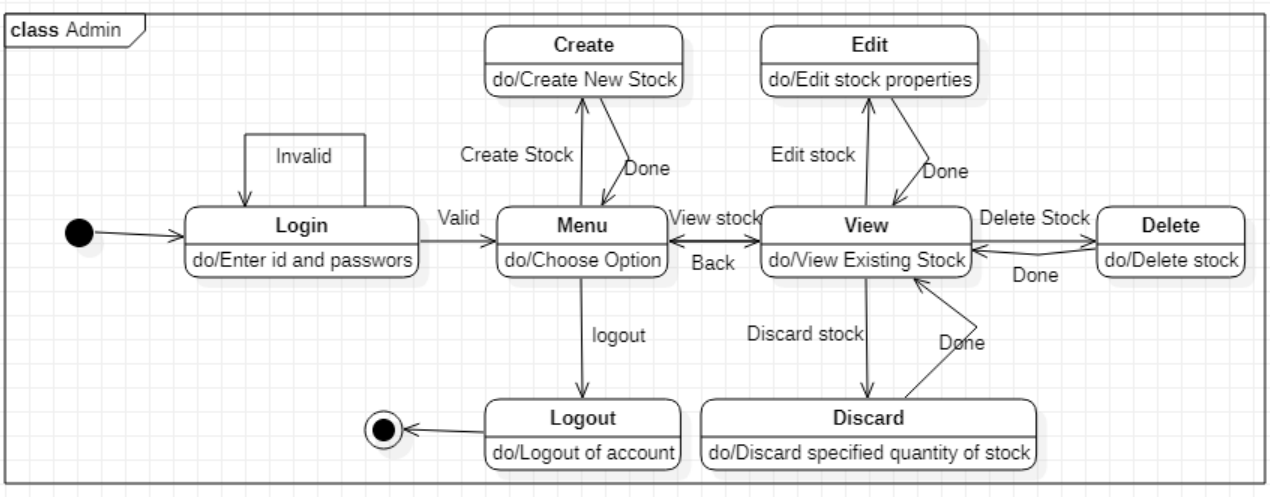

Fig 3.2

The state diagram shows the options available to admin.
First they login. If valid, they can access the menu. They can create new stock, view and edit or delete stock, discard a specified amount of stock or logout.

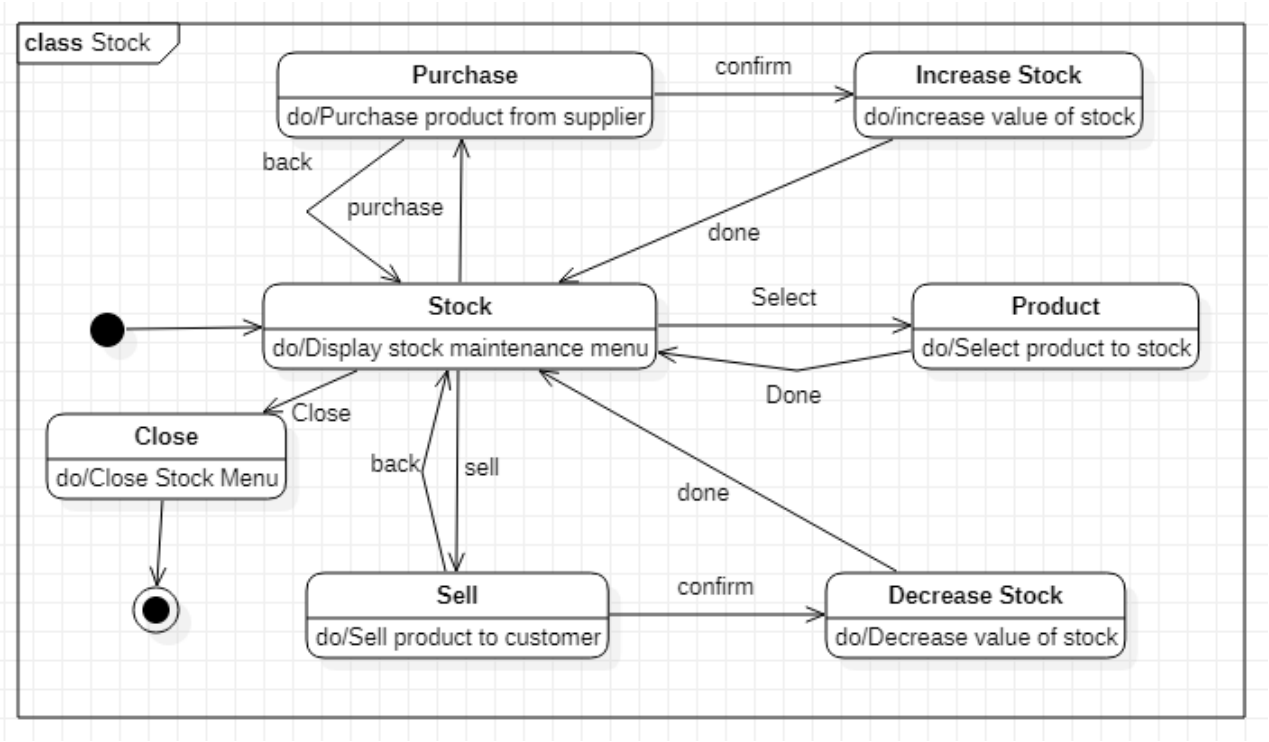

Fig 3.3

The state diagram shows the functionalities of stock.
On purchasing products from suppliers, the value of stock is increased.
On selling products to customers, the value of stock is decreased.
Product to stock can be selected.
Stock menu can be closed.

Use Case Diagrams:

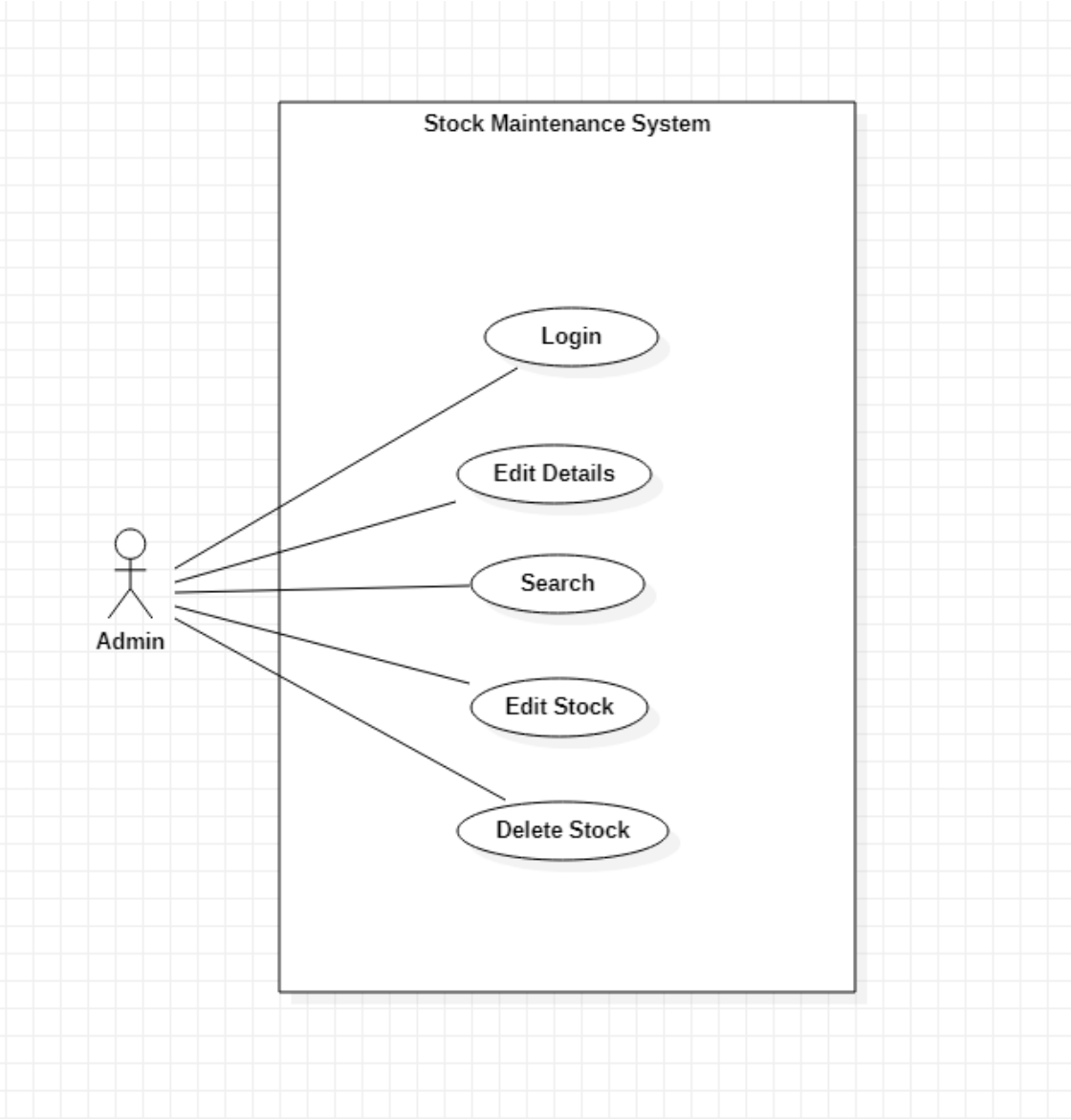

Fig 3.4

There is one actor:

- **Admin:** Person who manages the stock

There are five use cases:

- **Login:** Login to Stock Management Database
- **Edit Details:** Edit account details
- **Search:** Search for products
- **Edit Stock:** Edit Stock details
- **Delete Stock:** Delete stock from database

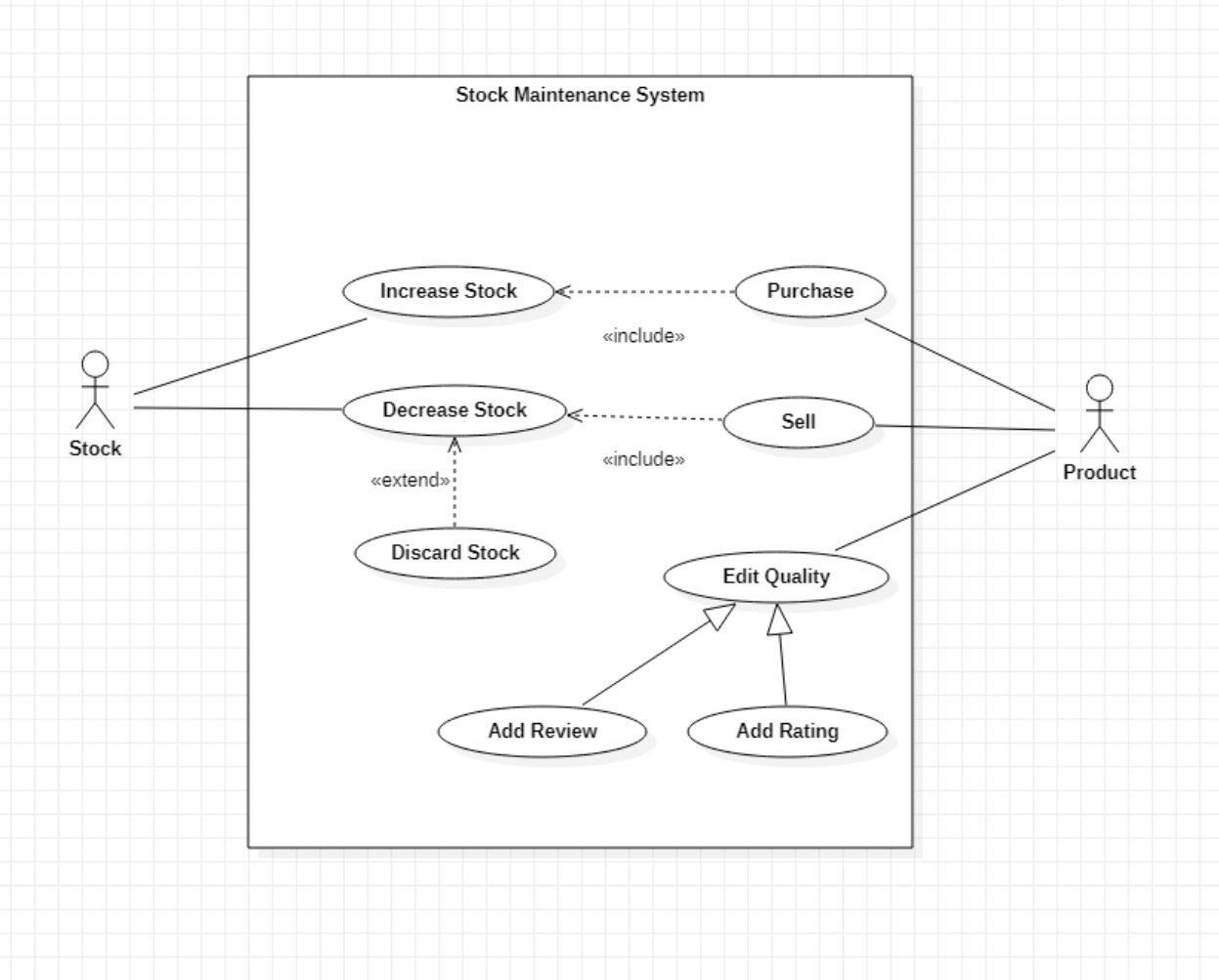

Fig 3.5

There are two actors:

- **Stock:** The place where merchandise is kept
- **Product:** The merchandise that is being stocked

There are eight use cases:

- **Increase Stock:** Increase stock of products
- **Decrease Stock:** Decrease stock of products
- **Discard Stock:** Discard faulty products
- **Purchase:** Purchase product from supplier
- **Sell:** Sell product to customer
- **Edit Quality:** Edit quality of product
- **Add Review:** Add product review
- **Add Rating:** Add product rating

Sequence Diagrams:

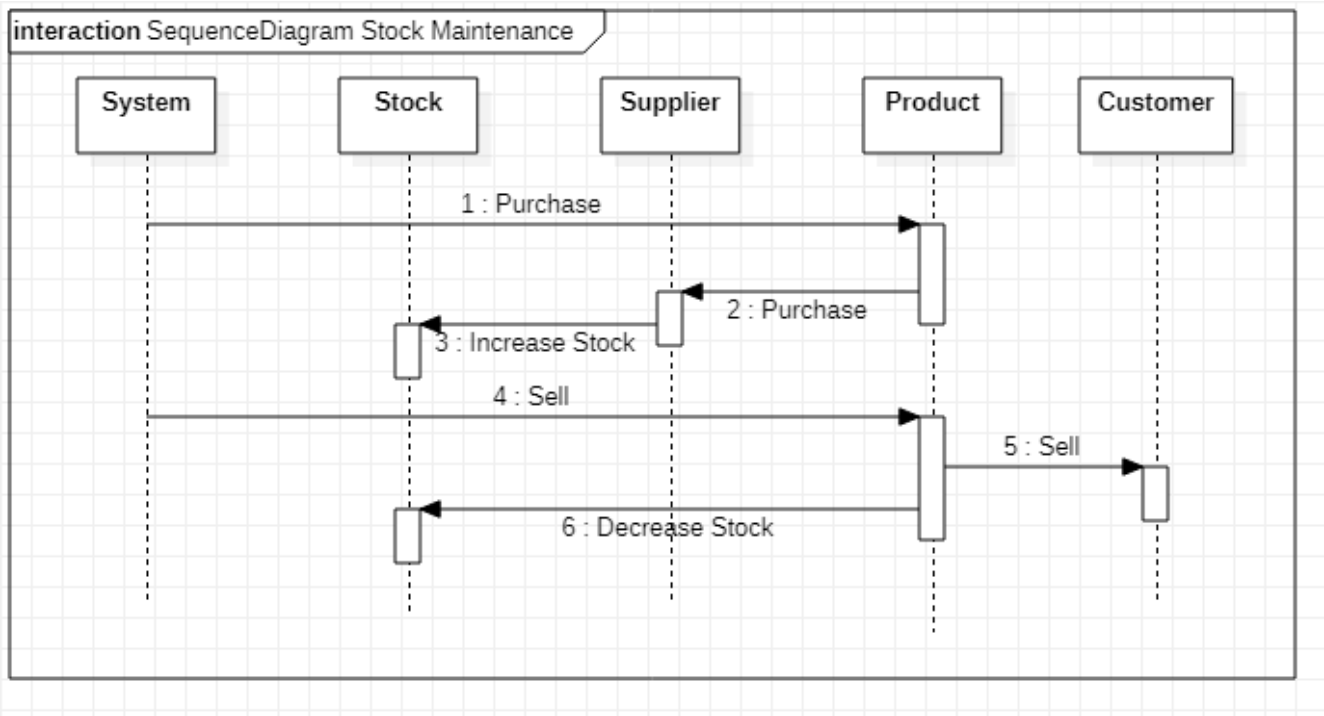

Fig 3.6

The sequence diagram shows the interaction of system with suppliers and customers in order to maintain stock of products.

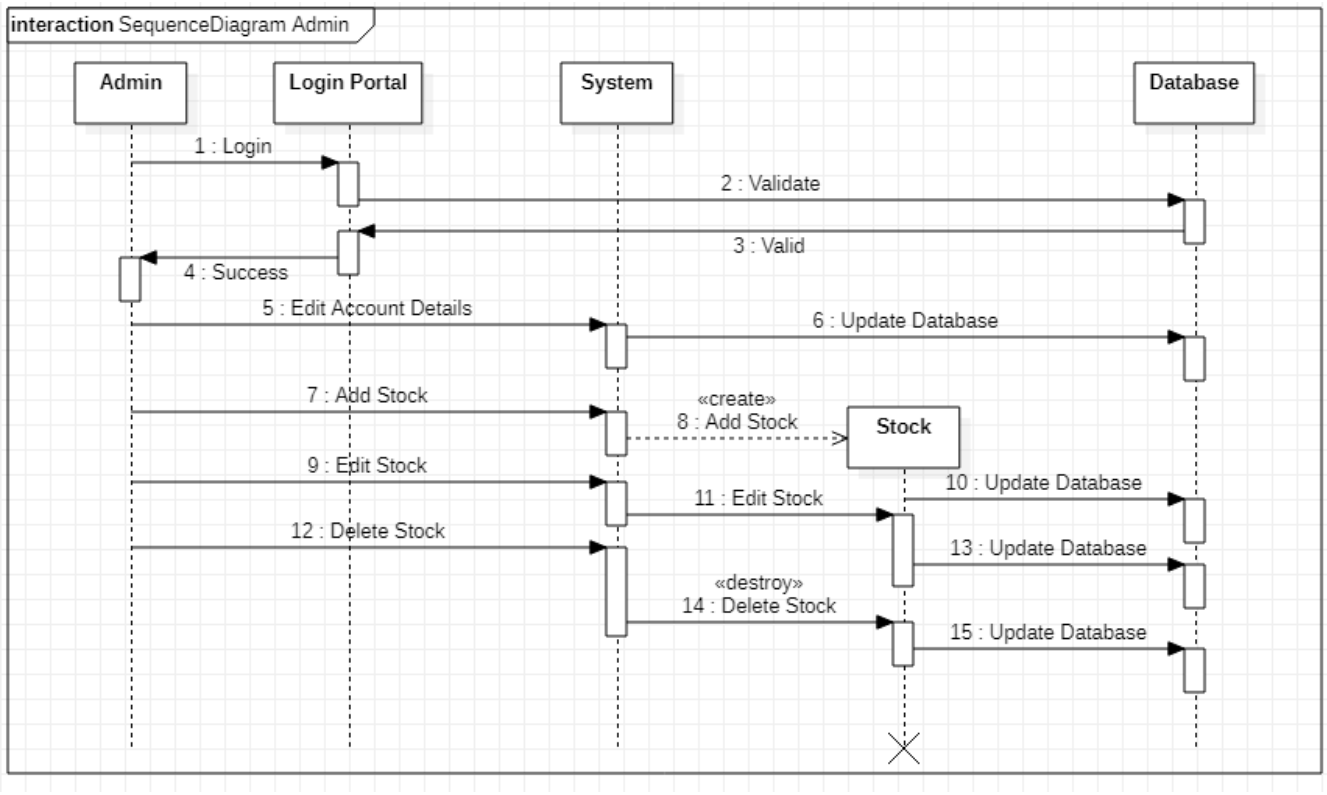

Fig 3.7

The sequence diagram shows the interaction of admin with system and database in order to manage stock.

