# 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 6$^{th}$ Semester Laboratory*

Bachelor of Technology
in
Computer Science and Engineering

*Submitted by:*

**S.K.BALAJI**

1BM19CS134

Department of Computer Science and Engineering

B.M.S. College of Engineering
Bull Temple Road, Basavanagudi, Bangalore 560 019
Mar-June 2021

# 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 S K BALAJI(1BM19CS134) during the 6th Semester Mar-June-2021.

Signature of the Faculty Incharge:

NAME OF THE FACULTY:

Sheetal V. A
Assistant Professor at,
Department of Computer Science and Engineering
B.M.S. College of Engineering, Bangalore

# Table of Contents

# 1. College Information System

**Problem statement:**

Design UML diagrams for College Information System with system requirements

specification. **Software Requirements Specification (SRS):**

A centralized approach and system for managing, storing, accessing and updating all the information and details present in relevance to students, and teaching and non-teaching faculty, increasing efficiency and convenience of information management in educational institutions.

 ● Educational institutions should be able to add, edit and view student personal details, like name, age, gender, email, phone number, address and so on.

 ● Educational institutions should be able to add, edit and view student academic details, like USN, department, semester and registered courses.

 ● Faculty should be able to view all student personal details, and should be able to view and edit internal evaluation marks and attendance of students.

 ● The COE office should be able to view all student details, and view and edit internal and examination marks, and publish results.

 ● Placement section should be able to view all student details, and add companies coming to the campus for placements.

 ● Management section should be able to view, add and edit teaching and non-teaching staff details.

 ● Students should not be allowed to edit their personal or academic details. ● The

 system should be convenient and easy to use by students, management and faculty.

 ● The system should be a reliable source of information viewing (most importantly, academic grades) for students, COE and faculty.

**Class Diagram:**

The below shown class diagram contains the following classes: Admin, Department, Staff, Course, Teaching, Non-Teaching, Placement, COE, Student, Hostel, Facility, Library with multiplicities as shown.
Association: Admin handles college,department Placement and COE, Student gets Facility, Courses are registered by student.
Generalization: COE,Hostel,Placement,Library,Staff are generalised
verison of the department

Aggregation:Department consists (aggregate of) Student and faculty
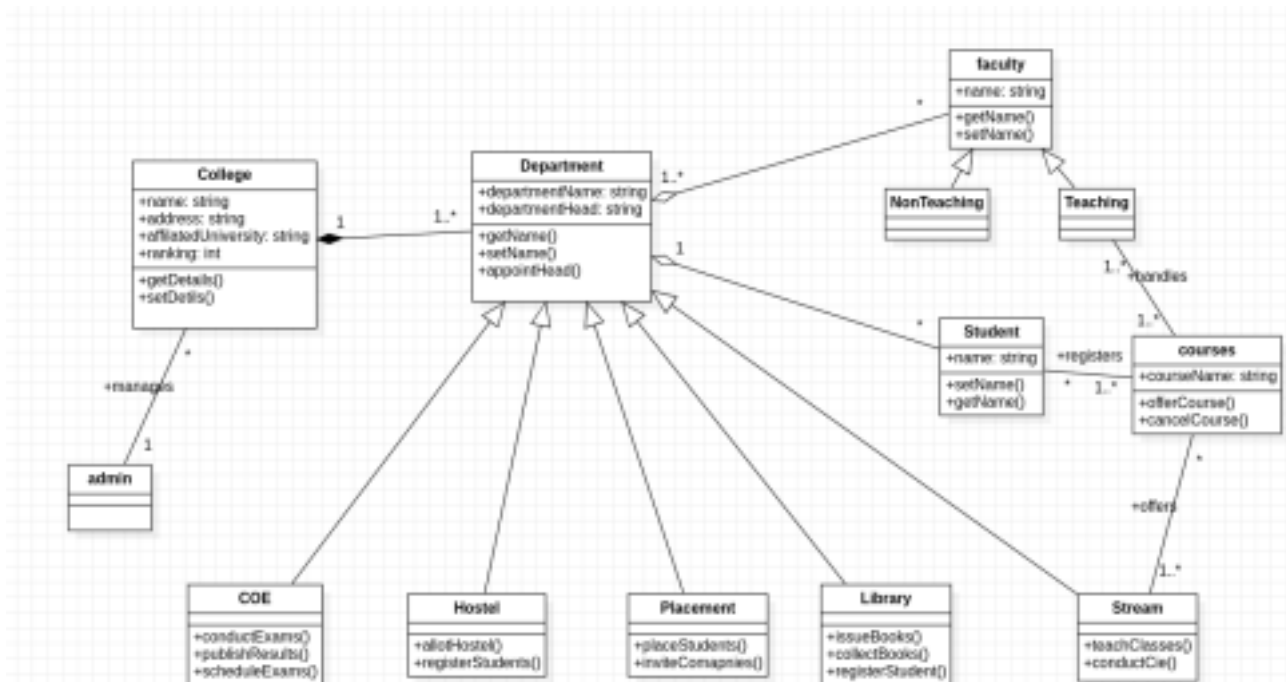Composition: College has (or is composed of) Department



Fig 1

## 1. Advance state diagram

The advanced state diagram depicted below contains one nested state and one submachine, which on successful login shows the course details and profileView procedure of student. It contains initial state and termination state with Courses as a nested state including the required simple states. It also has a submachine state named ProfileView with initial, termination state along with simple states; Display profile, Edit, Save.

**Displaying**
do/DisplayStudentDetails

select Edit

**Edit**
entry/editDetails

exit

select/Save

exit

**save**
entry/saveFields

exit
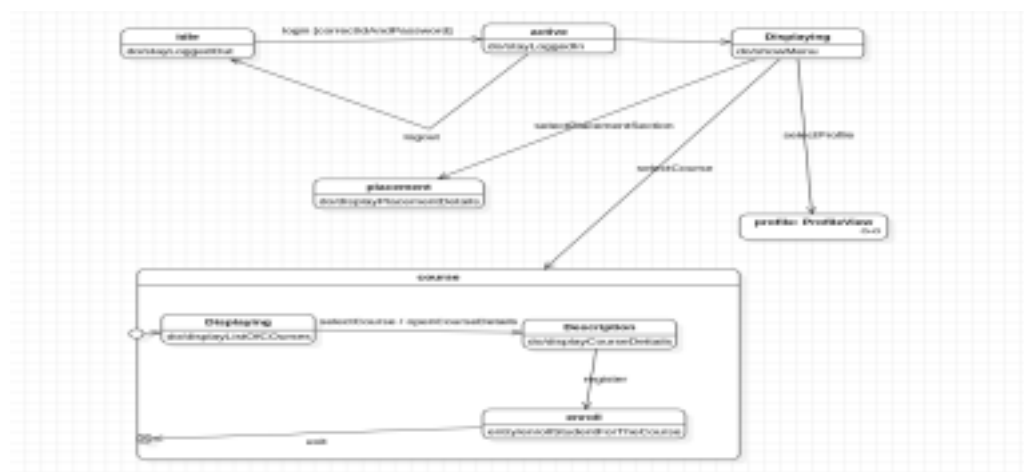
# 2. Hostel Management System

**Problem statement:**

Design UML diagrams for Hostel Management System with system requirements

specification. **Software Requirements Specification (SRS):**

The purpose of the Hostel Management System is to carry out different operations of a hostel. This system will provide ease of use to the staff of the hostel by performing all work on computers. It helps to manage student and staff records.

● Admin can login using credentials provided to him.

● Admin can allot room to students.