Activity Diagrams:

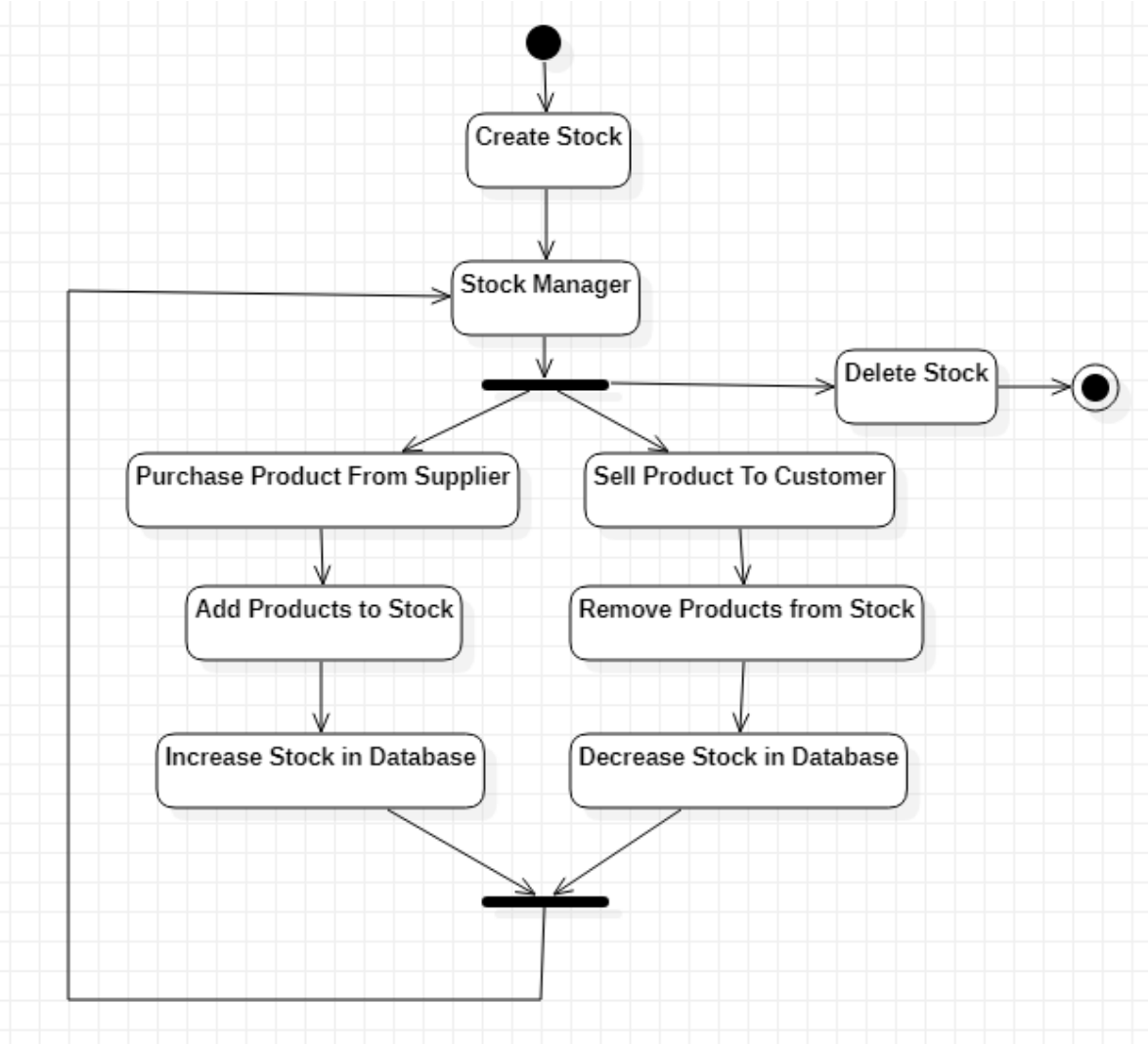

Fig 3.8

The activity diagram shows the activities involved in stock management.

They can purchase and add products to stock and then increase in database or sell products and remove products from stock and decrease in database or delete stock.

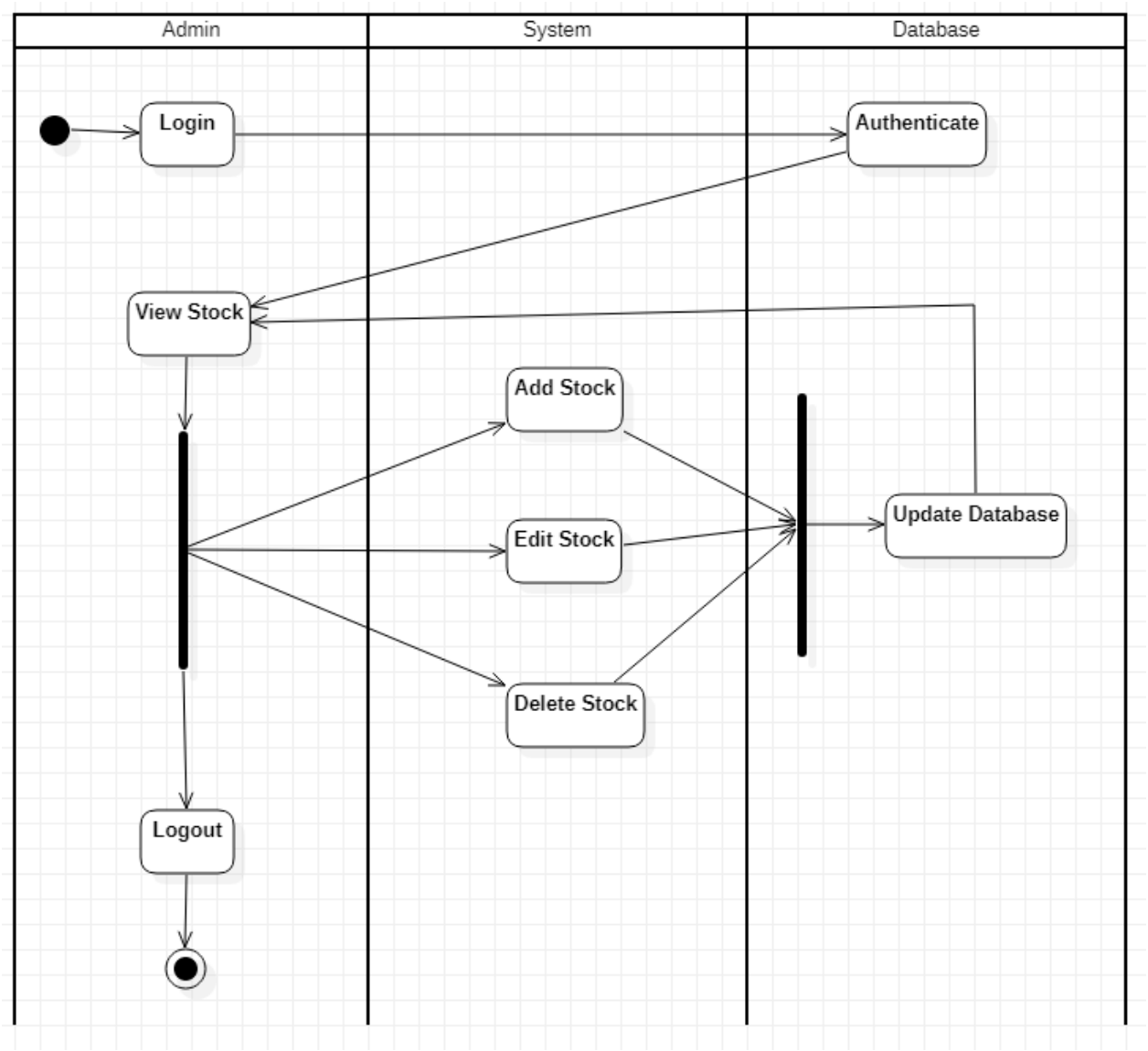

Fig 3.9

The advanced activity diagram shows the activities involved using swim lanes.

The swim lanes provide information about the persons responsible for the activities in the system.

The respective persons activities are shown under their swim lanes. In the diagram shown, admin, system and database are the swim lanes who perform the activities shown in the diagram.

4. Coffee Vending Machine

Problem Statement:

To design a Coffee Vending Machine to brew coffee based on user input. It reduces the time required to brew coffee by eliminating the need for manual preparation. The user enters the type of coffee they desire and inserts the required amount of money.

SRS:

Purpose:

The Coffee Vending Machine is an automated system. The system allows users to enter the type of coffee they desire and it is automatically prepares the coffee. It saves time and effort by eliminating the need for manual preparation of coffee. The ingredients are stored in the machine and the user does not require knowledge of how to make coffee in order to use it.

Requirements:

Functional Requirements:

- Choose Type:
  User can enter type of coffee they desire.
- Get Coffee:
  Checks the recipe for selected coffee type and brews the coffee.
- Set Temperature:
  Allows user to set temperature of coffee.
- Validate Inserted Money:
  Validates the money inserted by user and returns change.
- Dispense Coffee:
  Dispenses coffee if required money inserted.
- GUI:
  An interface is developed for above mentioned functional requirements.

Non-Functional Requirements:

- Cost:
  The software should be cheap and within budget.
- Time:
  The software product should be completed within deadlines
- Security:
  The data should be secure

Class Diagram:

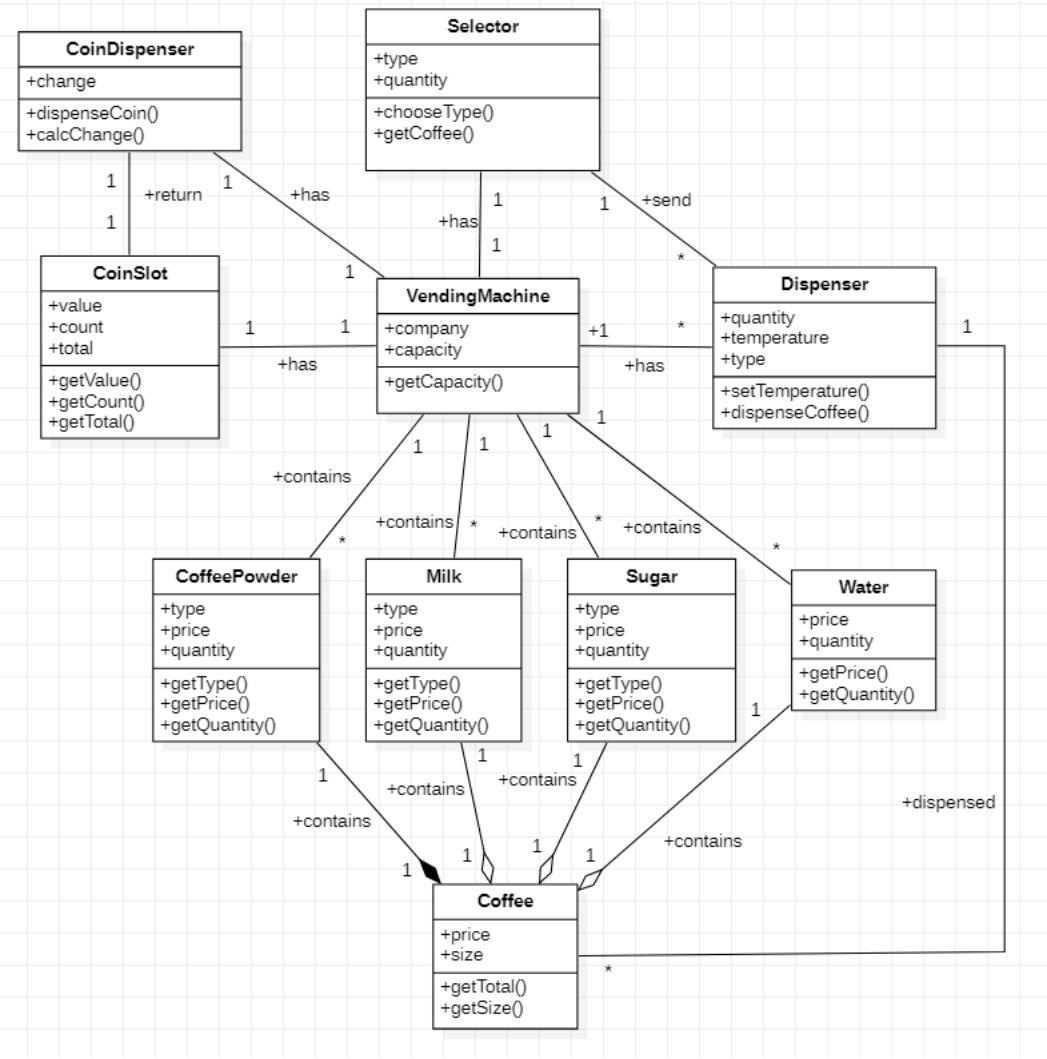

Fig 4.1

- Coin Dispenser: Calculates and returns change.
- CoinSlot: Calculates price and takes and validates coins from customer.
- Selector**:** Allows choosing type of coffee.
- VendingMachine: Manages ingredients
- CoffeePowder: Contains information about coffee powder.
- Milk: Contains information about milk.
- Sugar: Contains information about sugar.
- Water: Contains information about water.
- Coffee: Contains information about coffee.
- Dispenser: Dispenses coffee and sets temperature.

State Diagram:

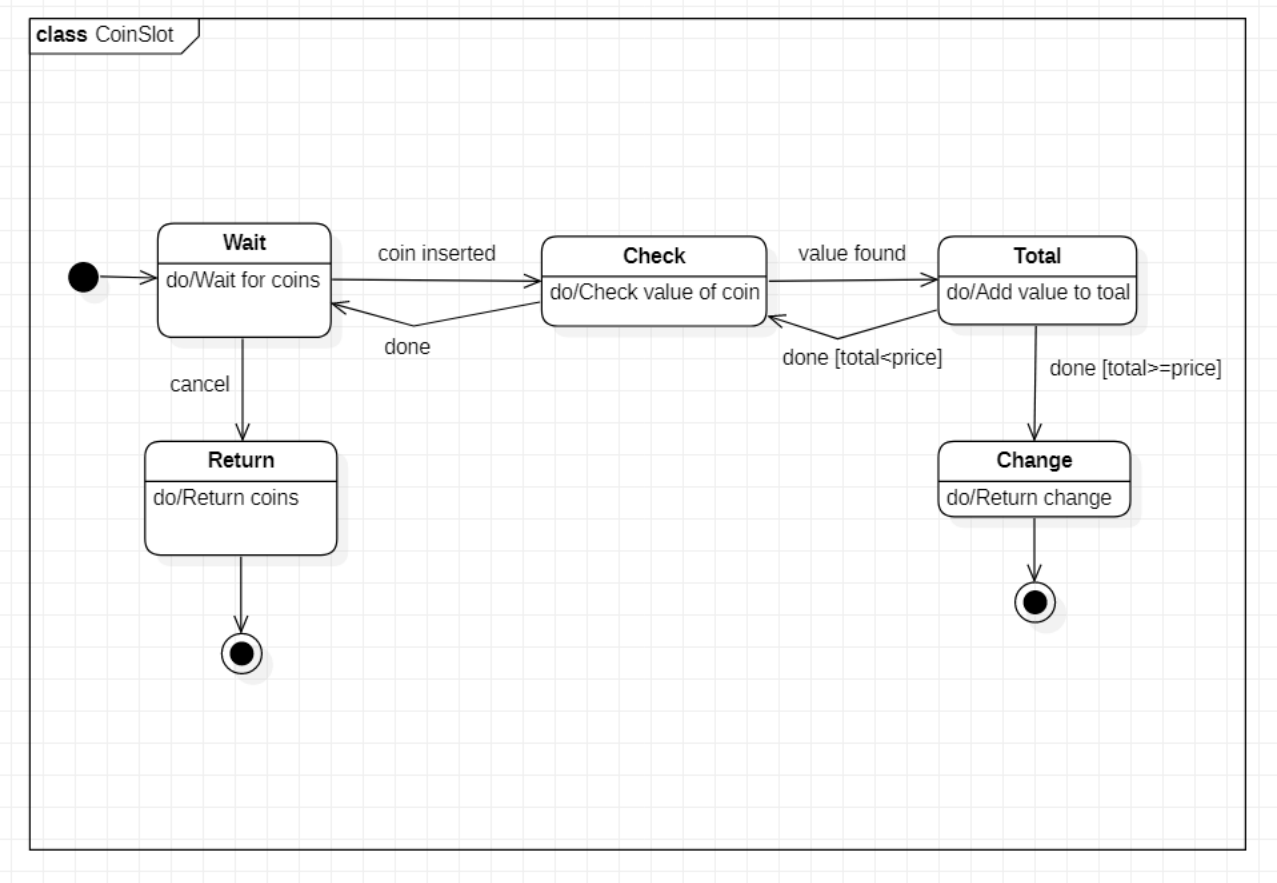

Fig 4.2

The state diagram shows the functioning of coin slot.
It waits for coins. Once coin is inserted, it checks its value. It adds it to total. If total>=price, it calculates and returns change. Otherwise, it continues waiting. If order canceled, coins are returned.

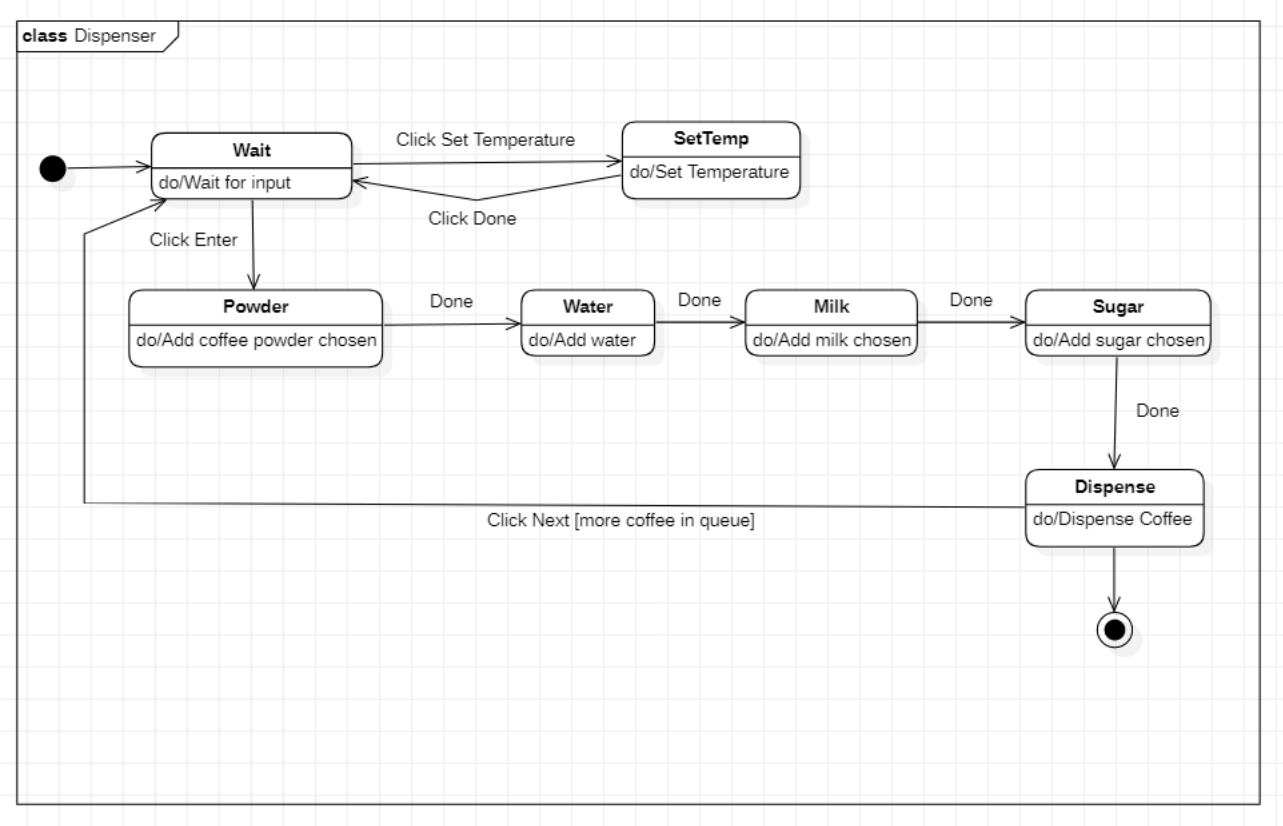

Fig 4.3

The state diagram shows functioning of dispenser.

It waits for user input. User can set temperature.

First, coffee powder is added followed by water, milk and sugar. Then the coffee is dispensed. If more orders where queued, it repeats the process.