● Students can login using the credential provided and can give feedback about

staff. ● Admin can review the feedback provided by students.

● Admin can appoint staff.

● Students can provide message feedback.

● Mess managers can review the mess feedback.

● Mess manager can update the menu list.

● Admin can assign work to staff members.

● The system should be easy to handle.

● System should give expected performance results.

● The response time should be small.

**Class Diagram:**

The below shown class diagram contains the following classes: Person, Student, Administrator, Warden, Hostel, Rooms_allocation, Receipt_generation, Rooms, Bed with multiplicities as shown.
Association: Admin handles hostel,hostelitie stays in hostel,warden manges hostel

Generalization: Person is generalization
Aggregation:Hostel is composed of Mess and rooms
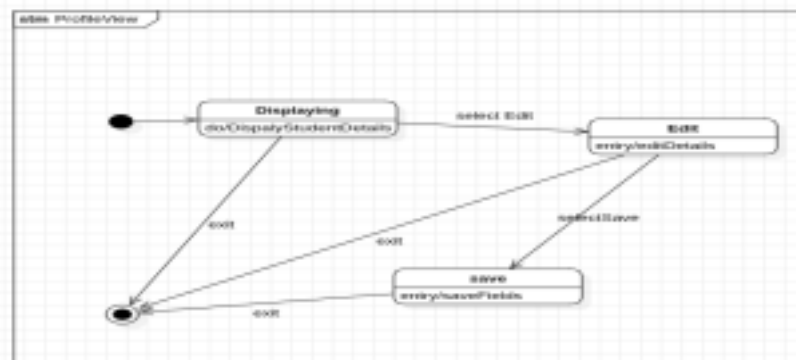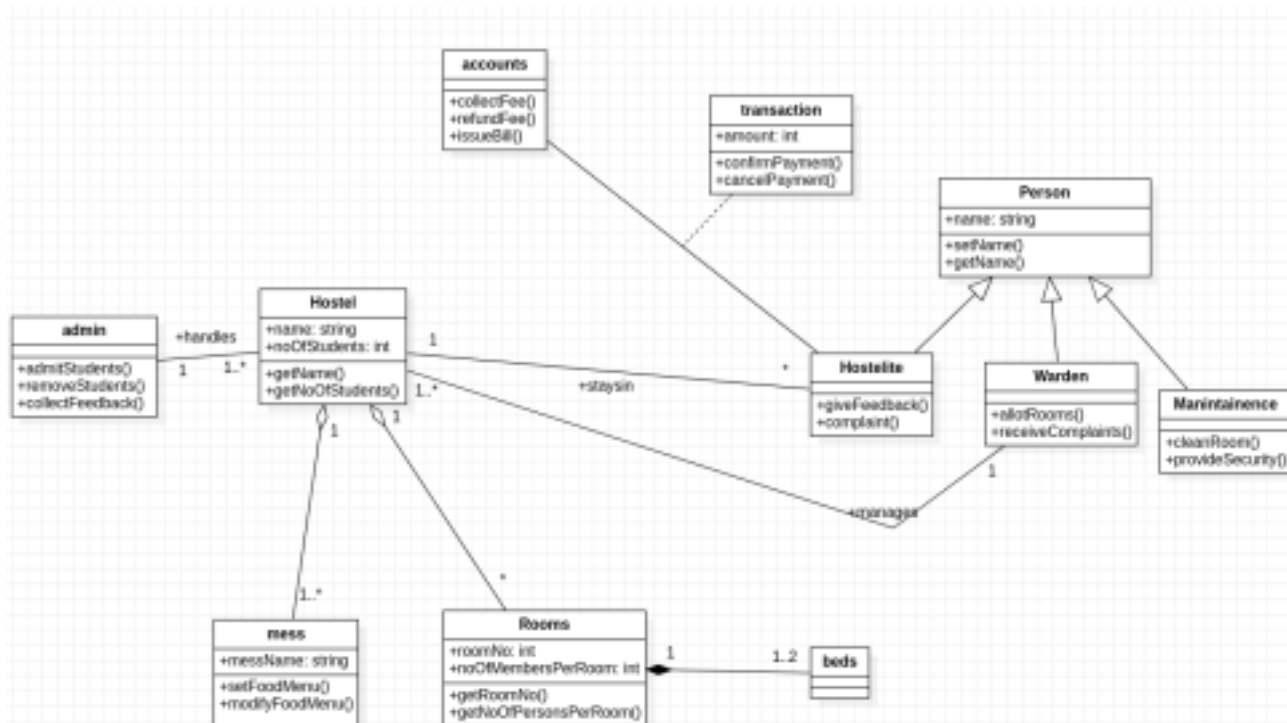Composition: Rooms has (or is composed of) beda

### 1. Advance state diagram

The advanced state diagram depicted below contains one nested state and one submachine, which on successful login shows the mess details and room allocation procedure. It contains initial state and termination state with Mess as a nested state including the required simple states. It also has a submachine state named RoomAllocation with initial, termination state along with simple states; Blocks, Rooms, Allocate,Payment.
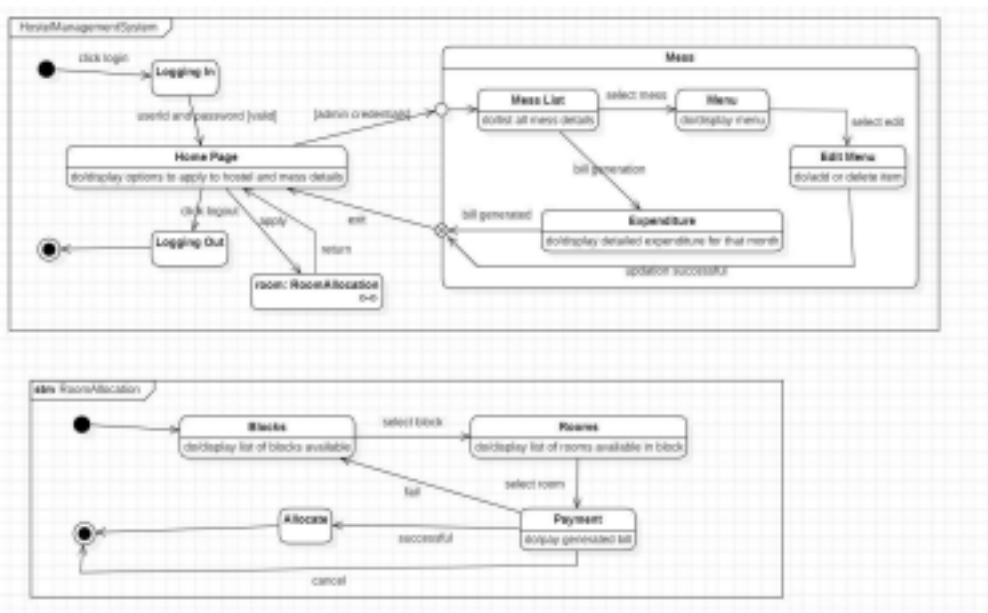
# 3. Stock Maintenance System

**Problem statement:**

Design UML diagrams for Stock Maintenance System provided with system requirements specification.

**Software Requirements Specification (SRS):**

The stock maintenance system will allow the employees to record information of the items available in the store and generate reports based on the total amount of sales. The new system will have a windows-based desktop interface to allow employees to enter the information of sales, purchase orders, change employee preferences and create reports. The system retains information on all the items in the shop. The system retains the records of the cost, expiry date, vendor details, discount, quantity. The employee maintains the information of the sale of the item. He can add the items at the right time and update the database. The customer can view the availability of the required items and the price of the items. The customer can just view them but cannot make any changes.