Use Case Diagrams:

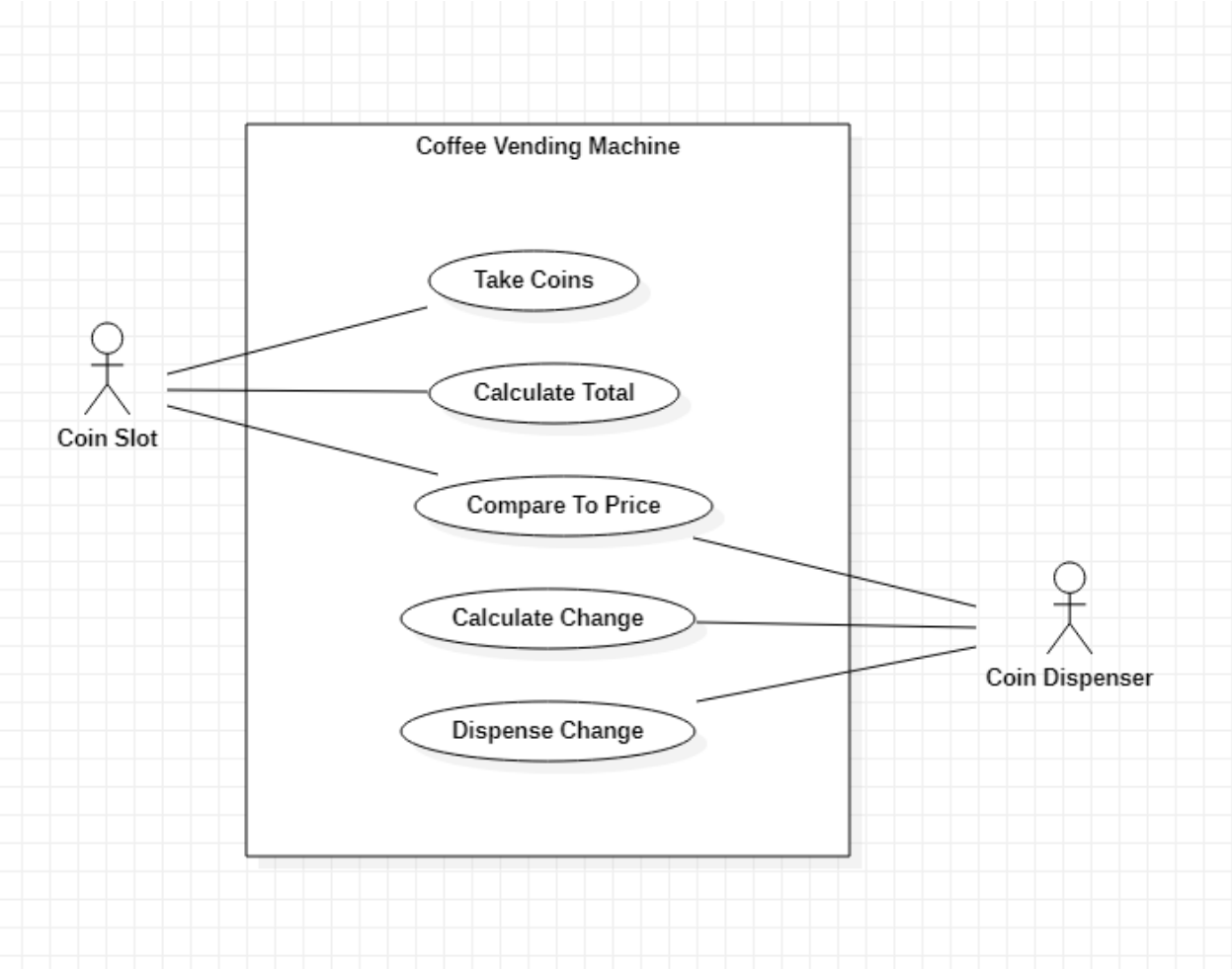

Fig 4.4

There are two actors:

- **Coin Slot:** The slot in which coins are inserted
- **Coin Dispense:** The dispenser that returns change

There are five use cases:

- **Take Coins:** Take coins from user
- **Calculate Total:** Calculate total value of inserted coins
- **Compare To Price:** Compare value inserted to price of coffee
- **Calculate Change:** Calculate change to return
- **Dispense Change:** Return change to user

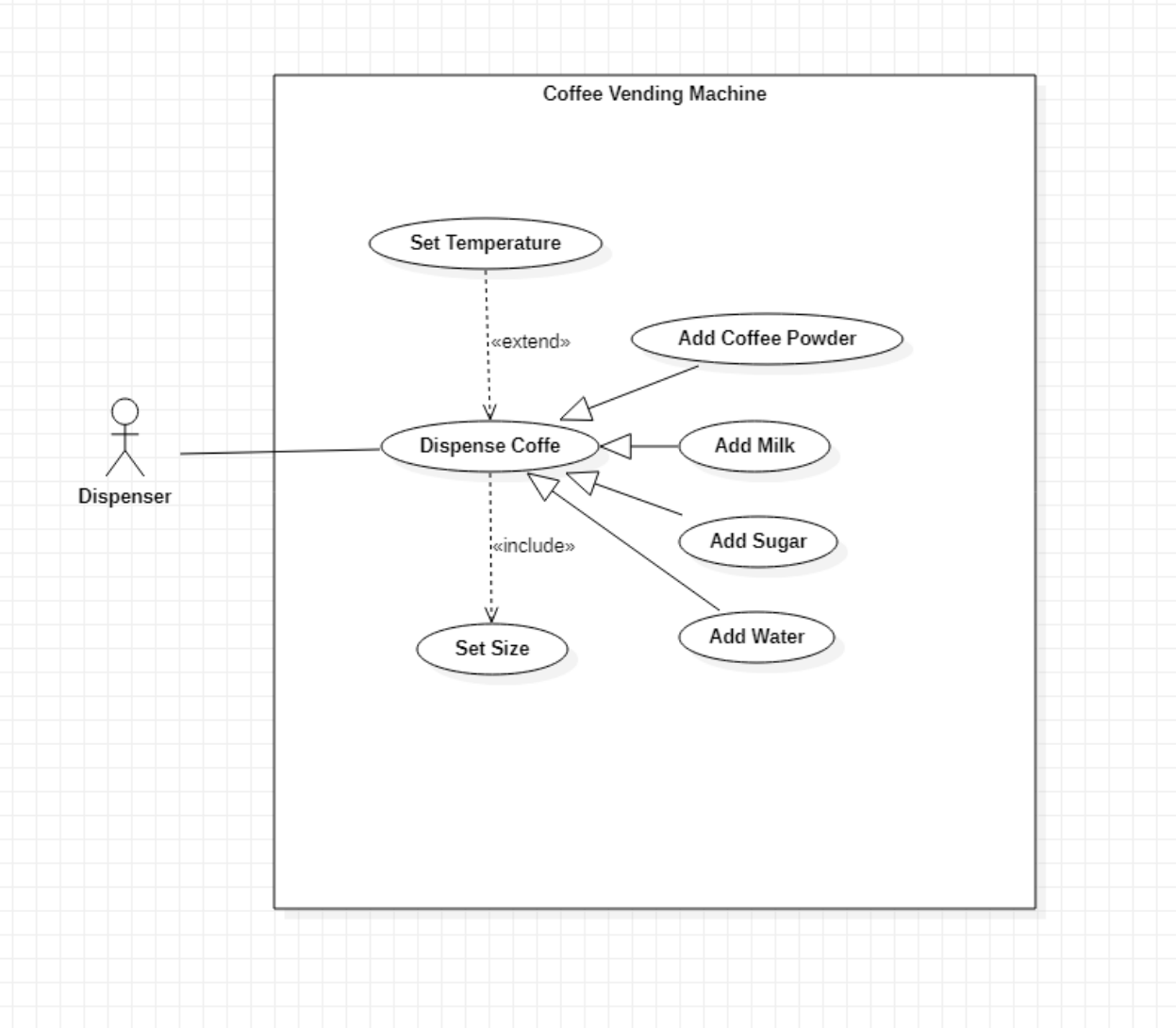

Fig 4.5

There is one actor:

- **Dispenser:** The equipment that mixes and dispenses coffee

There are seven use cases:

- **Dispense Coffee:** Dispense coffee as selected
- **Add Coffee Powder:** Add coffee powder to mix
- **Add Milk:** Add milk to mix
- **Add Sugar:** Add sugar to mix
- **Add Water:** Add water to mix
- **Set Size:** Set size of coffee desired
- **Set Temperature:** Set desired temperature for coffee

Sequence Diagrams:

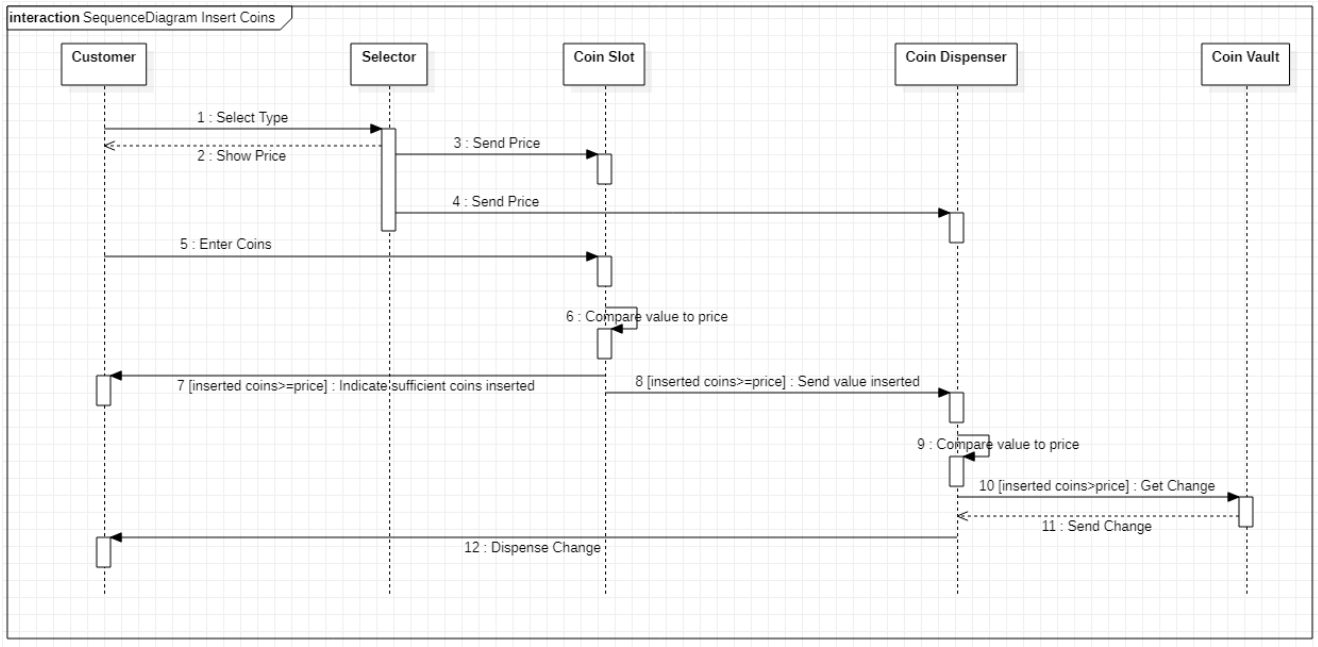

Fig 4.6

The sequence diagram shows the interaction of customer with selector, coin slot, and coin dispenser. It represents how coins are inserted to satisfy price and return change.

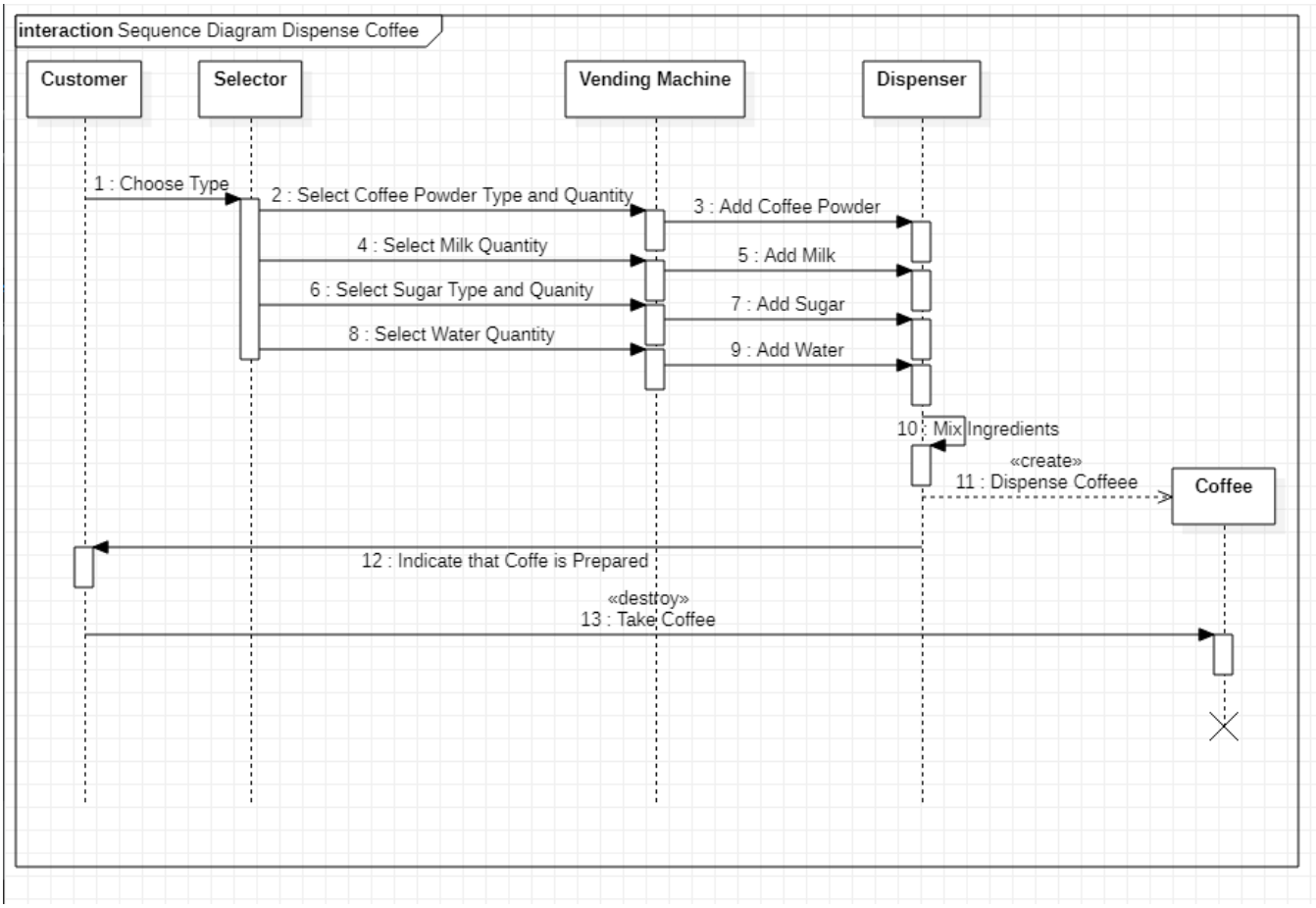

Fig 4.7

The sequence diagram shows the interaction of customer with dispenser through selector and vending machine in order to dispense coffee.

Activity Diagrams:

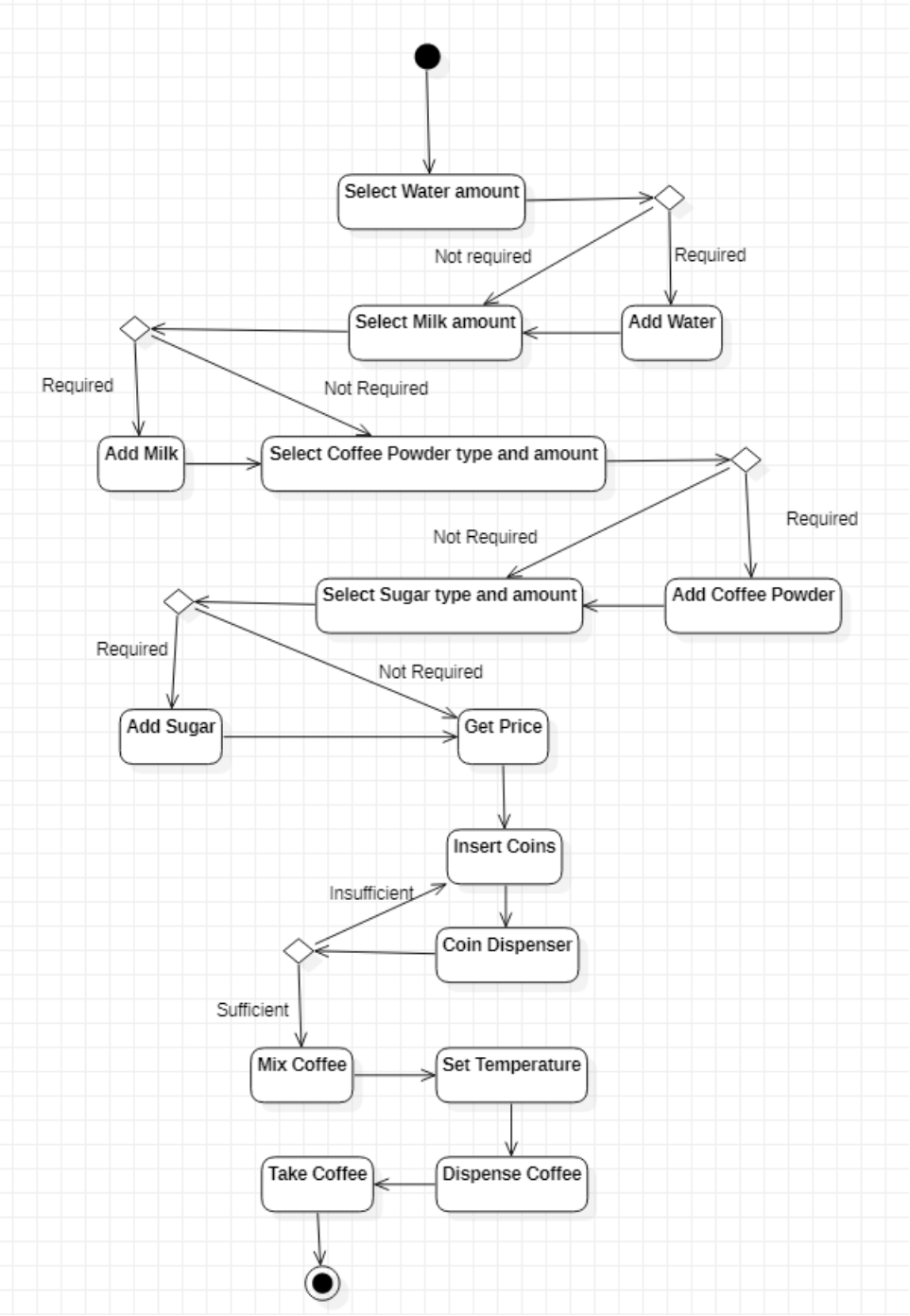

Fig 4.8

The activity diagram shows the activities involved in dispensing coffee.

User can select ingredients followed by their addition. Then price is calculated, coins inserted, coffee mixed, temperature set, coffee dispensed and taken.

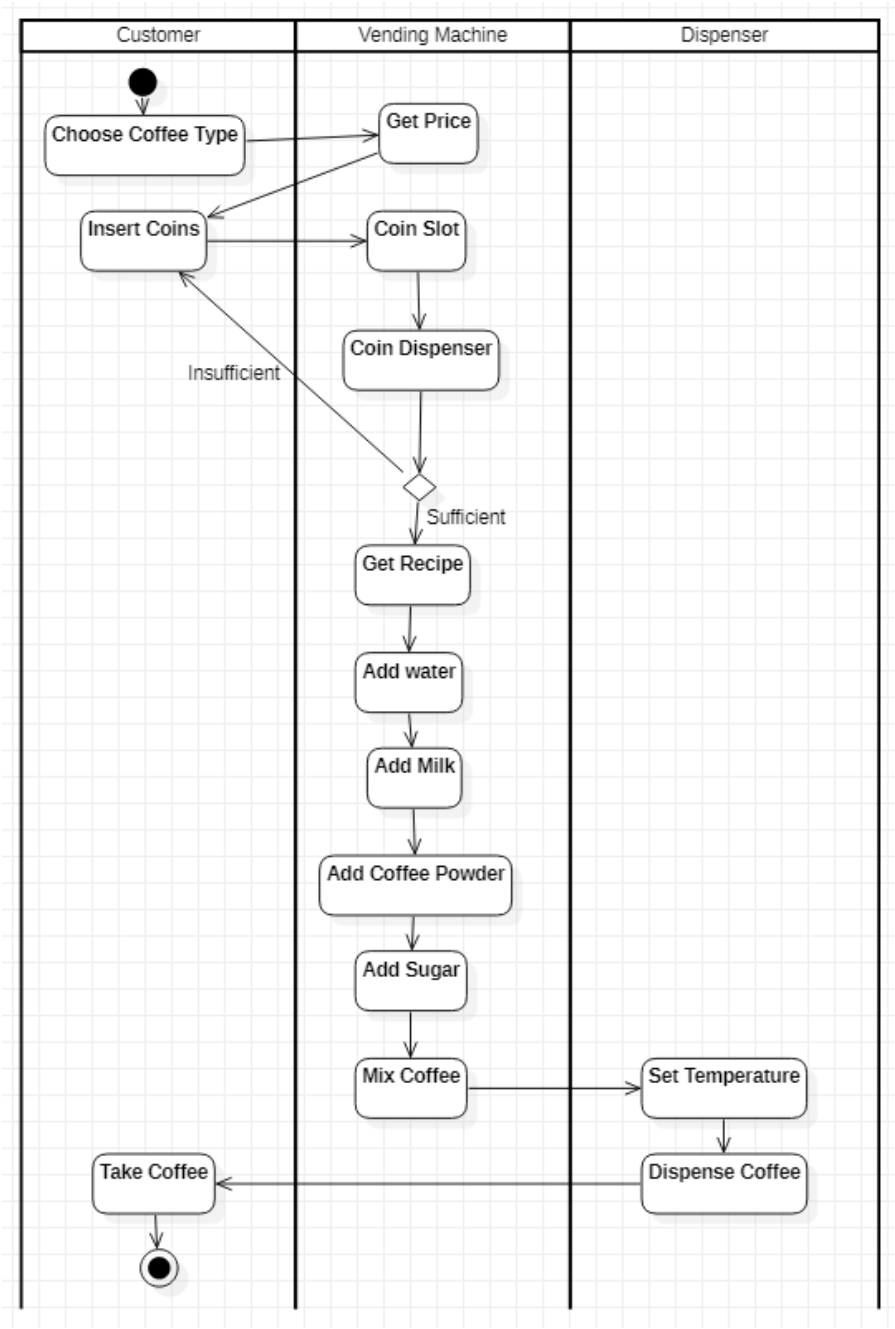

Fig 4.9

The advanced activity diagram shows the activities involved using swim lanes.

The swim lanes provide information about the persons responsible for the activities in the system.

The respective persons activities are shown under their swim lanes. In the diagram shown, customer, vending machine and dispenser are the swim lanes who perform the activities shown in the diagram.

5. Online Shopping System

Problem Statement:

To design an Online Shopping System to simplify to allow user to purchase products online. The system will allow users to search for and buy products online. The payment is made online and user can track the order.

SRS:

Purpose:

The Online Shopping System is an online system. The system is developed to allow users to search for and purchase products online. The user can make payments online and track the delivery of the product. All transactions are made and online and information is directly stored in database thereby reducing paperwork, travelling and time consumption.

Requirements:

Functional Requirements:

- Provide User Login:
  Users can login and edit details of their account.
- Show Products:
  User can search for and view details of products
- Provide Cart:
  User can add product to cart and buy items together.
- Provide Online Payment Facility:
  User can pay for items online.
- Provide Delivery Tracking Facility:
  User can view details of delivery and check estimated delivery time.
- Provide Admin Login:
  Admin login is required to update available.
- GUI:
  An interface is developed for above mentioned functional requirements.

Non-Functional Requirements:

- Cost:
  The software should be cheap and within budget.
- Time:
  The software product should be completed within deadlines
- Security:
  The data should be secure

Class Diagram:

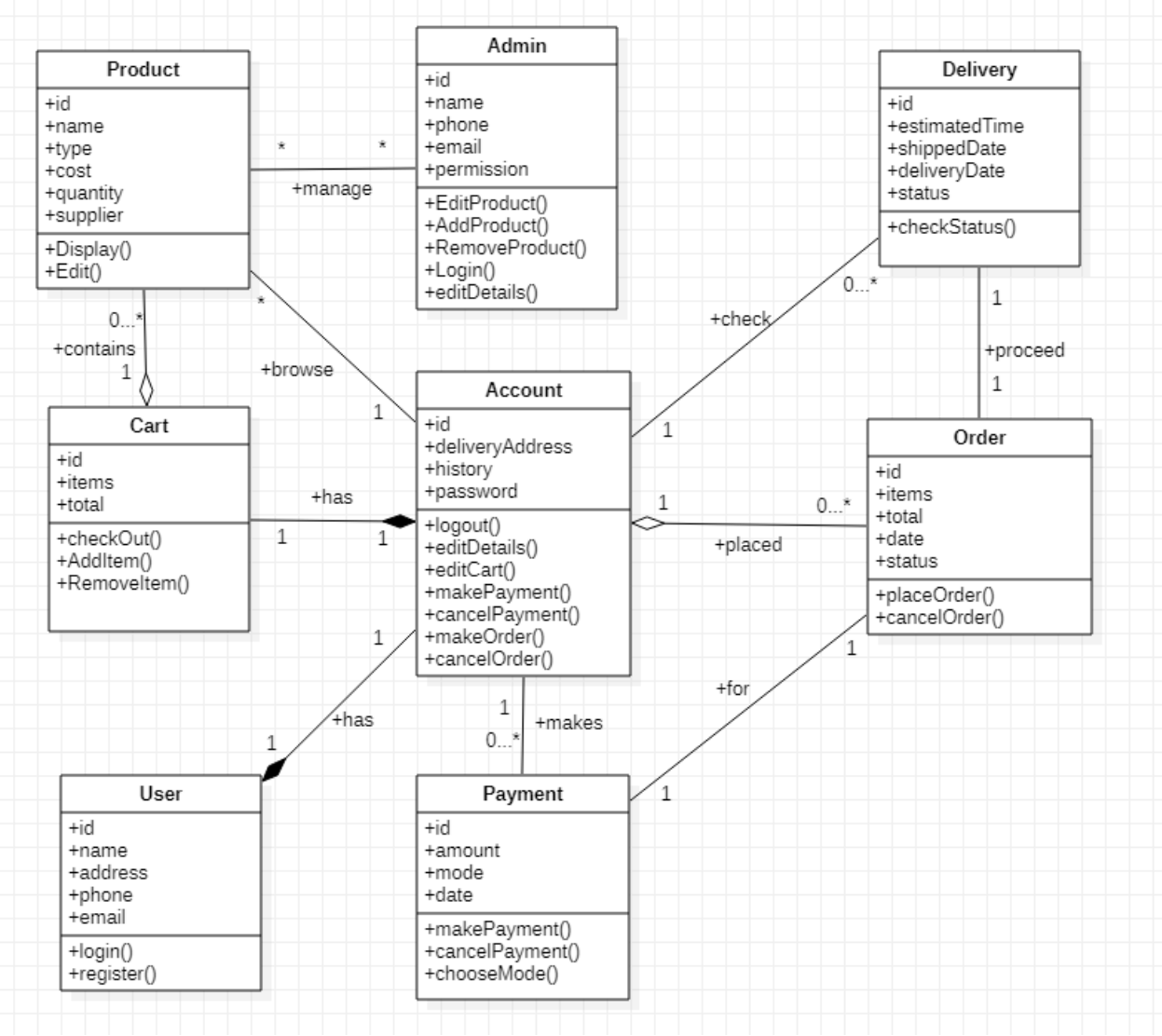

Fig 5.1

- Product: Contains information about product.
- Admin: Manages available products.
- Cart: Contains products chosen for order.
- Account: Used to interact with online shopping portal. Allows browsing products, managing cart and payment.
- User: Contains information about user.
- Payment: Contains information about and manages payments.
- Order: Contains information about and manages order.
- Delivery: Contains information about delivery.

State Diagram:

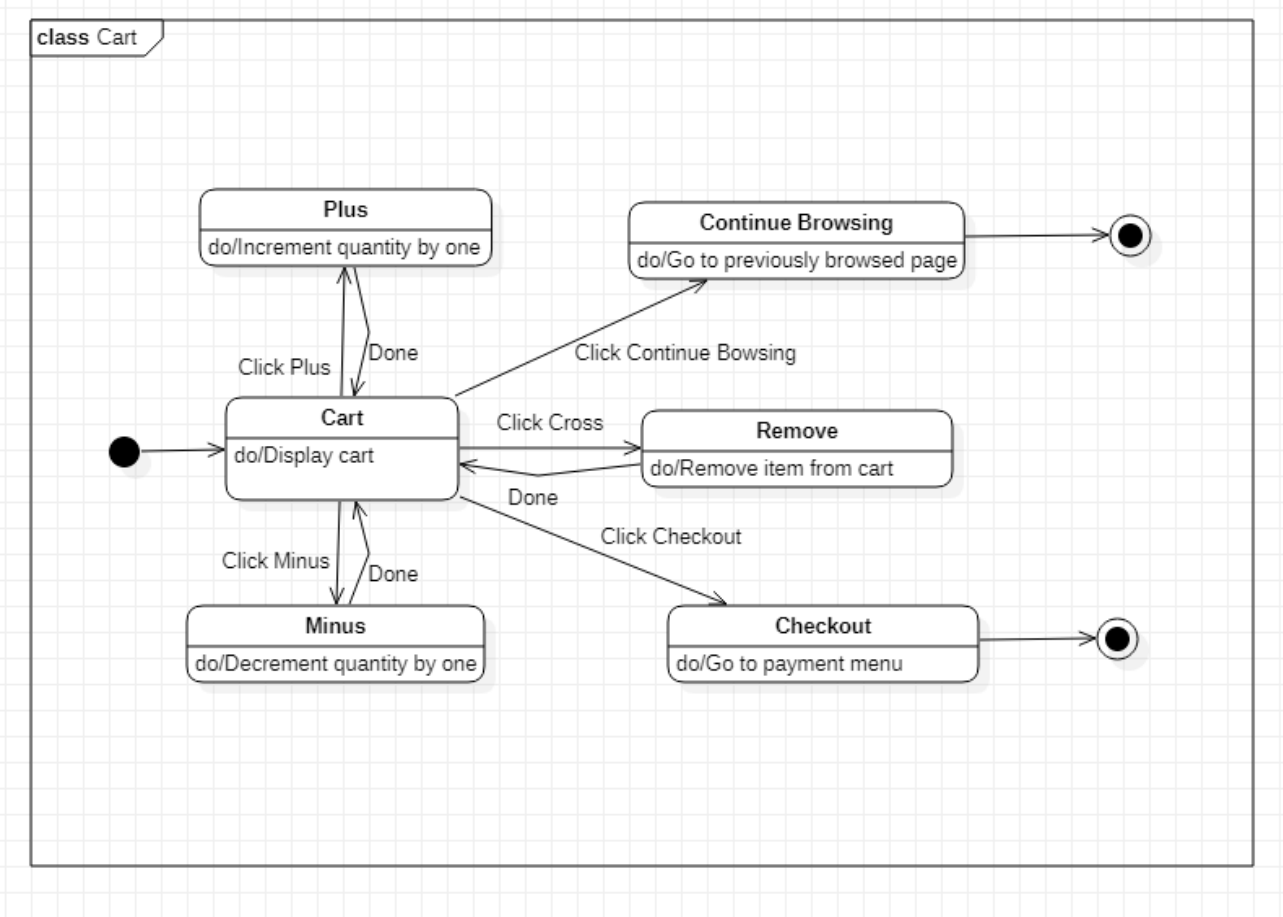

Fig 5.2

The state diagram shows the functioning of cart.
First it displays cart. Clicking plus increases quantity and minus decreases quantity. Items can be removed from cart. You can continue browsing or checkout.

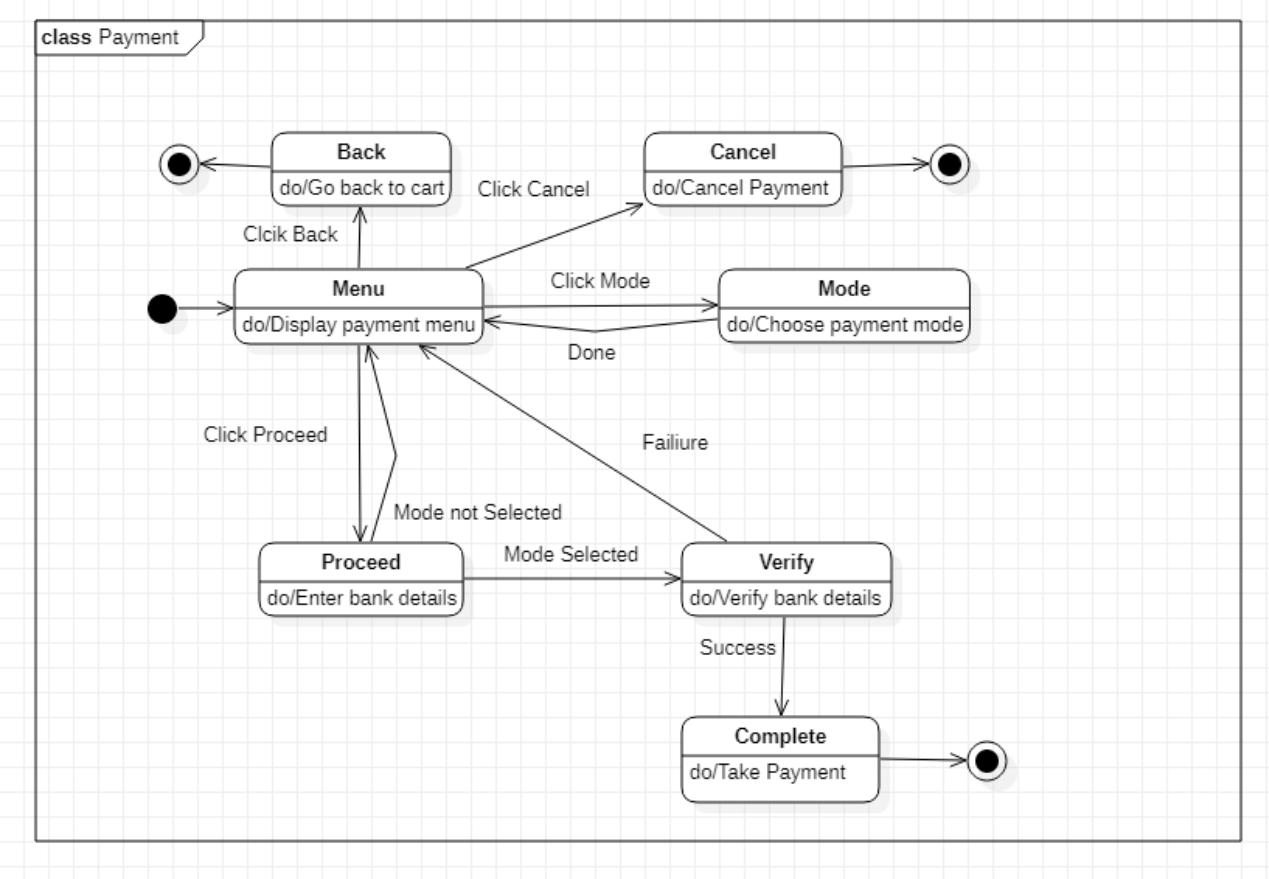

Fig 5.3

The state diagram shows payment process.
First, payment menu is displayed.
You can go back to cart or cancel payment.
Payment mode can be selected.
You can proceed to enter bank details.
If verification is successful, payment is completed. Otherwise, you are returned to payment menu.

Use Case Diagrams:

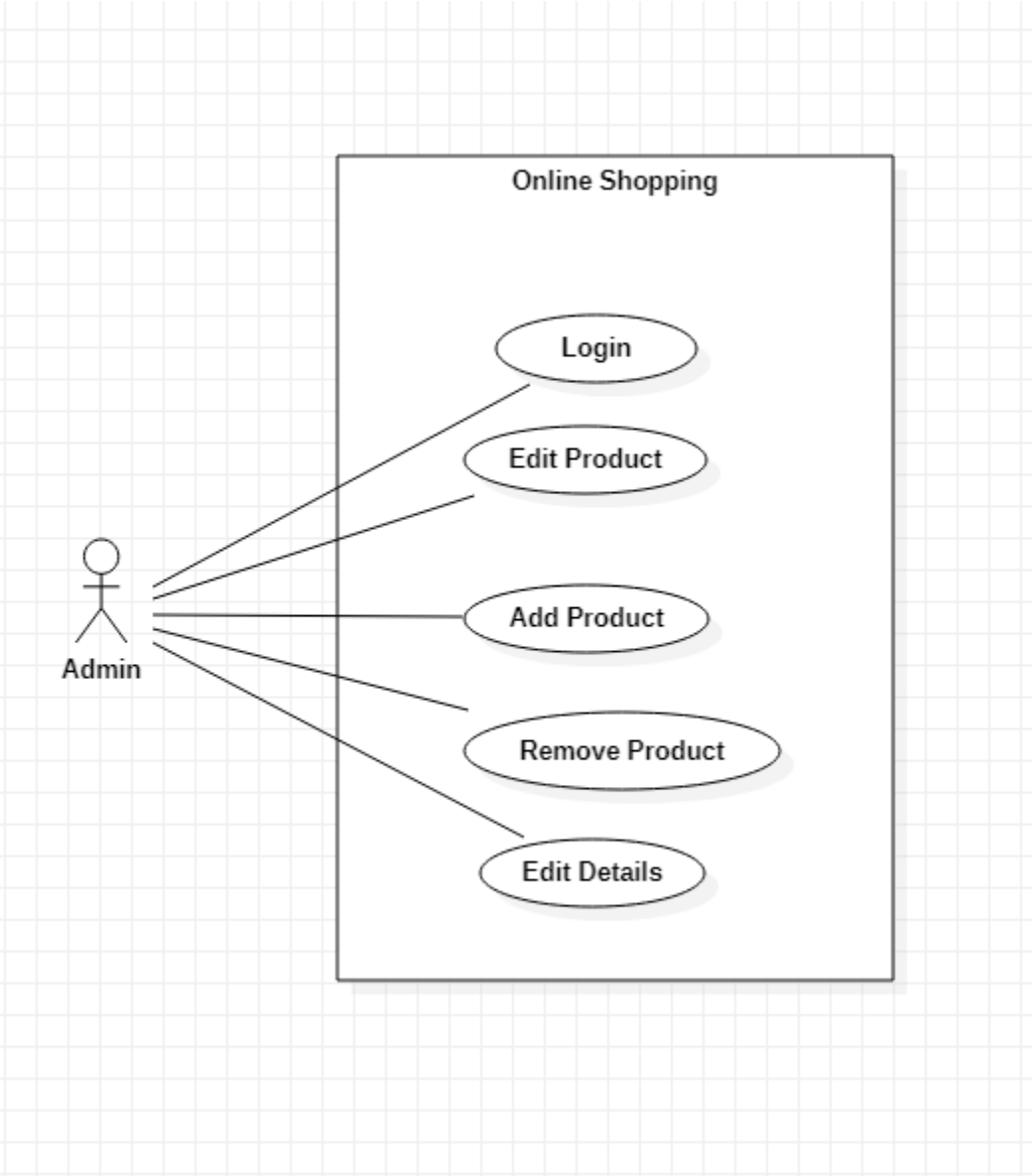

Fig 5.4

There is one actor:

- **Admin:** Person who manages products for sale

There are five use cases:

- **Login:** Login to online shopping website
- **Edit Product:** Edit Product details
- **Add Product:** Add product to store
- **Remove Product:** Remove product from store
- **Edit Details:** Edit account details

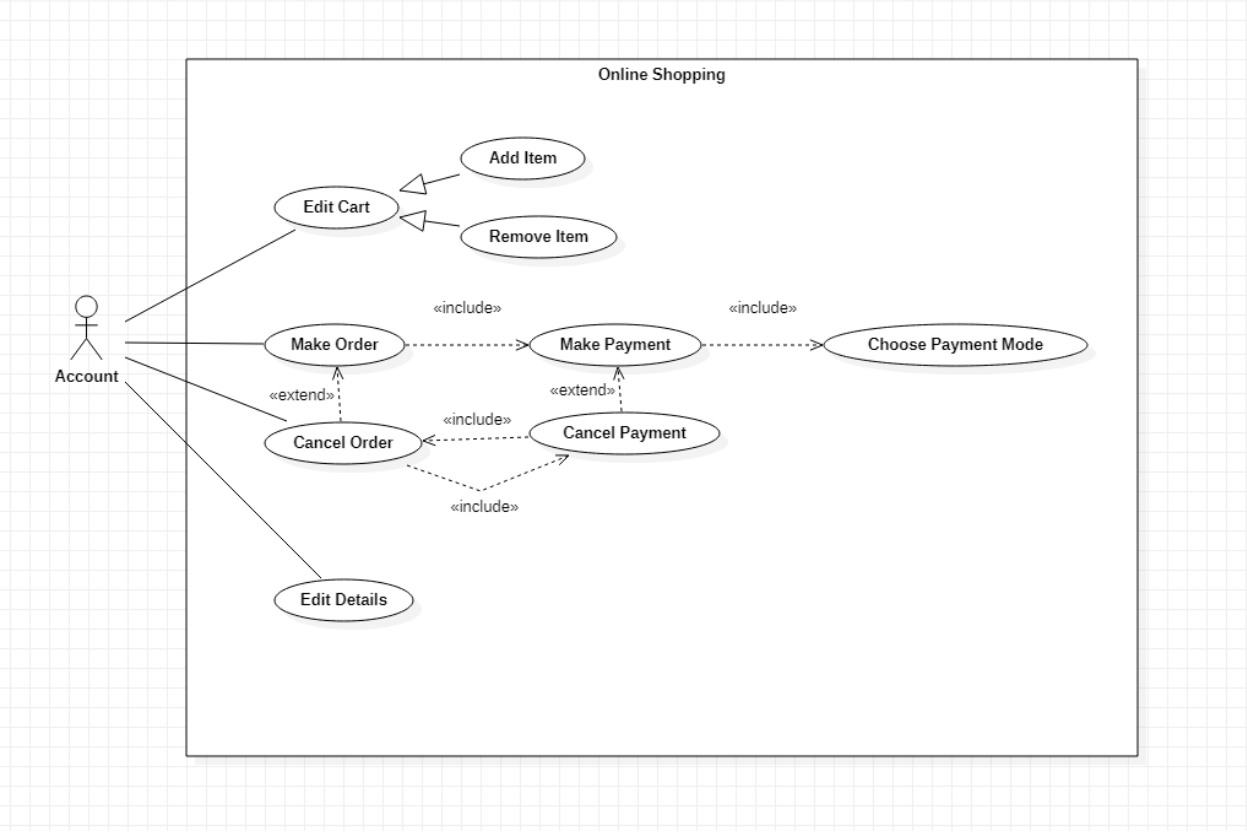

Fig 5.5

There is one actor:

- **Account:** Created by user to make transactions

There are nine use cases:

- **Edit Cart:** Edit shopping cart
- **Add Item:** Add items to shopping cart
- **Remove Item:** Remove items from shopping cart
- **Make Order:** Confirm order for items from cart
- **Make Payment:** Make payment for order
- **Cancel Order:** Cancel order before delivery
- **Cancel Payment:** Cancel payment for an order
- **Choose Payment Mode:** Choose payment mode for order
- **Edit Details:** Edit account details

Sequence Diagrams:

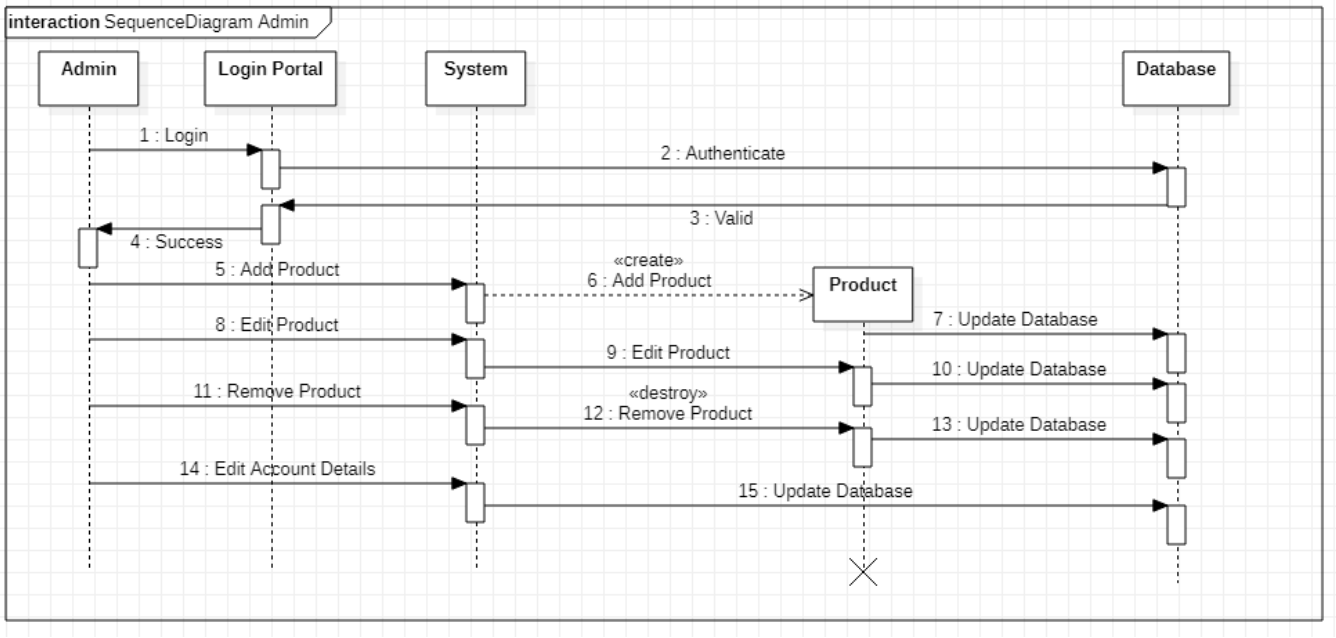

Fig 5.6

The sequence diagram shows the interaction of admin with system and database in order to manage products available on shopping portal.