 The process of the stock maintenance system is that the customer logs in to the particular site to place the order for the customer product. The stock maintenance system is described sequentially through steps

• The customer logs in to the particular site.

• They fill the customer details.

• They place the orders for their product.

• The vendor logs in and views the customer details and orders

**Class Diagram:**

The below shown class diagram contains the following classes: Role, Permisssion, Store, User, Stock, Product, Customer and Payment with multiplicities as shown.
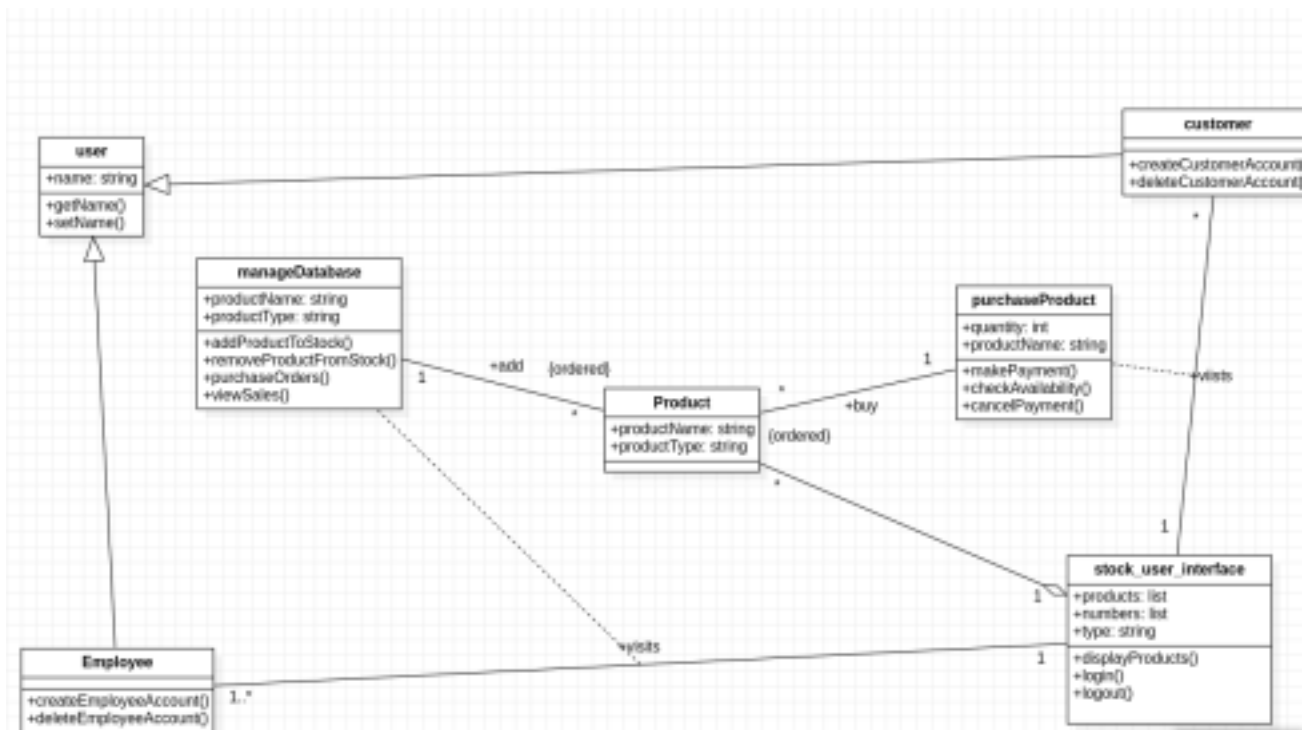
# 1. Advance state diagram

The advanced state diagram depicted below contains one nested state and one submachine, which on successful login shows the StockStatus details and StockPurchase procedure. It contains initial state and termination state with Maintaining as a nested state including the required simple states. It also has a submachine state named StockPurchase with initial, termination state along with simple states; Inventory check, Sell, Payment, Validation.
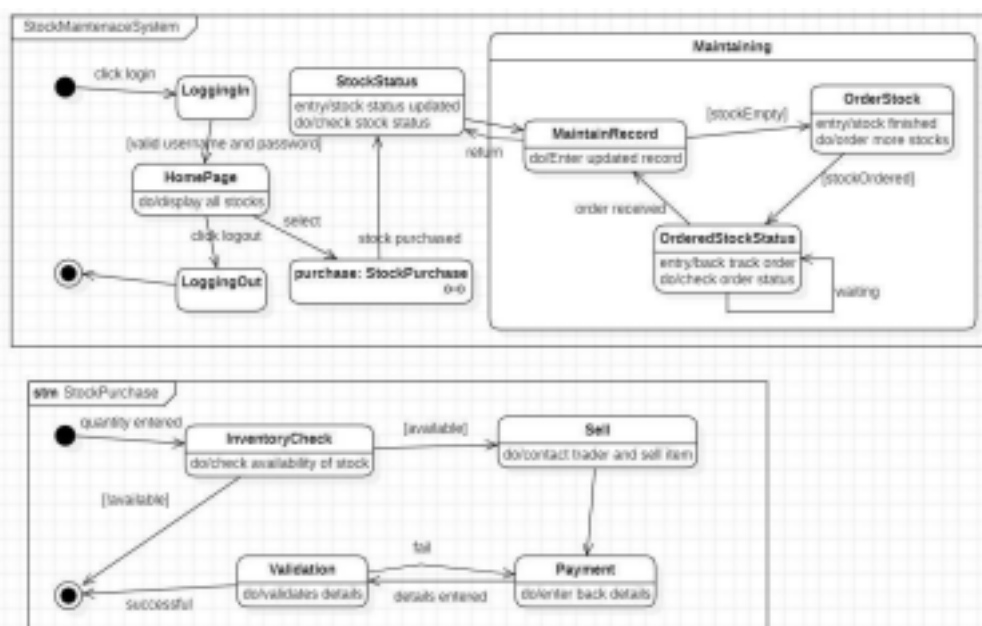
# 4. Coffee Vending Machine

**Problem statement:**

Design UML diagrams for Coffee Vending Machine with system requirements

specification. **Software Requirements Specification (SRS):**

The Objective of the system is to prepare a coffee vending machine for commercial purposes. The system will be able to prepare coffee by processing all its required ingredients. Users will be provided with sophisticated and easy to use user interfaces.

There are many different types of coffee makers using a number of different brewing principles, in the most common devices, coffee grounds are placed in a paper or metal filter inside a funnel, which is set over a glass or ceramic coffee pot, a cooking pot in the kettle family. Cold water is poured into a separate chamber, which is then heated up to the boiling point, and directed into the funnel.

• Cash Box:Knows amount of money put in; Give change; Knows price of coffee; Turns front panel on and off.

• Front panel:Captures selection; Knows what to mix in each; Instructs mixer when to

mix. • Mixer:Knows how to talk to the dispensers.

• Dispenser [cup-, coffee powder-, sugar-, creamer-, water-]:Knows how to dispense a fixed amount, knows when it is empty.

Features:

• Small carbon footprint

• Energy saving advanced power management system

• Comprehensive drink range

• Simple user interface

• One touch

servicing Working:

Coffee vending machines are quite simple and basic. The way they work is not too different to how a tabletop coffee machine or even a drip coffee machine operates. If you think about it, making coffee is simply adding together coffee beans or grounds to hot water and mixing with milk and sugar, that's exactly what a hot drink vending machine does.

Functions:

• Add heat:to heat the coffee we have 3 options. We could use a heating element where the water gravity fed into a tubular heating element, external to the water reservoir, and boiled out. Secondly, we could use a submersible heating element placed inside of the water reservoir to heat all of the water at once. Thirdly, we could use an external hot plate to heat one or multiple walls of the water reservoir and thus heat the water through surface convection. Direct Water:The fluids could be directed from the water reservoir to their final destination via tubing, gravity reed, and pump.

• Contain Water/Coffee:To contain the water and coffee we could use one reservoir, two reservoirs or a funnel. If one reservoir was used for both the water and coffee container, our design would be a percolating or French press coffeemaker.

• Reduce Noise:To reduce the overall noise we consider two options: noise dampening material and internal brew mechanism. To lessen the noise produced by our designs we could fill or cover the outer shell of a noise dampening material. We could also keep the brew mechanism, whether it is drip spout.

Maintenance: When it comes to the ways in which coffee vending machines work, it's not all about the coffee, it's also about the upkeep and maintenance of the machine. With regular visits, suppliers should empty the cash drawer, reconcile the proceeds against sales, empty the waste grounds, refill ingredients and cups, and generally undertake any work to both the interior and exterior to keep everything running smoothly, such as ensuring there's no build up of dirt around the exterior buttons that could cause them to stick, and making sure nothing is blocking the internal sensors that could prevent some ingredients from being added to the mixing chamber.

**Class Diagram:**

The below shown class diagram contains the following classes: Coffee Machine, Cash Box, Selector, Dispenser Register, Dispenser, Ingredient, Recipe, Product, Product Register. with multiplicities as shown.
Association: Customer buys Product, Customer buys Stocks, Customer pays Payment.
Generalization: User is generalized to Permission class and User is Generalized to Role.
Aggregation: Coffee Machine contains Cash Box, Selectors, Dispense Register has Dispenser, Dispenser is composed of Ingredients.
Composition: Recipe is made up of Ingredients
Enumerations: TypeOfCoffee
**1. Advance state diagram**

The advanced state diagram depicted below contains one nested state and one submachine, which on successful login shows the CollectingMoney procedure and DispenseItem procedure. It contains initial state and termination state with CollectingMoney as a nested state including the required simple states. It also has a submachine state named DispenseItem with initial, termination state along with simple states; SettingRow, SettingColumn, Pouring.
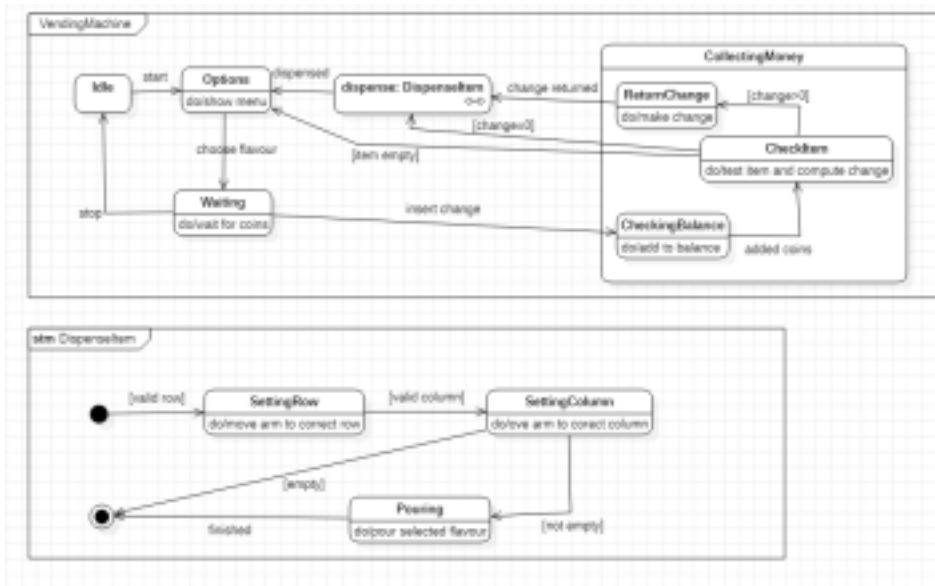
**Coinbox**
+isFull: boolean
+amountCollected: float
+validCoin: boolean
+collectAmount()
+generateBill()
+checkCoin()
+returnCoin()

**Vendor**
+vendor_id: string
+refill()
+collect_money()

**Customer**
+phoneNumber: long int
+email: string
+selectCoffee()
+payBill()
+takeCoffee()
+placeCup()

**MilkDispenser**
+temperature: float
+volume: float
+age: int
+dispenseMilk()
+clean()
+stop()
+calculateVolume()
+measureTemperature()

**User**
+name: string
+display_details()
+update details()

**CoffeeDispenser**
+coffeeType: string
+status: string
+sugarLevel: int
+dispenseCoffee()
+clean()
+stop()

**Item**
+name: string
+hotOrCold: string
+cost: float
+getDetails()

**Menu**
+itemCount: int
+select()
+display()
+update()

**CoffeeMachine**
+firmwareVersion: float
+status: string
+displayMenu()
+dispenseCoffee()
+checkForMug()

**Dispenser**
+status: string
+check_availablity()

**WaterDispenser**
+temperature: float
+volume: float
+dispenseWater()
+clean()
+stop()
+calculateVolume()
+measureTemperature()
+refill()

+has   1   1   +displays   1   1   1   1   +has

1..*   1

**Transaction**
+amount: float
+tax: float
+items: string
+calcTotal()

«enumeration»
**Status**
Active
Inactive
Busy

**VendingMachine**
Idle   start   Options   do/show menu   dispensed
dispense: DispenseItem   change returned   ReturnChange   do/make change   [change>0]

**CollectingMoney**
[change=0]
CheckItem   do/test item and compute change
choose flavour
[item empty]
Waiting   do/wait for coins   insert change   CheckingBalance   do/add to balance   added coins
stop