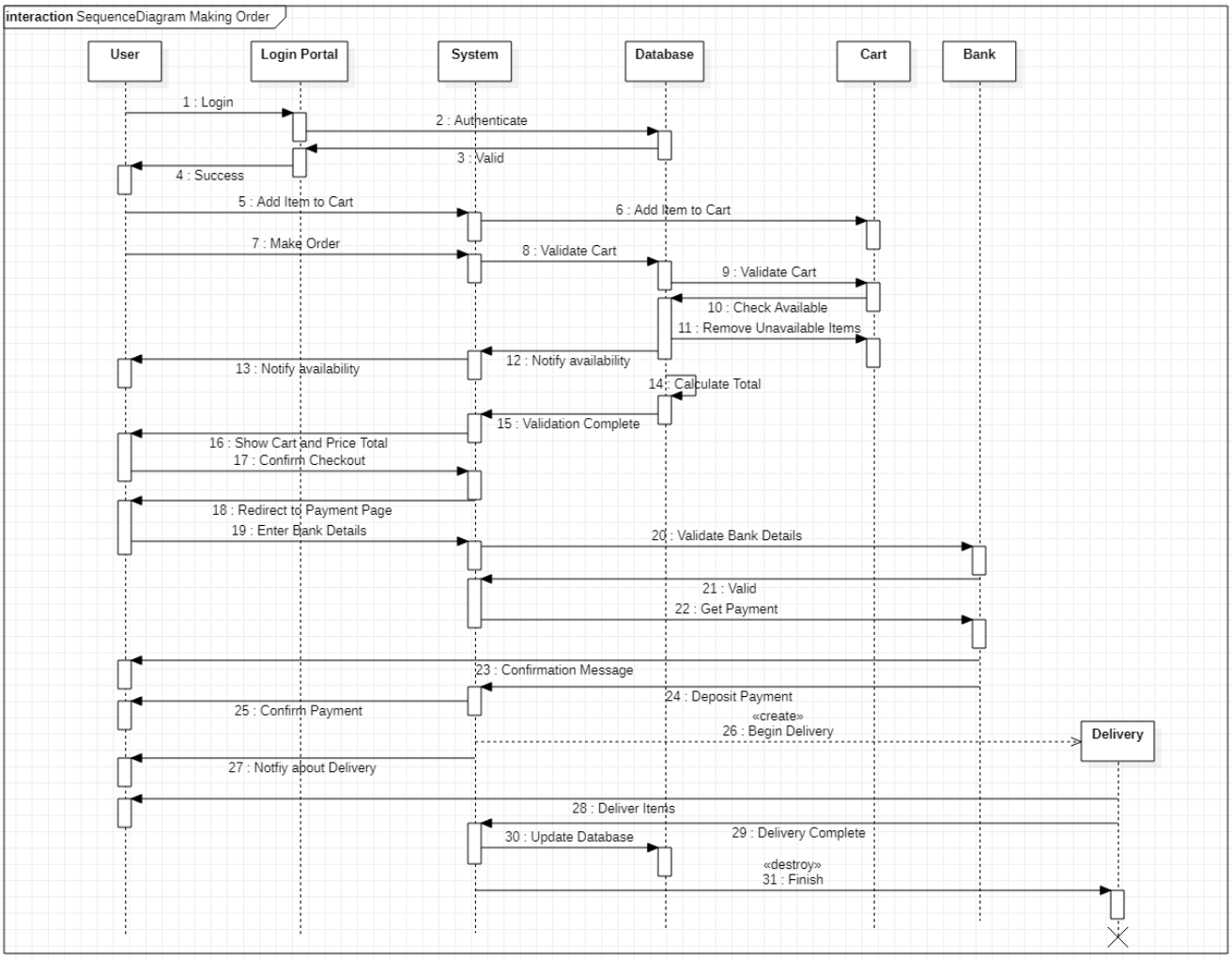

Fig 5.7

The sequence diagram shows the interaction of user with system and bank in order to add products to cart, make orders and carry out payments.

Activity Diagrams:

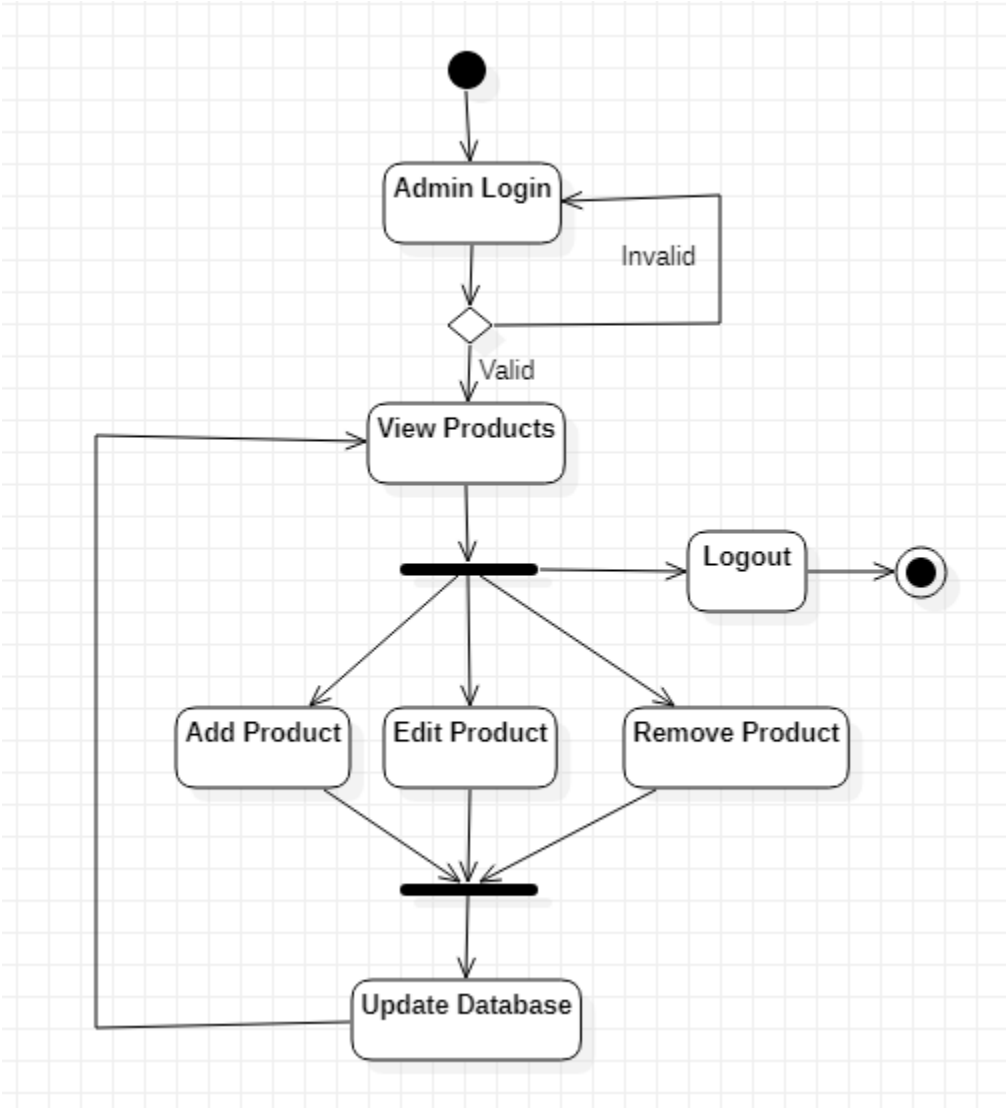

Fig 5.8

The activity diagram shows the activities involved in product management by admin.

First they need to login, then they can view products, then add, edit or remove products. Database is updated.

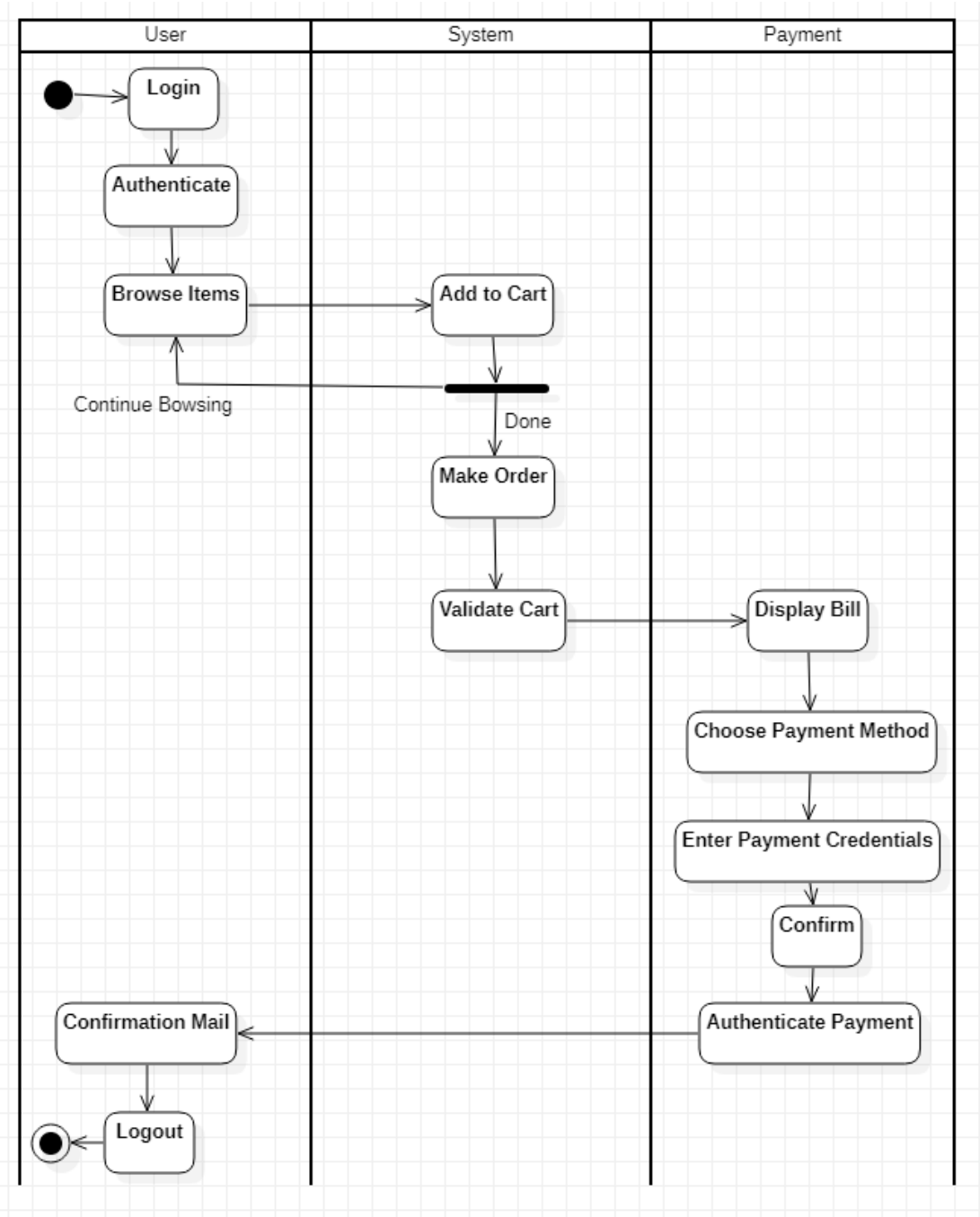

Fig 5.9

The advanced activity diagram shows the activities involved using swim lanes.

The swim lanes provide information about the persons responsible for the activities in the system.

The respective persons activities are shown under their swim lanes. In the diagram shown, user, system and payment are the swim lanes who perform the activities shown in the diagram.

6. Railway Reservation System

Problem Statement:

To design a Railway Reservation System to simplify the process of booking train tickets. The system will allow users to book tickets online. The system can access information regarding train schedules and seat availability and quickly provide tickets and eliminates unnecessary paperwork, travelling and time consumption.

SRS:

Purpose:

The Railway Reservation System is an online system. The system is developed to facilitate faster booking of train tickets. The system allows users to book train tickets online. The system can access information regarding train schedules and seat availability and quickly provide tickets and eliminates unnecessary paperwork, travelling and time consumption. Administrators can edit train schedules through the system as well.

Requirements:

Functional Requirements:

- Provide User Login:
  Users can login and edit account details.
- Provide Facility to book tickets online:
  User can enter trip details and tickets will be provided after system verification.
- Provide Online Payment Facility:
  User can pay for tickets online.
- Provide Admin Login:
  Admin login is required to update train schedules.
- GUI:
  An interface is developed for above mentioned functional requirements.

Non-Functional Requirements:

- Cost:
  The software should be cheap and within budget.
- Time:
  The software product should be completed within deadlines
- Security:
  The data should be secure

Class Diagram:

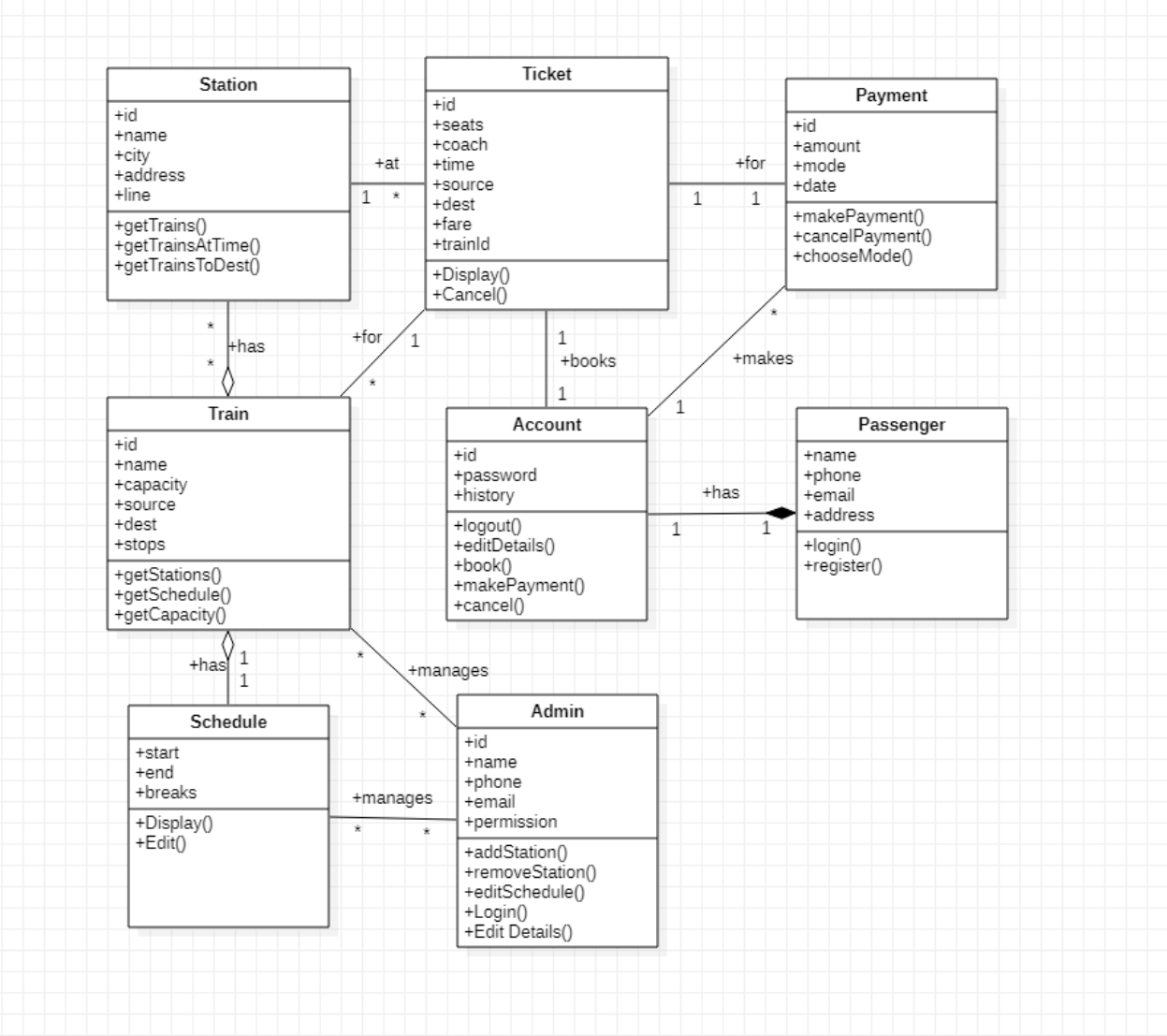

Fig 6.1

- Station: Contains information about station.
- Ticket: Contains information about ticket.
- Payment: Contains information about and manages payment.
- Train: Contains information about train.
- Schedule: Contains information about schedule.
- Account: Allows passenger to interact with system such as booking tickets and making payments.
- Passenger: Contains information about passenger.
- Admin: Manages schedule.

State Diagram:

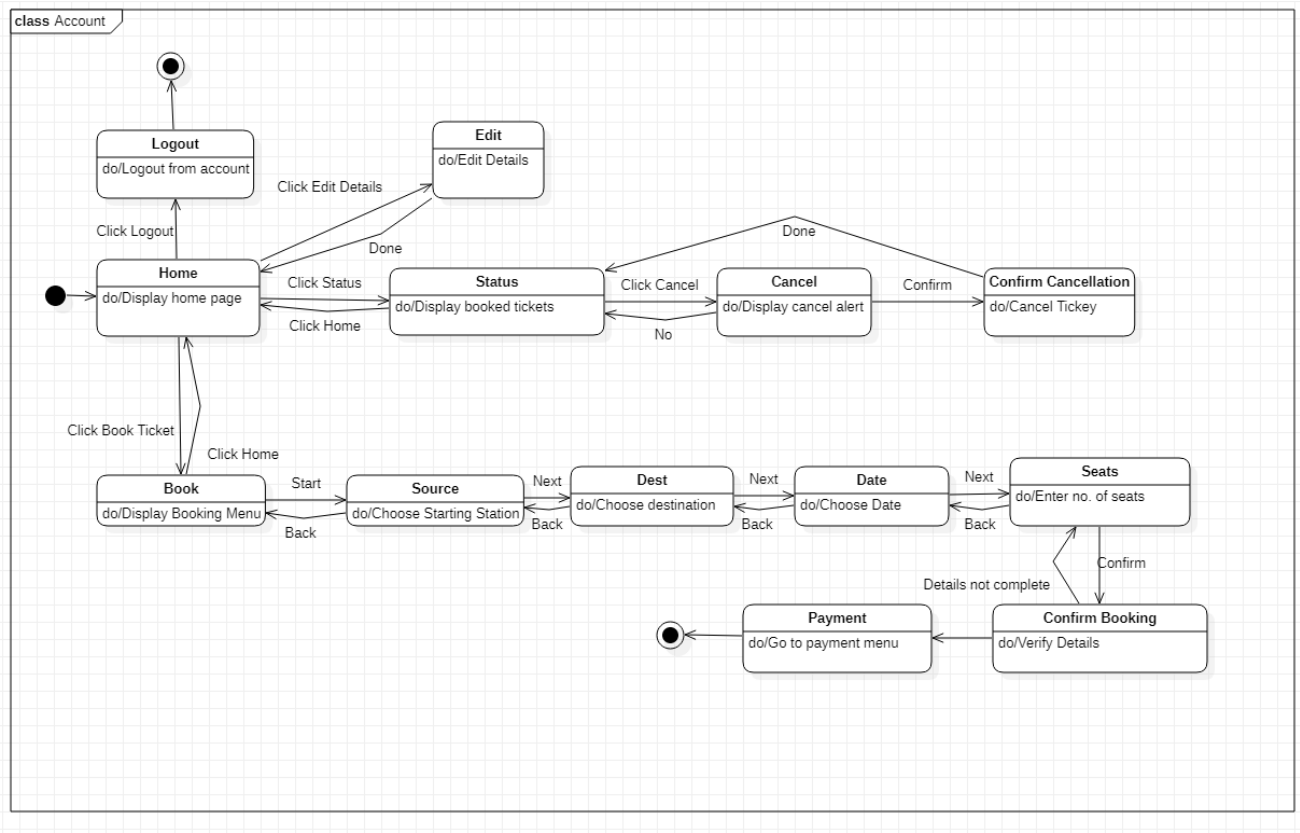

Fig 6.2

The state diagram shows options available to account.
First, you are shown the home page.
You can logout or edit account details.
You can see the status of booked tickets and can cancel them if you want.
When booking tickets, you enter source, destination, date and number of seats required before confirming. If details are completed, booking is verified and you are taken to payment menu.

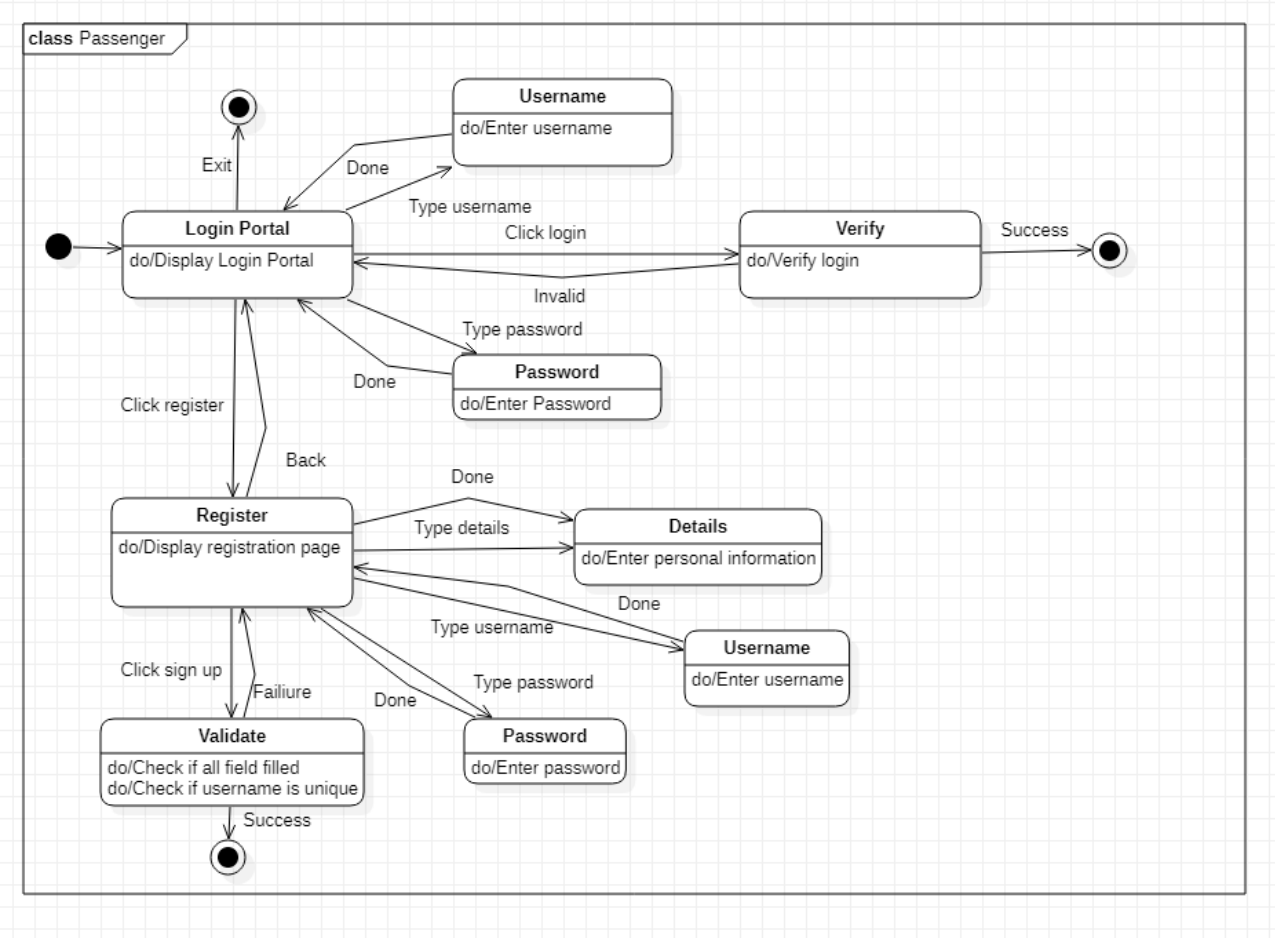

Fig 6.3

The state diagram shows options available to passenger.
First they are shown the login portal. They can directly exit from here.
They can enter username and password. Then Login. If successful, they leave login portal and access system.
When registering, you enter personal details, username and password.
Then details are verified and you can access system in successful.
If invalid, you are taken back.

Use Case Diagrams:

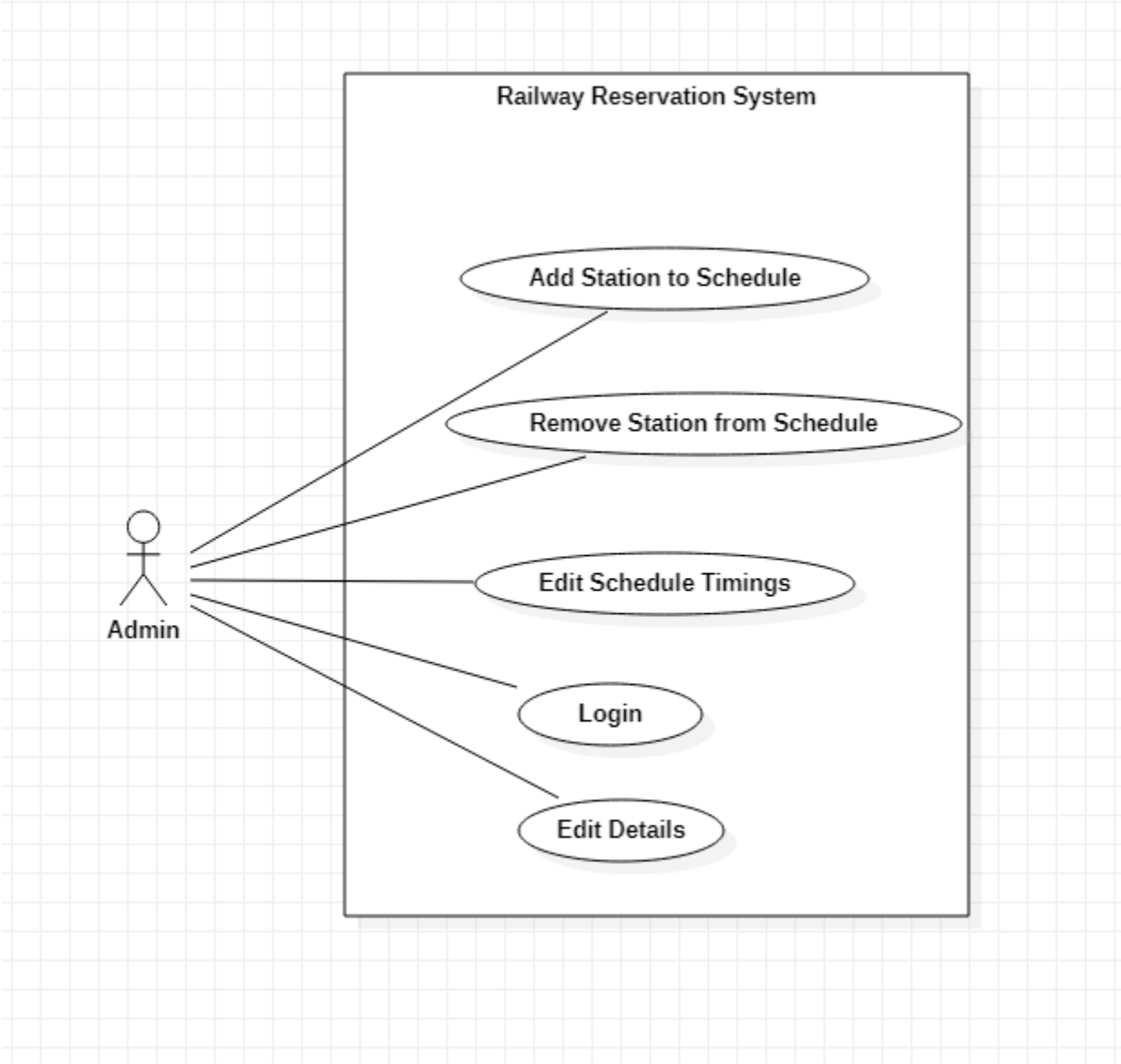

Fig 6.4

There is one actor:

- **Admin:** Person who manages train schedules

There are five use cases:

- **Add Station to Schedule:** Add station to schedule of train
- **Remove Station from Schedule:** Remove station from schedule of train
- **Edit Schedule Timings:** Edit timings in schedule of train
- **Login:** Login to railway reservation database
- **Edit Details:** Edit account details

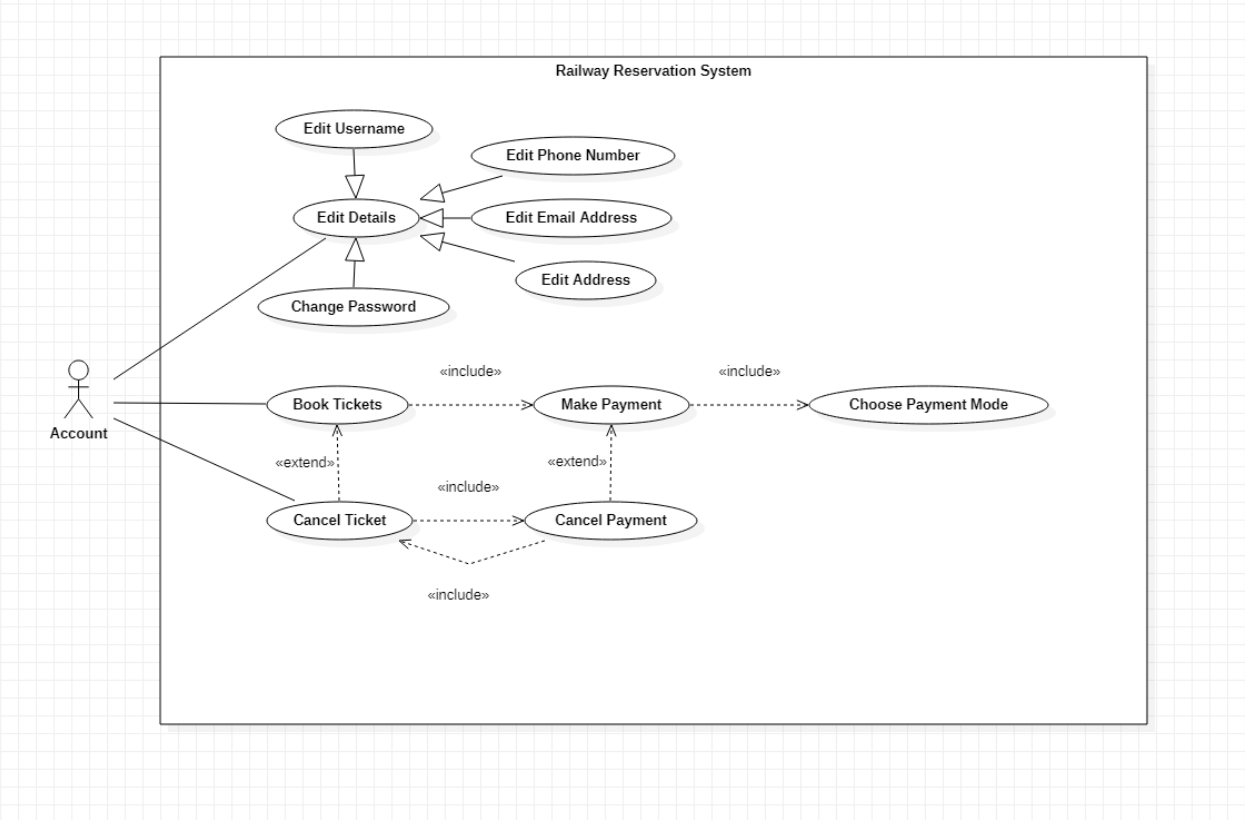

Fig 6.5

There is one actor:

- **Account:** Created by user to book tickets

There are eleven use cases:

- **Edit Details:** Edit account details
- **Edit Username:** Edit account username
- **Edit Phone Number:** Edit account phone number
- **Edit Email Address:** Edit account email address
- **Edit Address:** Edit account address
- **Change Password:** Change password for account
- **Book tickets:** Confirm booking of tickets
- **Make Payment:** Make payment for tickets
- **Cancel Ticket:** Cancel booked tickets
- **Cancel Payment:** Cancel payment for tickets
- **Choose Payment Mode:** Choose payment mode for booking

Sequence Diagrams:

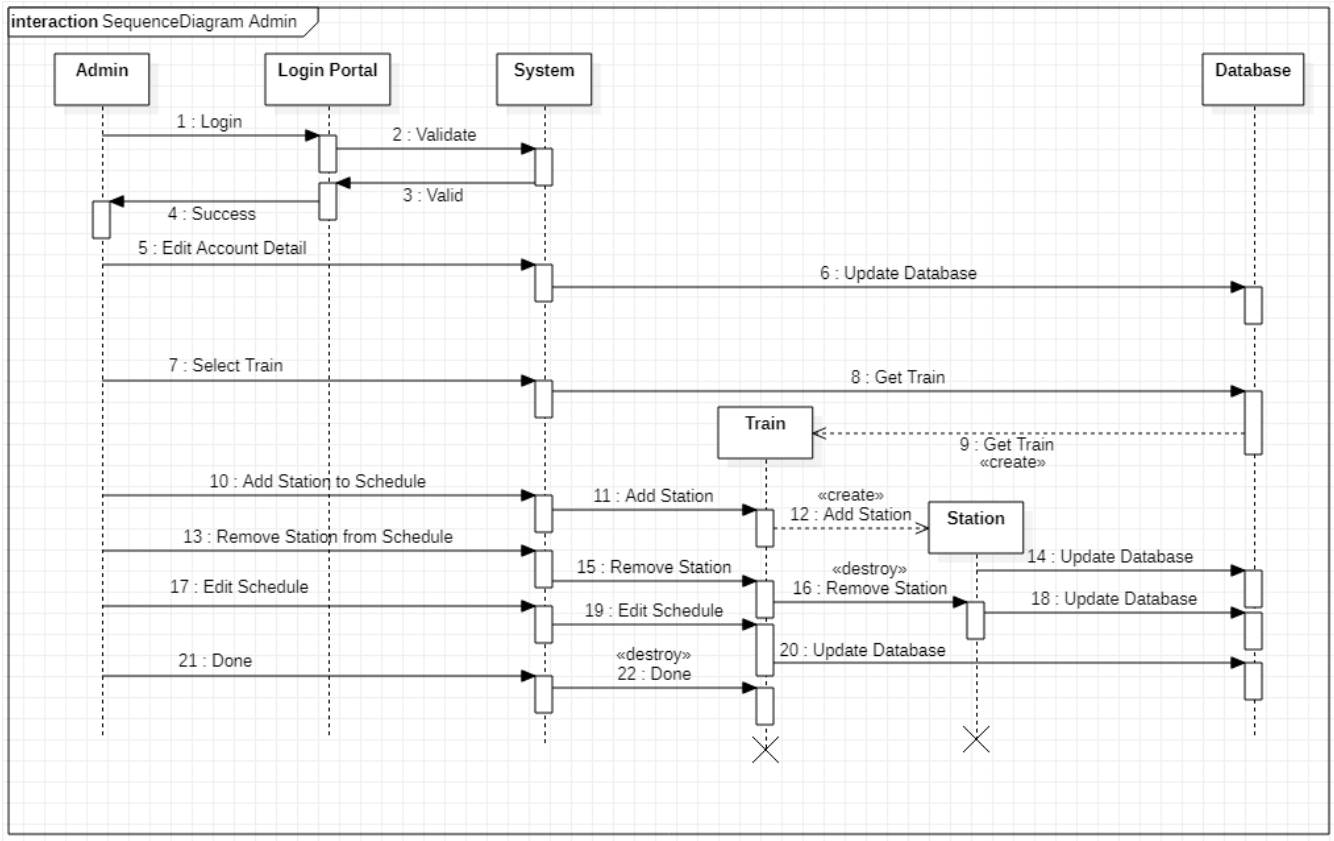

Fig 6.6

The sequence diagram shows the interaction of admin with system in order to manage train schedules.

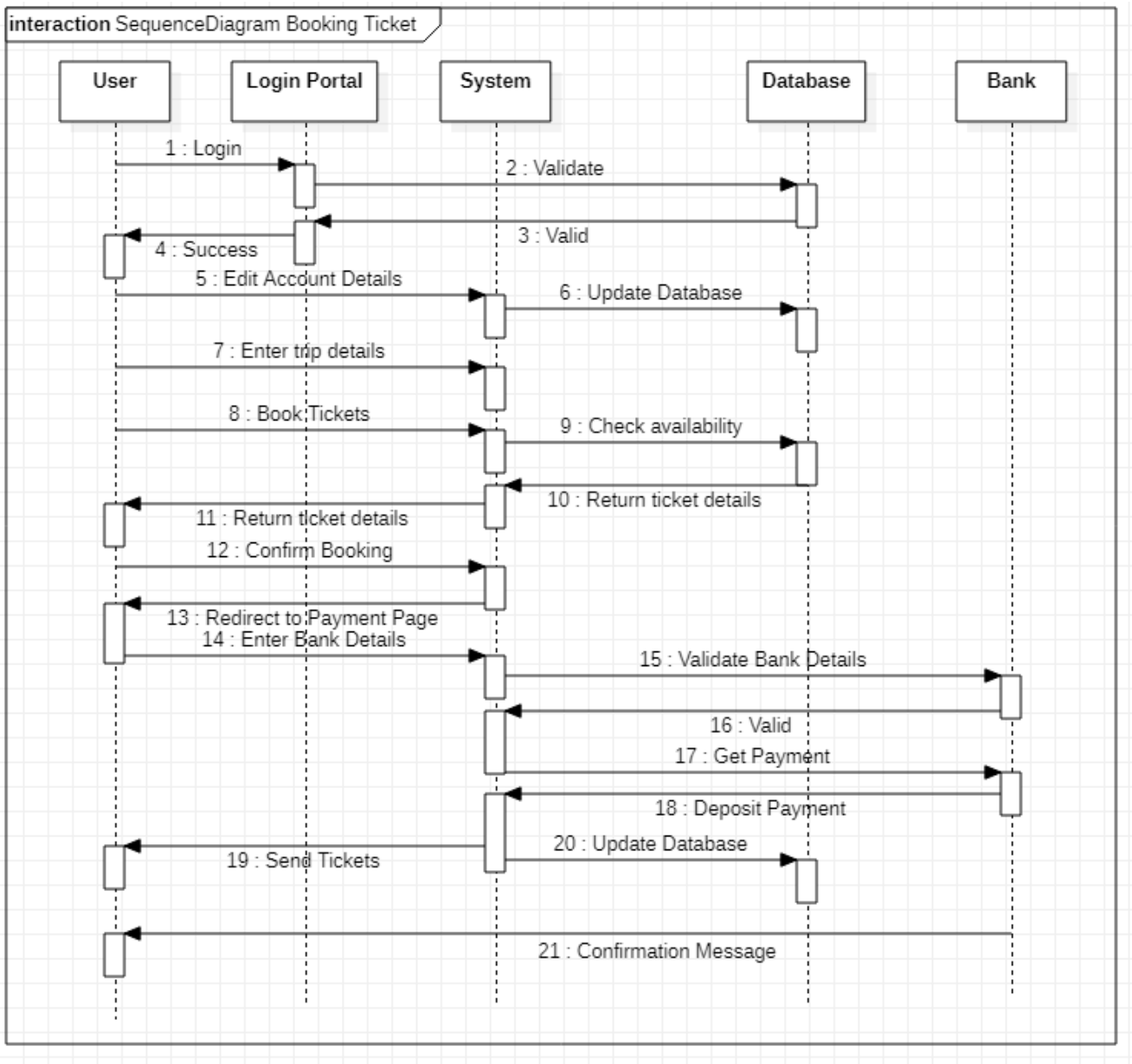

Fig 6.7

The sequence diagram shows the interaction of user with system and bank in order to book tickets and make payments.