**stm DispenseItem**
[valid row]   SettingRow   do/move arm to correct row   [valid column]   SettingColumn   do/move arm to correct column
[empty]
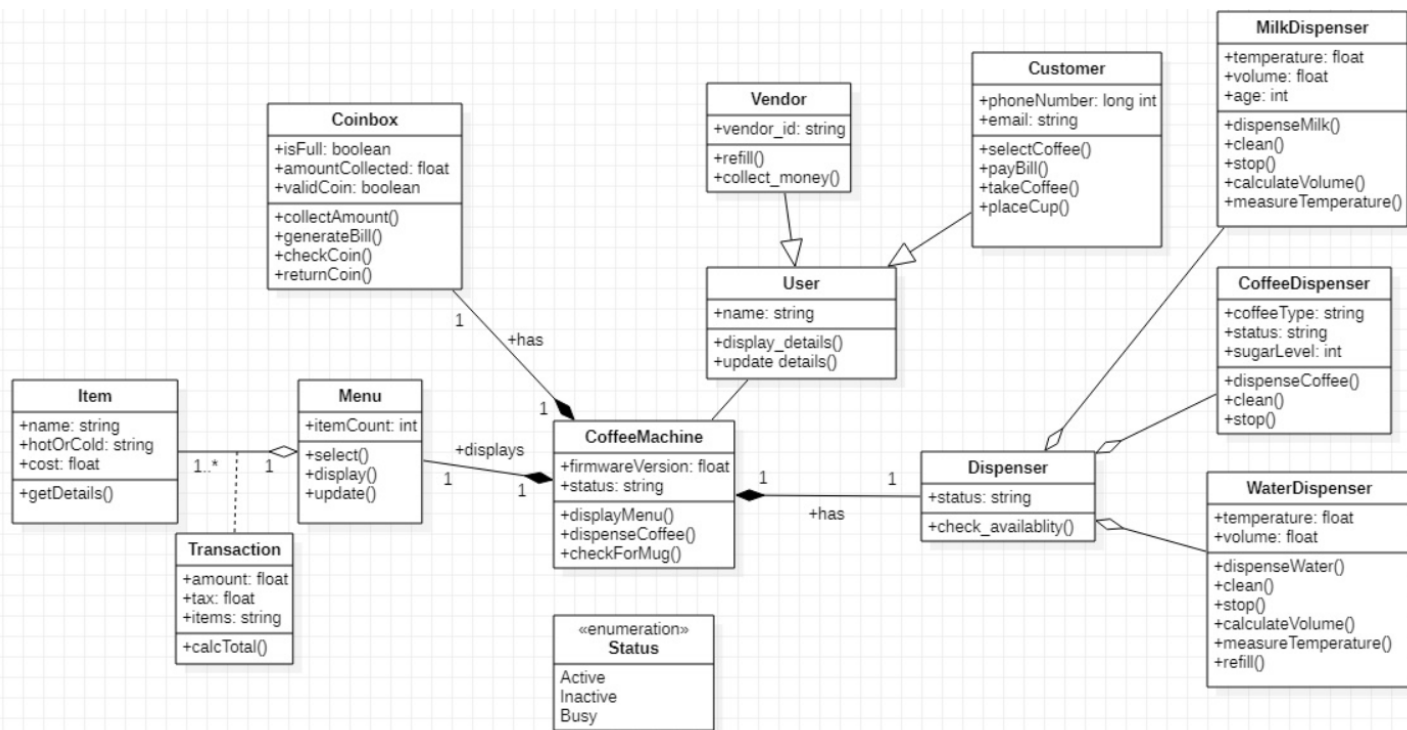Pouring   do/pour selected flavour   [not empty]
finished

Fig 4.3

# 5. Online Shopping System

**Problem statement:**

Design UML diagrams for Online Shopping System with system requirements

specification. **Software Requirements Specification (SRS):**

The online shopping system allows the users and vendors to exchange products remotely and reduces the amount of cost and time substantially. The software provides the following facilities to the customers:

• Facilitates easy shopping online anywhere with free shipping (conditions

apply). • Provides information about the products in categories

• Can avail the facility of purchasing second hand products

• Can reserve if the particular product is not available

• Customers are provided with up to date information on the products available

• Provides email facility for future correspondence

• Provides backup facility

• Can add nearly ten products to their shopping cart at a time

The software will not provide the following facilities to the customers:

• Cannot reserve the product for more than two days.

• Cannot reserve more than two products

• Responsibility of damages

• The product cannot be changeable once confirmed

The software provides the following facilities to the merchants:

• Facilitates easy bidding facility

• Provides complete information about the customers
• Provides complete information about their products
• Can avail the facility of email correspondence and avail the brand catalog facility

**Class Diagram:**

The below shown class diagram contains the following classes: WebUser, Customer, ShoppingCart, Account, Product, Order, Payment, Netbanking, COD with multiplicities as shown.

Association: WebUser owns ShoppingCart, ShoppingCart has Product, Order consists of Product, Payment completes Order.

Generalization: Netbanking and COD are generalized to Payment class.

Association class: LineItem with ShoppingCart and Product

Composition: Account is composed of ShoppingCart, Customer, and Order.

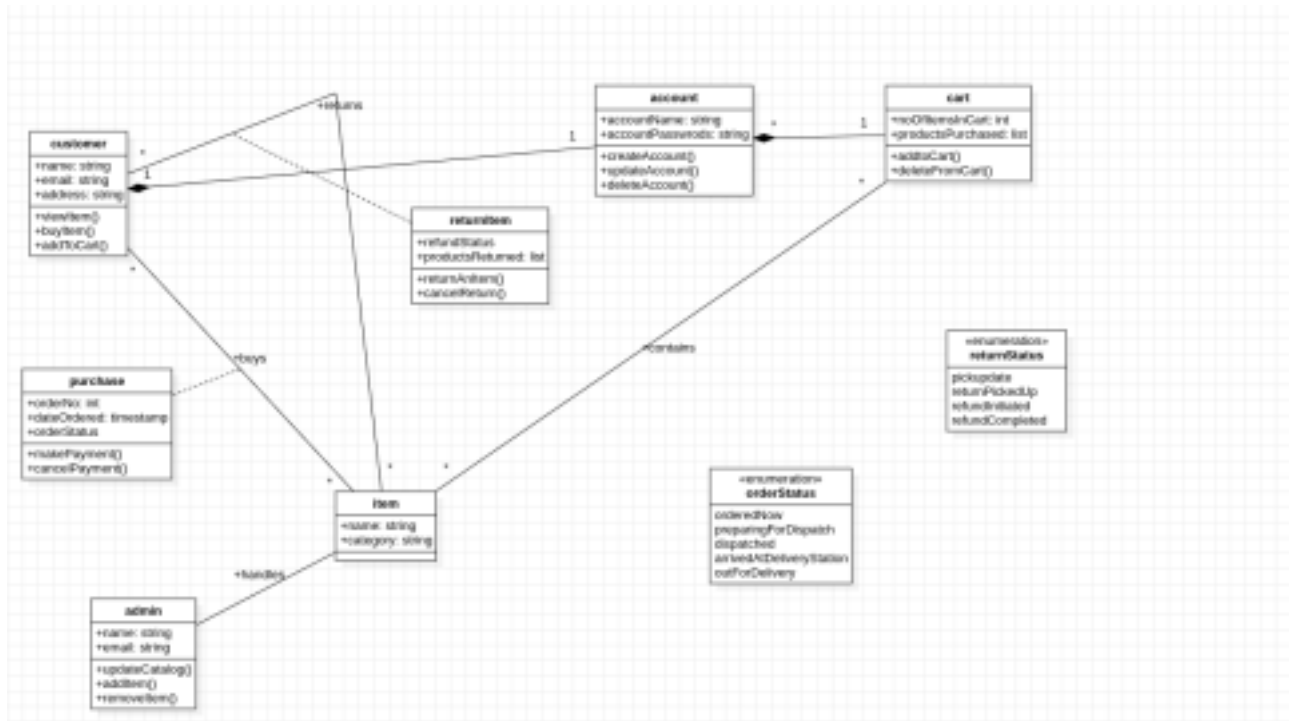Enumerations: UserState, OrderStatus



Fig 5.1

## 1. Advance state diagram

The advanced state diagram depicted below contains one nested state and one submachine, which on successful login shows the AddToCart procedure and PaymentSystem procedure. It contains initial state and termination state with AddToCart as a nested state including the required simple states. It also has a submachine state named PaymentSystem with initial, termination state along with simple states; Method, Card, Validation, Processing.

## OnlineShoppingSystem

● —click login→ **LoggingIn**

**LoggingIn** —[valid username and password]→ **HomePage**

**HomePage**
do/display catalogue

**HomePage** —click logout→ **LoggingOut**

**HomePage** ←select→ **Catalogue**

**Catalogue**
do/display items

**Catalogue** —select→ **Item**

**Item**
do/display product details

**LoggingOut** → ◉

### AddToCart

**payment: PaymentSystem**
O→O

**Order**

**payment: PaymentSystem** —pay→ **Order**

**payment: PaymentSystem** ←success→ **HomePage** (order)

**Quantity**
do/change quantity

**Order** —order→ **Quantity**

**Cart**
do/display items in cart

**payment: PaymentSystem** —fail→ **Cart**

**Cart** —order→ **payment: PaymentSystem**

**Item** —add item→ **Cart**

**Quantity** ←remove item— **Cart**

**Cart** —change quantity→ **Quantity**

## stm PaymentSystem

● —amount→ **Method**

**Method**
do/choose payment method

**Method** —select→ **Card**

**Card**
do/enter details

**Card** → **Validation**

**Validation**
do/validate details

**Validation** —fail→ ◉

**Validation** —success→ **Processing**

**Processing**
do/process amount from user account to system

**Processing** —success / fail→ ◉

## 6. Railway reservation system

**Problem statement:**

Design UML diagrams for Railway Reservation System with system requirements

specification. **Software Requirements Specification (SRS):**

To develop a user-friendly Railway Reservation System to enable passengers to book tickets online and make payment online as well. Railway reservation system project which provides the train timing details, reservation, billing and cancellation on various types of reservation namely,

• Confirm Reservation for Seat.

• Reservation against Cancellation.

• Waiting list Reservation.

• Online Reservation.

• Tatkal Reservation

This system enables the Advance booking in any class, against general and ladies quota, on payment of fare in full for adults and children, a maximum of six berths/seats at a time, for journey between any two stations served by a train. It also provides details about

1. Timetable

2. Train Fares

3. Current status of reservation position

4. Train available between a pair of stations

5. Accommodation available for a train/date combination Types of tickets: General and

Tatkaal **Class Diagram:**

The below shown class diagram contains the following classes: RailwayStation, Train, Admin, Passenger, TicketBooking, Payment, Ticket, General and Tatkal with multiplicities as shown. Association: Passengers takes Train, Admin supervises TicketBokking, TicketBooking

pays Payment.
Generalization: General and Tatkal are generalized to Tickett class.
Association class: Bank with TicketBooking and Payment

Composition: RailwayStation contains train, Passenger books TicketBooking, TicketBooking has Ticket.
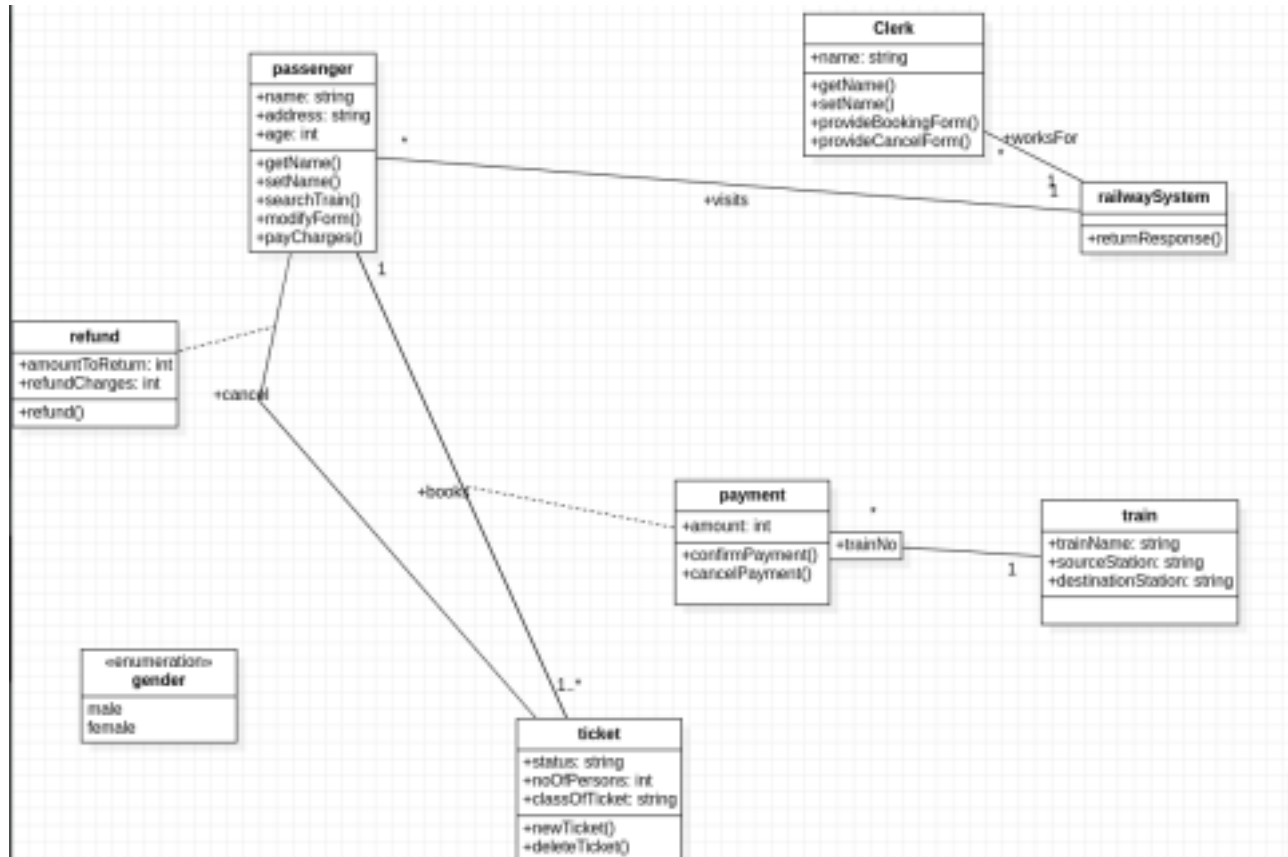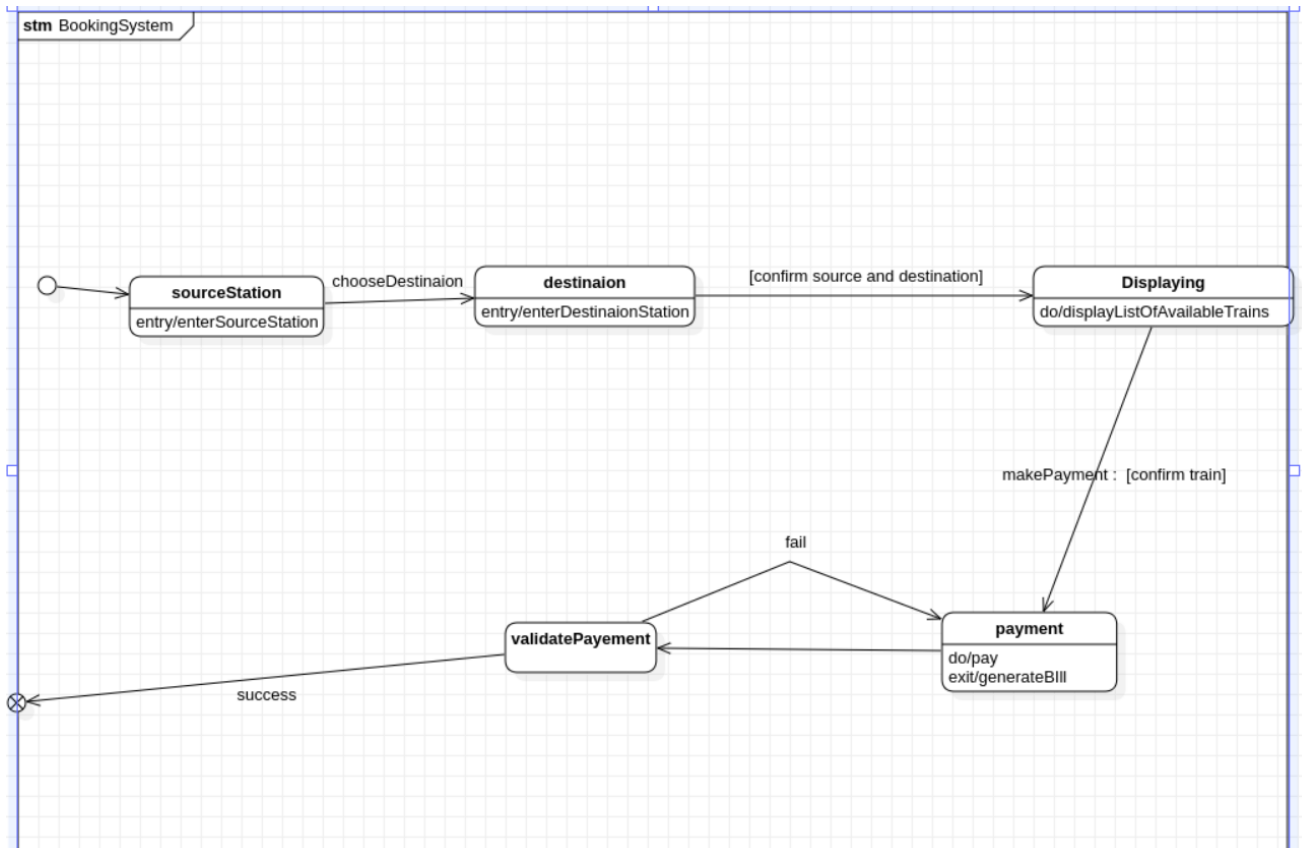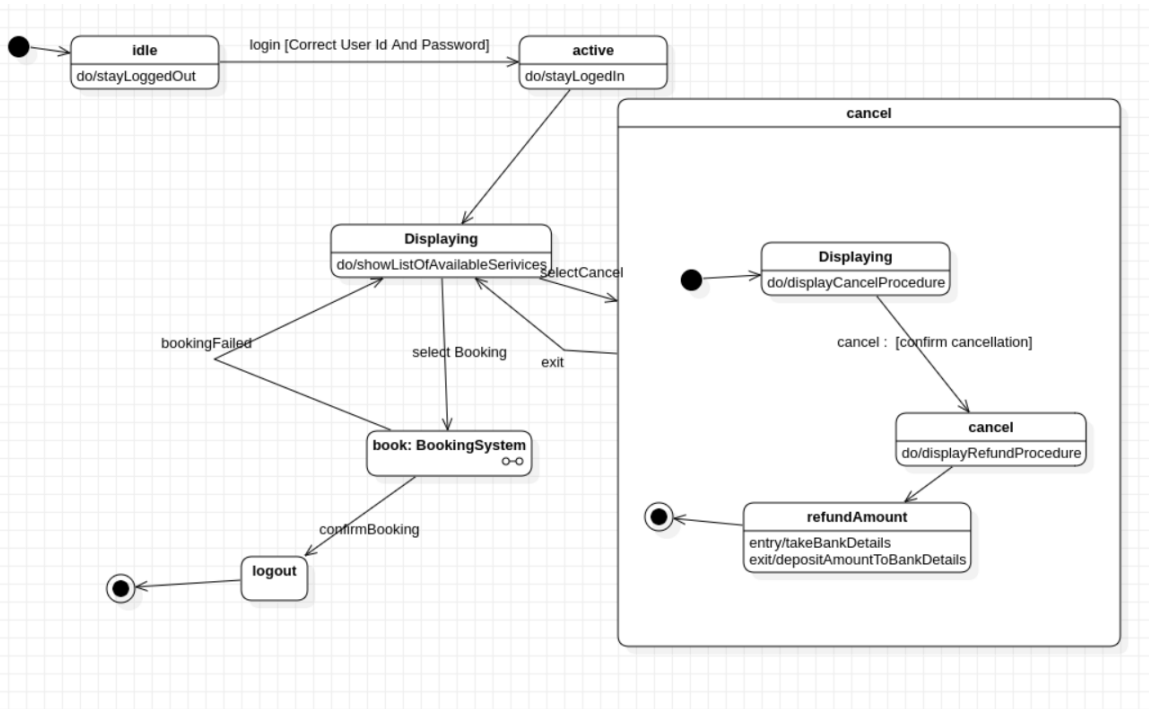