Activity Diagrams:

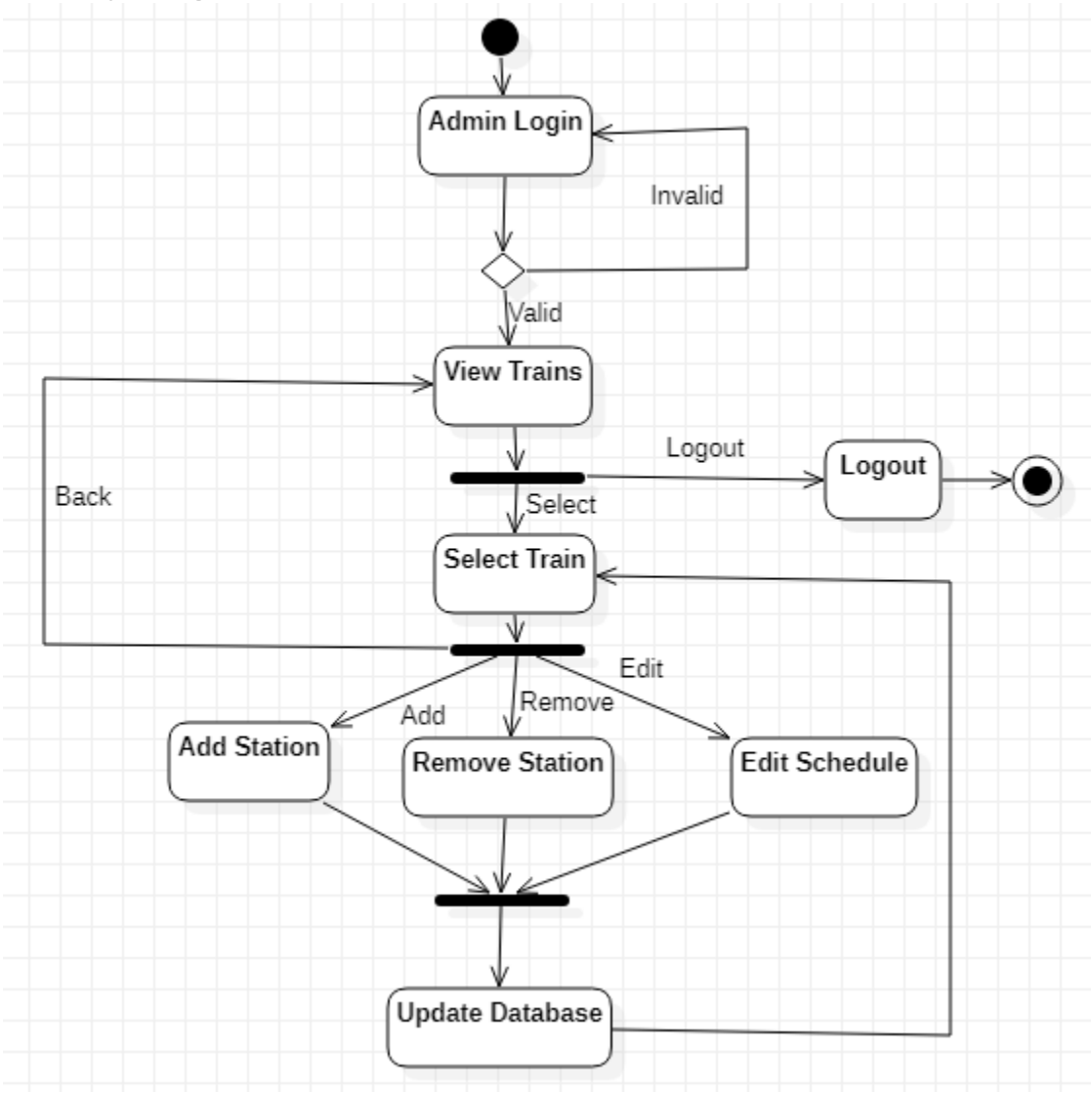

Fig 6.8

The activity diagram shows the activities involved in schedule management by admin.

First they need to login, then they can view and select trains and add or remove stations from schedule or edit timings. Then database is updated.

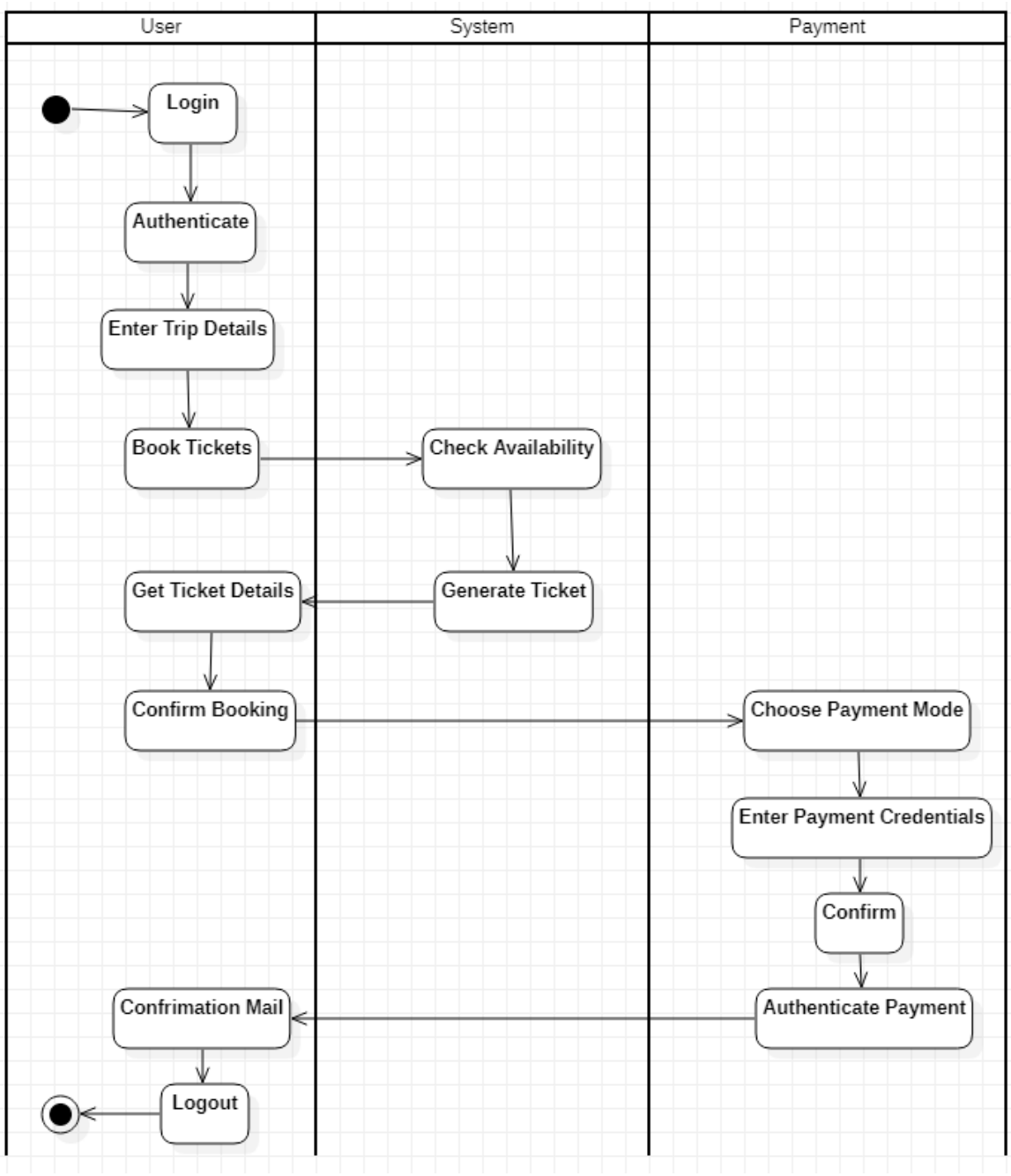

Fig 6.9

The advanced activity diagram shows the activities involved using swim lanes.

The swim lanes provide information about the persons responsible for the activities in the system.

The respective persons activities are shown under their swim lanes. In the diagram shown, user, system and payment are the swim lanes who perform the activities shown in the diagram.

7. Graphics Editor

Problem Statement:

To design a Graphic Editor to allow user to represent information using graphics. The editor will allow users to draw, edit text, color, etc. using tools provided. It allows for quick and easy graphical representation of data.

SRS:

Purpose:

The Graphic Editor is a visual editing software. The Graphics Editor is developed to allow users to visually represent data using graphics. The editor allows users to draw, edit text, color, etc. using tools provided. Various inbuilt templates are provided to represent information. It allows for quick and easy graphical representation of data.

Requirements:

Functional Requirements:

- Provide Document Options:
  Allows users to open new or existing documents, save documents, print, share or export documents, etc. Document templates can be loaded. Layers can be added and objects can be grouped.
- Provide Toolbox:
  Tools can be selected from toolbox to draw objects. Tools can be customized according to user preference.
- Provide Palette:
  Colors can be selected from palette. Palette can be customized according to user preference.
- Provide Shapes:
  User can add predefined shapes to document.
- GUI:
  An interface is developed for above mentioned functional requirements.

Non-Functional Requirements:

- Cost:
  The software should be cheap and within budget.
- Time:
  The software product should be completed within deadlines
- Security:
  The data should be secure

Class Diagram:

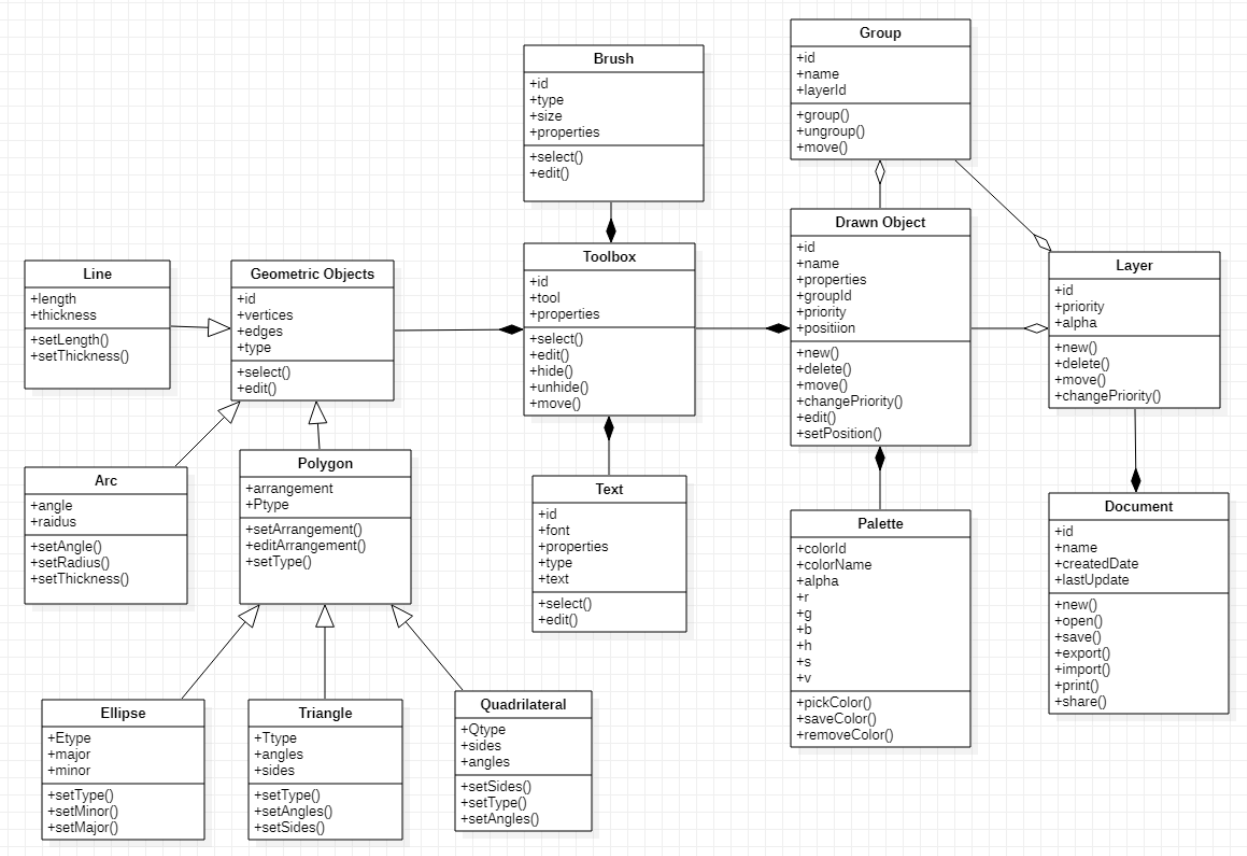

Fig 7.1

- Line: Allows modification of line object.
- Arc: Allows modification of arc object.
- Ellipse: Allows modification of ellipse object.
- Triangle Allows modification of triangle object.
- Quadrilateral: Allows modification of quadrilateral object.
- Polygon: Allows modification of polygon object.
- Geometric Object: Allows selection and modification of geometric objects.
- Toolbox: Allows selection and modification of tools.
- Palette: Allows selection and modification of colors.
- Brush: Allows selection and modification of brush.
- Text: Allows selection and modification of text.
- Drawn Object: Allows drawing and editing of objects.
- Group: Allows grouping of objects.
- Layer: Manages layers in document.
- Document: Provides options for managing document.

State Diagram:

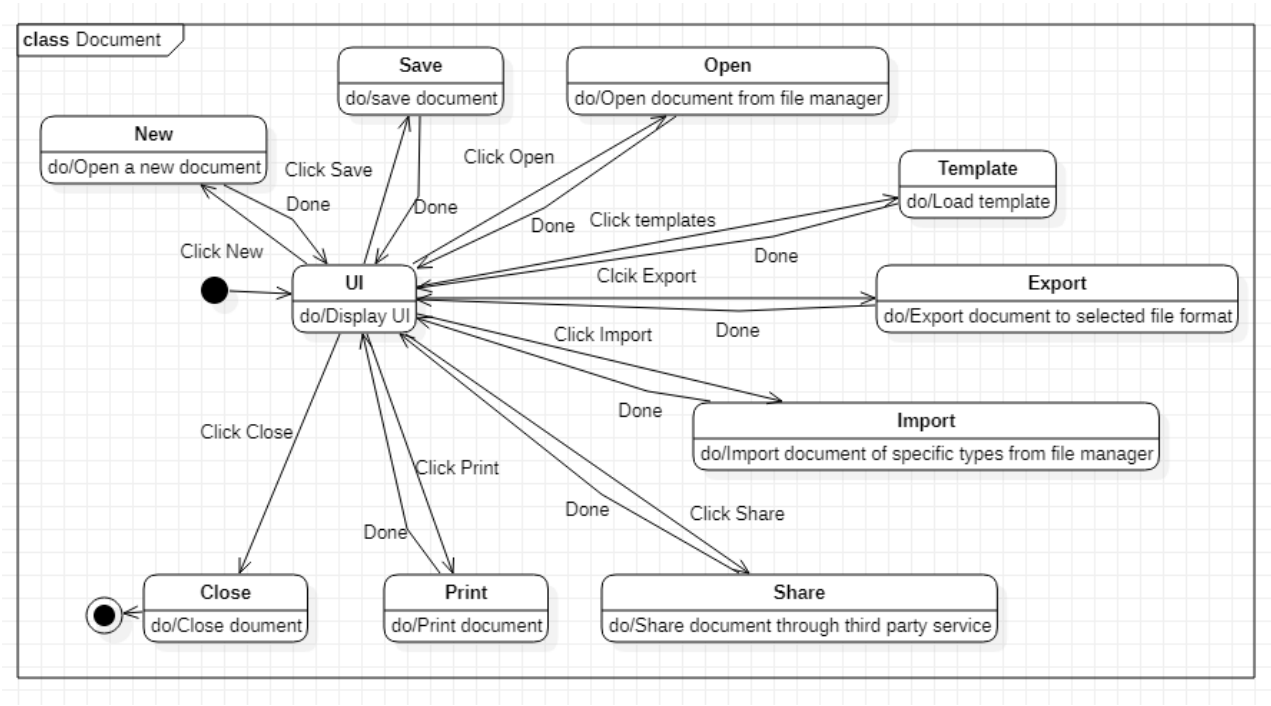

Fig 7.2

The state diagram shows the options available to document.
You can create a new document, open an existing document, save document, load an existing template, export or import document, print or share document or close and exit graphic editor.

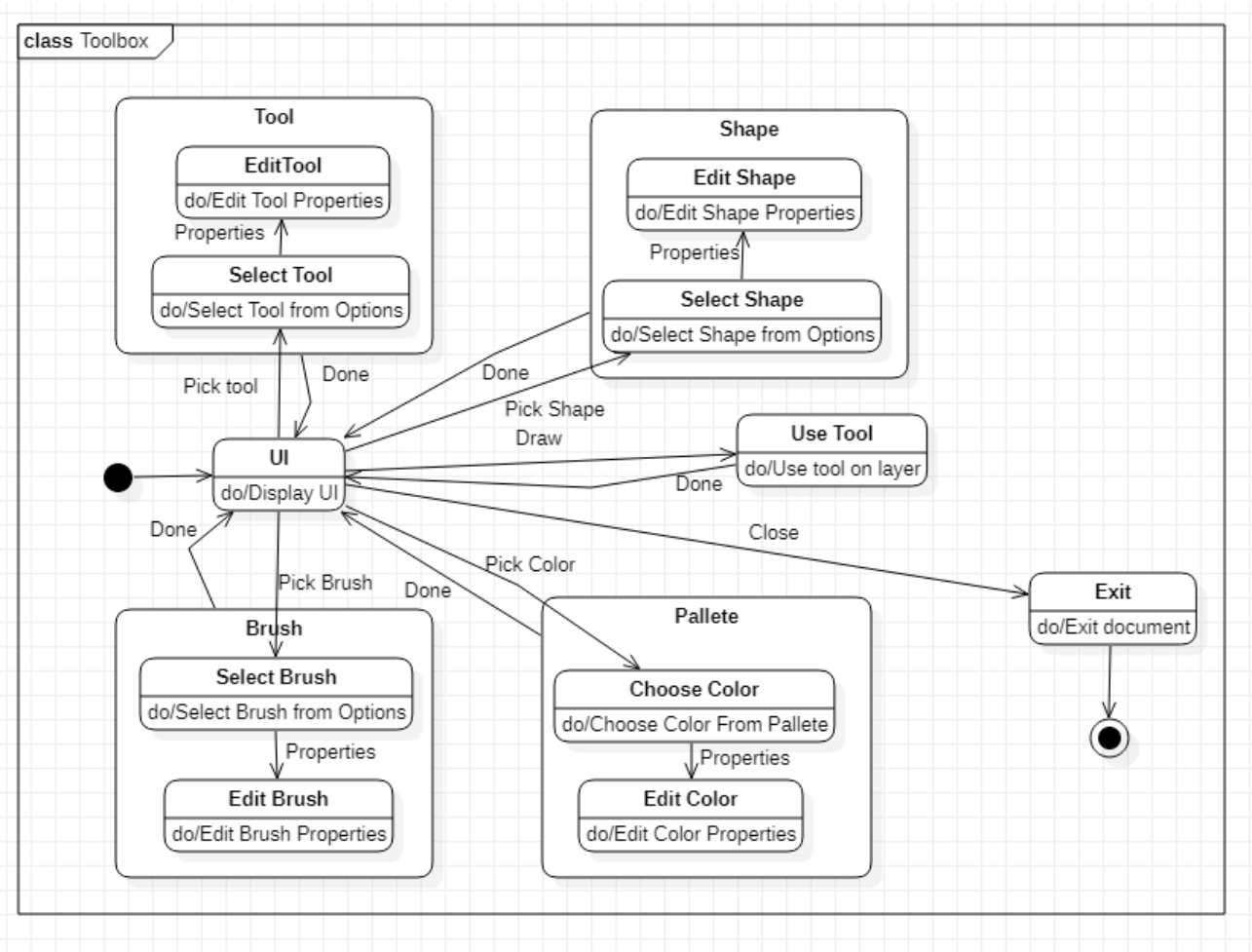

Fig 7.3

The state diagram shows the working of toolbox.

You select a tool and modify its properties.

You select a shape and modify its properties.

You select a brush and modify its properties.

You select a color from palette and modify its properties.

You can use select tool to draw objects.

Use Case Diagrams:

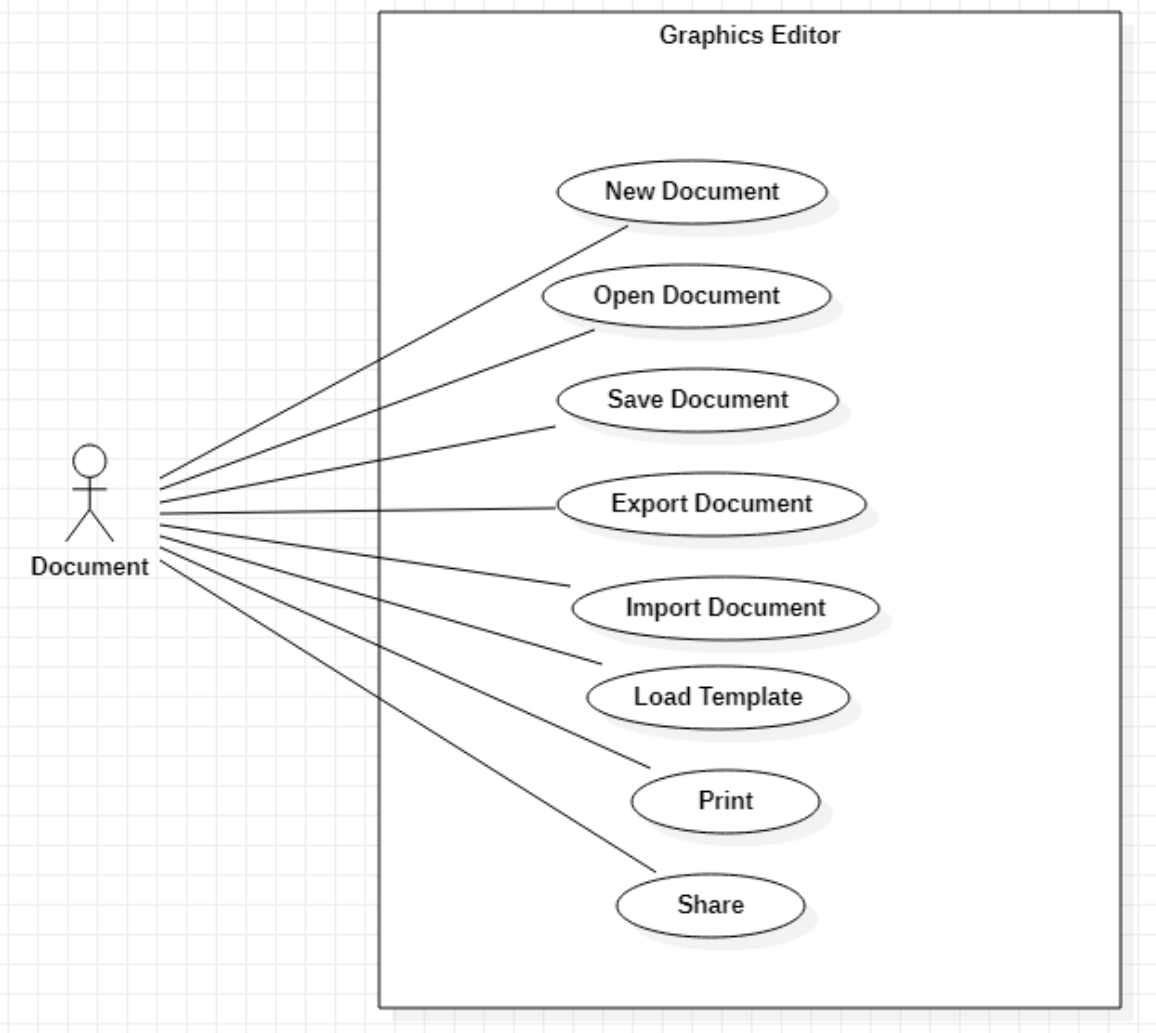

Fig 7.4

There is one actor:

- **Document:** File that stores changes made by user

There are eight use cases:

- **New Document:** Create a new document
- **Open Document:** Open am existing document
- **Save Document:** Save document on computer
- **Export Document:** Export document to other file formats
- **Import Document:** Import document from other file formats
- **Load Template:** Load template for document
- **Print :** Print document
- **Share:** Share document with others

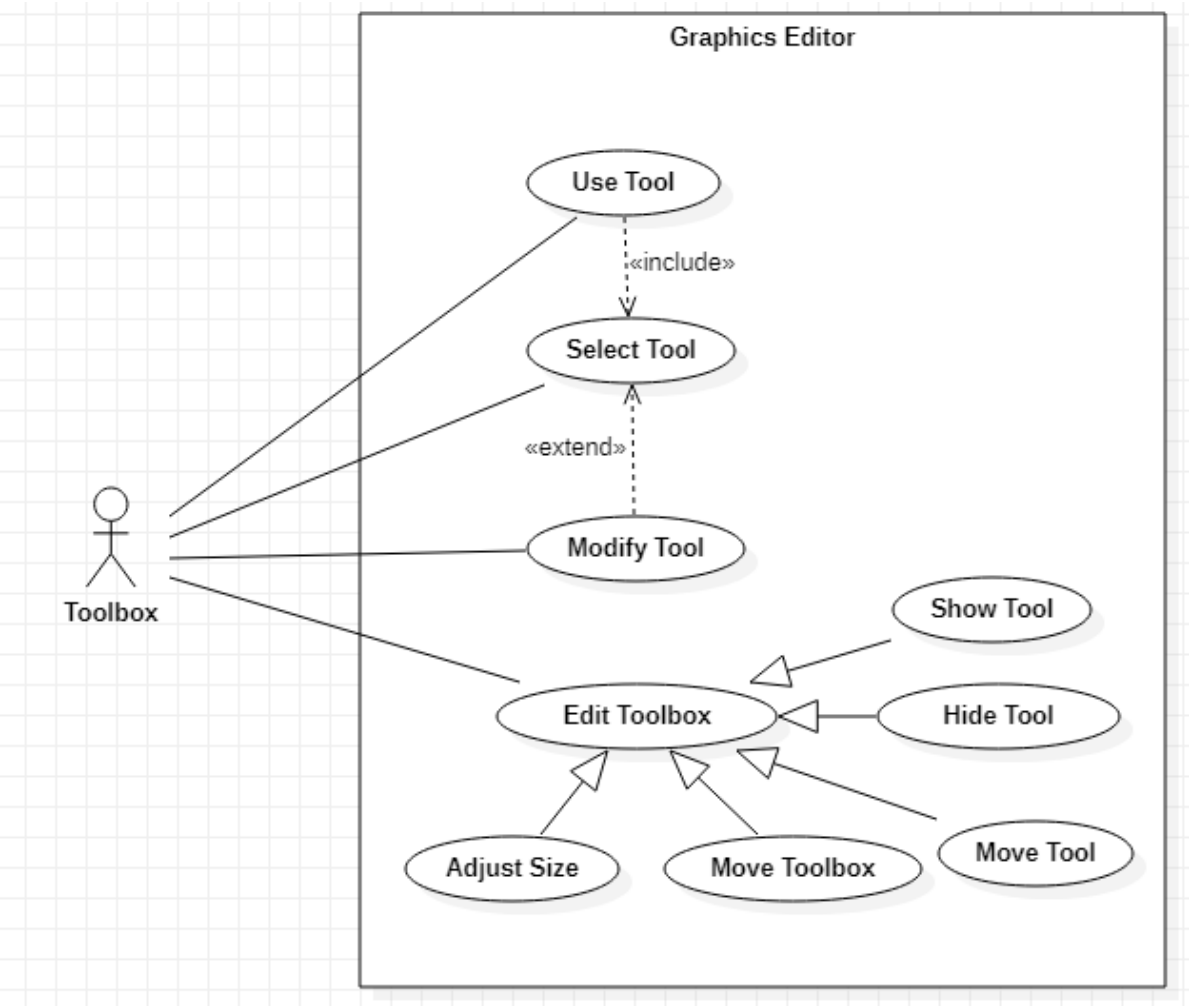

Fig 7.5

There is one actor:

- **Toolbox:** UI interface that provides options for user to modify document.

There are nine use cases:

- **Use Tool:** Use selected tool
- **Select Tool:** Select tool from toolbox
- **Modify Tool:** Modify selected tool settings
- **Edit Toolbox:** Edit toolbox UI
- **Show Tool:** Show tool in toolbox
- **Hide Tool:** Hide tool in toolbox
- **Move Tool:** Move tool in toolbox
- **Move Toolbox:** Move toolbox on screen
- **Adjust Size:** Adjust size of toolbox

Sequence Diagrams:

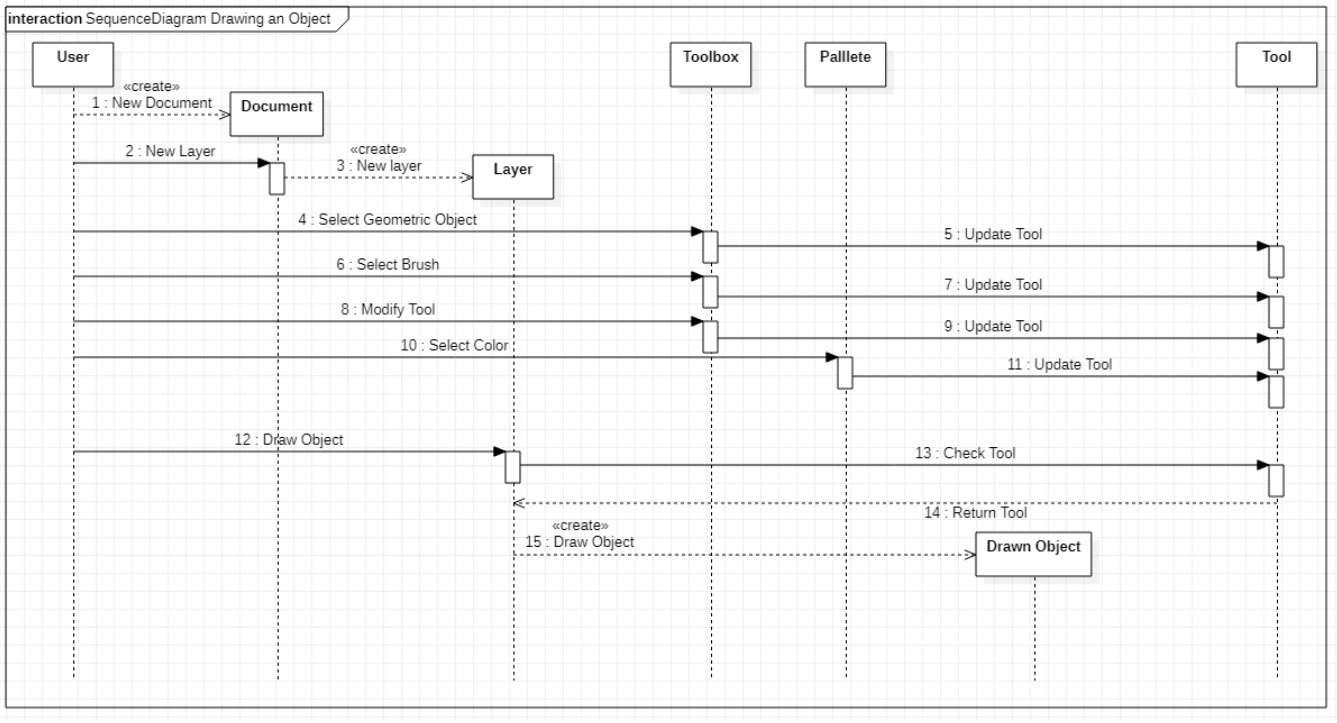

Fig 7.6

The sequence diagram shows the interaction of user with document, toolbox and palette in order to select and modify tools and draw objects on layers.

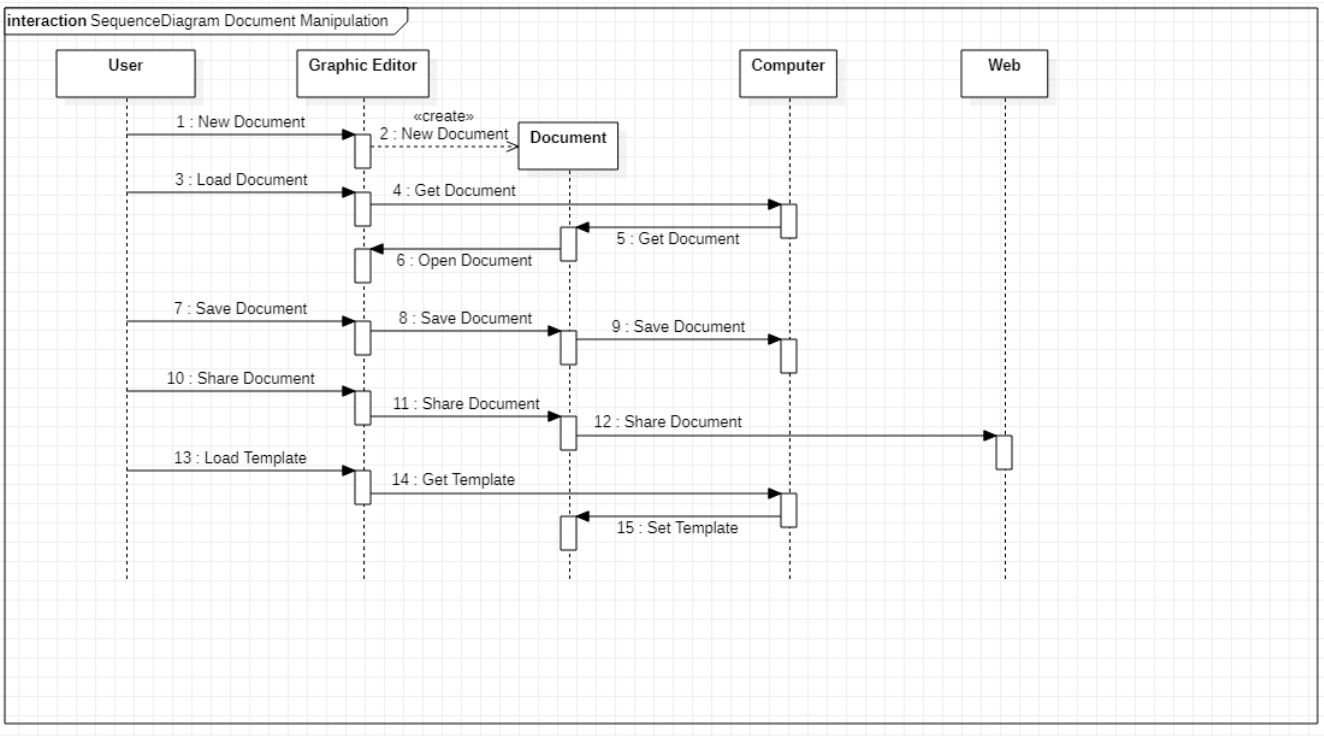

Fig 7.7

The sequence diagram shows the interaction of user with graphic editor in order to use the various options available to use with documents.

Activity Diagrams:

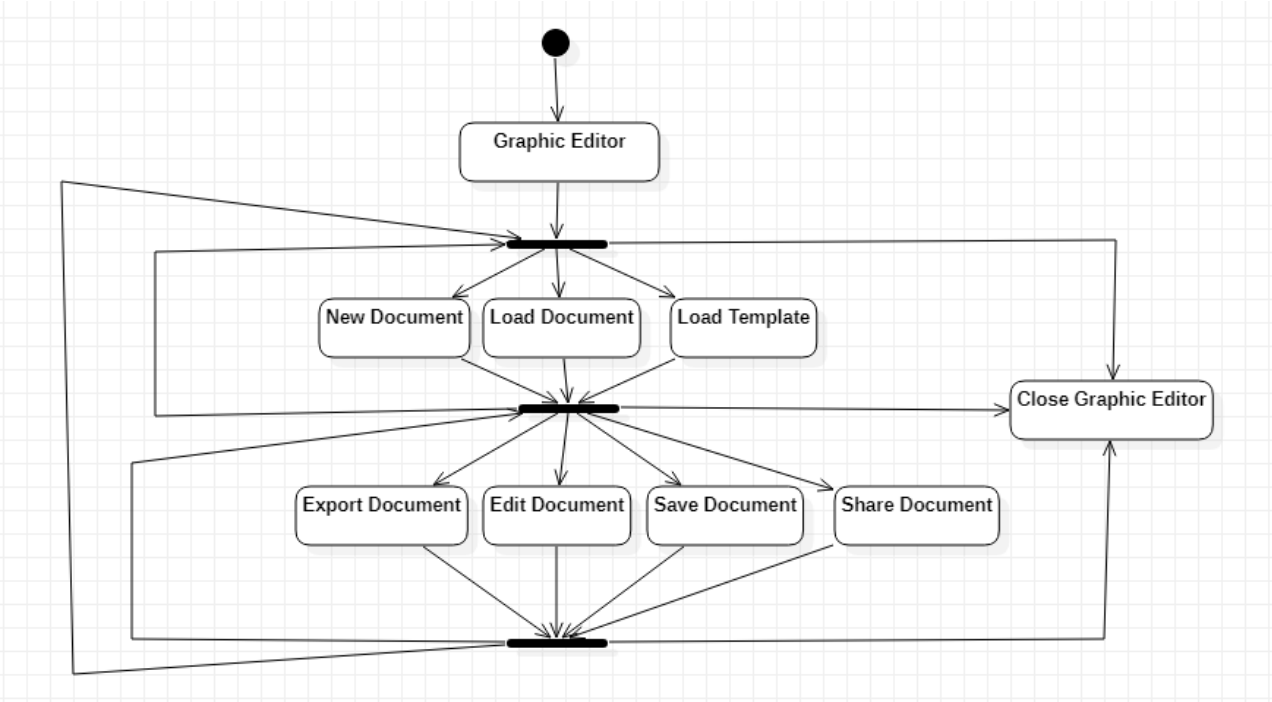

Fig 7.8

The activity diagram shows the activities involved in document management using graphics editor.

You can create a new document, load an existing document or template.

Then you can export, edit, save or share document.

Graphic editor can be closed at any time.

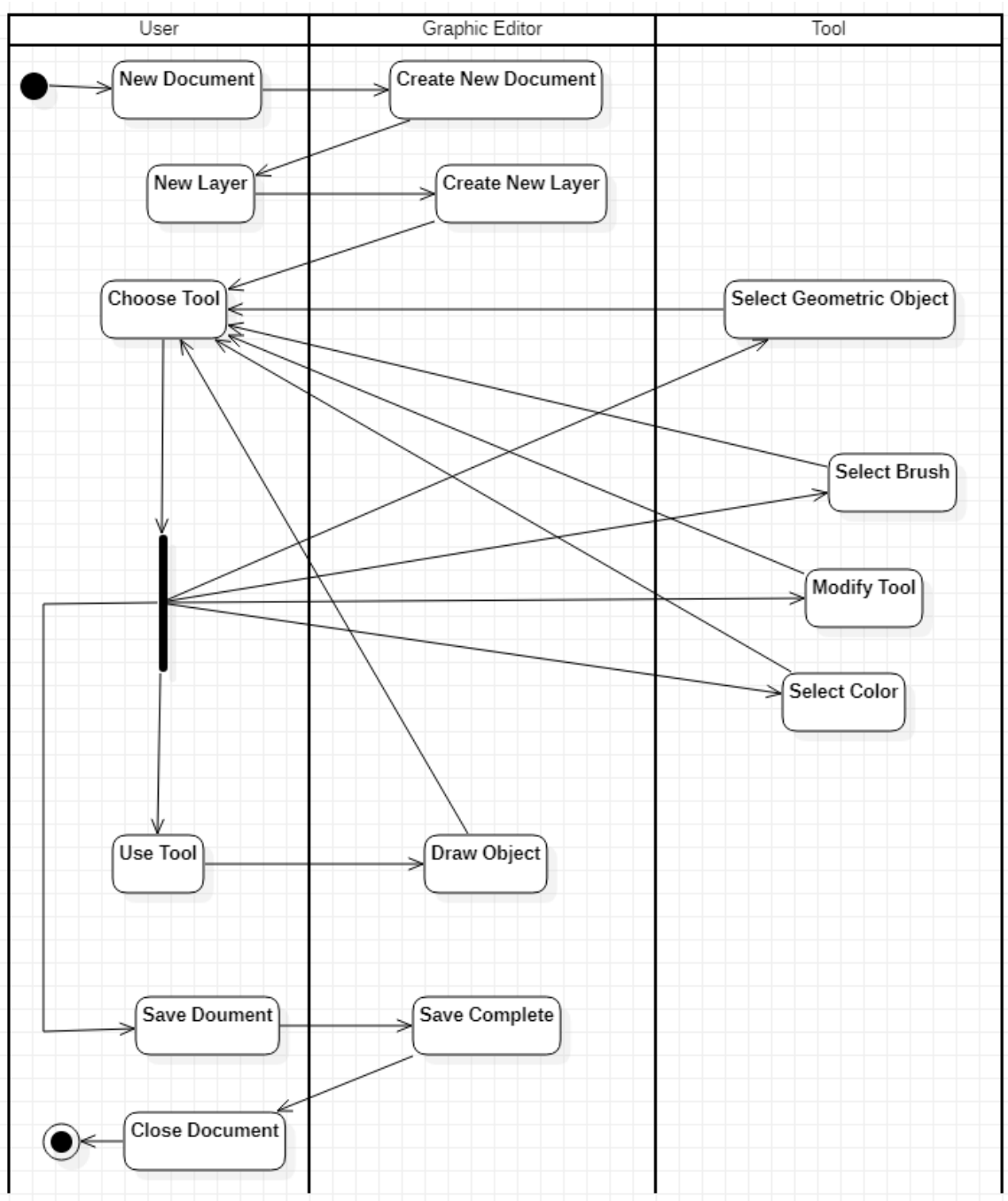

Fig 7.9

The advanced activity diagram shows the activities involved using swim lanes.

The swim lanes provide information about the persons responsible for the activities in the system.

The respective persons activities are shown under their swim lanes. In the diagram shown, user, graphic editor and tool are the swim lanes who perform the activities shown in the diagram.