Fig 6.1

# 1. Advance state diagram

The advanced state diagram depicted below contains one nested state and one submachine, which on successful login shows the ChooseTrain details and PaymentSystem procedure. It contains initial state and termination state with ChooseTrain as a nested state including the required simple states. It also has a submachine state named PaymentSystem with initial, termination state along with simple states; Method, Card, Validation, Processing.

**idle**
do/stayLoggedOut

login [Correct User Id And Password]

**active**
do/stayLogedIn

**cancel**

**Displaying**
do/showListOfAvailableSerivices

selectCancel

**Displaying**
do/displayCancelProcedure

cancel : [confirm cancellation]

**cancel**
do/displayRefundProcedure

bookingFailed

select Booking

exit

**book: BookingSystem**

**refundAmount**
entry/takeBankDetails
exit/depositAmountToBankDetails

confirmBooking

**logout**

---

**stm** BookingSystem

**sourceStation**
entry/enterSourceStation

chooseDestinaion

**destinaion**
entry/enterDestinaionStation

[confirm source and destination]

**Displaying**
do/displayListOfAvailableTrains

makePayment : [confirm train]

fail

**validatePayement**

**payment**
do/pay
exit/generateBIll

success

# 7. Graphics Editor

**Problem statement:**

Design UML diagrams for Graphics Editor with system requirements

specification. **Software Requirements Specification (SRS):**

The graphics editor provides an Application Programmer's Interface that enables a programmer to develop their own graphical model editor for a specific type of model. This API in turn, relies on extending the Eclipse Graphical Editing Framework to provide an environment in which the editor functions, and the programmer can create a graphical editor and palette of shapes in order to modify an underlying model. The graphical editor provides an interface with which the programmer implements said editor for a given underlying model. Such an instance of the graphical editor allows a user to drag objects from a specified model into a working graphical diagram.

It should support following functionalities:

• It contains the toolbox which contains tools like: Line, Circle, Rectangle, Arc,Text, Draw, Eraser

• Color box or palette

• Standard toolbar with options for New, Open, Save, toolbox and Text

Toolbox. • One integrated view to users for toolbar, color box, menu, and

graphic screen. • Easy handling of tools for users.

• Ability to group several drawings into one i.e., complex drawing.

• Provision of zoom in and zoom out.

• Different shadings of line tool are provided
**Class Diagram:**

The below-shown class diagram contains the following classes: GraphicEditor, Document, Sheet, Group, Object, Text, ZeroDimension, OneDimension, TwoDimension, Point, Line, Arc, Circle, Rectangle, Ellipse with multiplicities as shown.
Generalization: Text, ZeroDimension, one dimension, and TwoDimension are generalized to Object class. Point is generalized to ZeroDimension, Line and Arc are generalized to one dimension, Circle, Rectangle and Ellipse are generalized to TwoDimension. Composition:

GraphicsEditor made of Document, Document contains Sheet, Sheet has Group, Group has Objected.
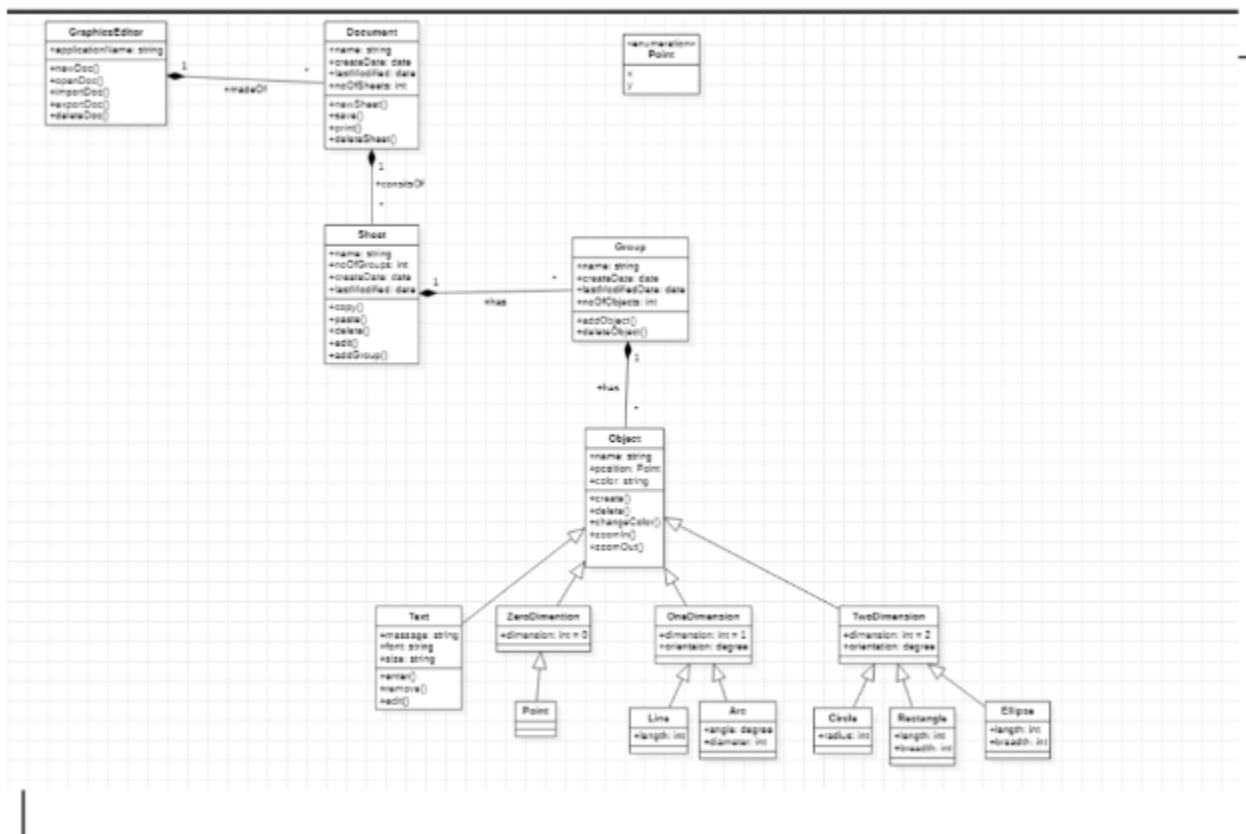
Enumeration: Point



Fig 7.1

# 1. Advance state diagram

The advanced state diagram depicted below contains one nested state and one submachine, which on successful login shows the Saving procedure and DrawingSystem procedure. It contains initial state and termination state with Saving as a nested state including the required simple states. It also has a submachine state named DrawingSystem with initial, termination state along with simple states; Shapes, Display and format each shape.
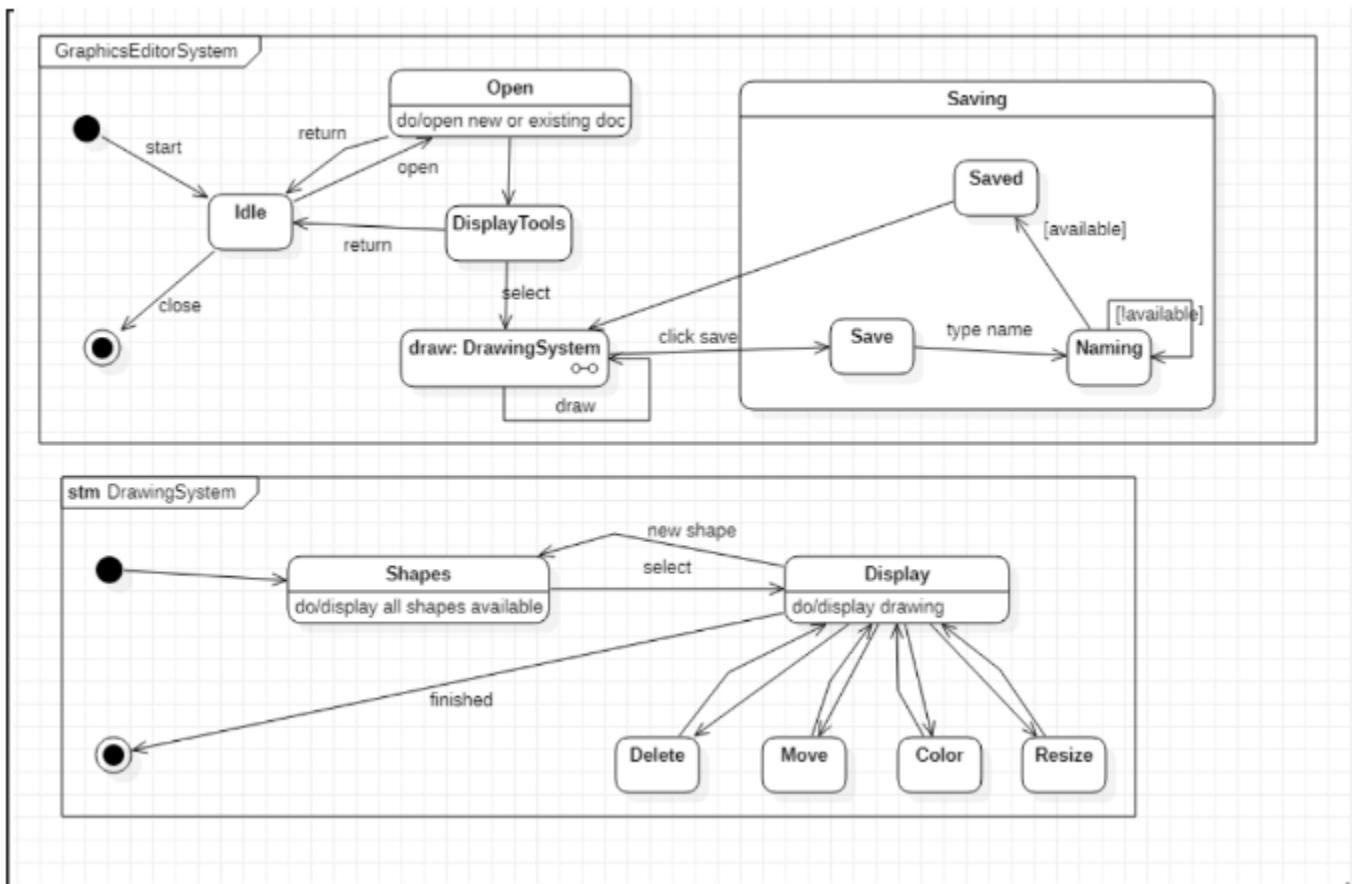
Fig 7.3