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:

SANNIDHI KASTURI

1BM19CS143

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 SANNIDHI KASTURI (1BM19CS143) during the 6th Semester Mar-July-2022.

Signature of the Faculty Incharge:

NAME OF THE FACULTY: Dr. Latha N.R.

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 Department, Placement and COE, Student gets Facility, Coursesare chosen by student.
Generalization: Hostel, library are generalized to Facility class.
Aggregation: Staff consists (aggregate of) Teaching and Non-Teaching.
Composition: Department has (or is composed of) Course and Saff.

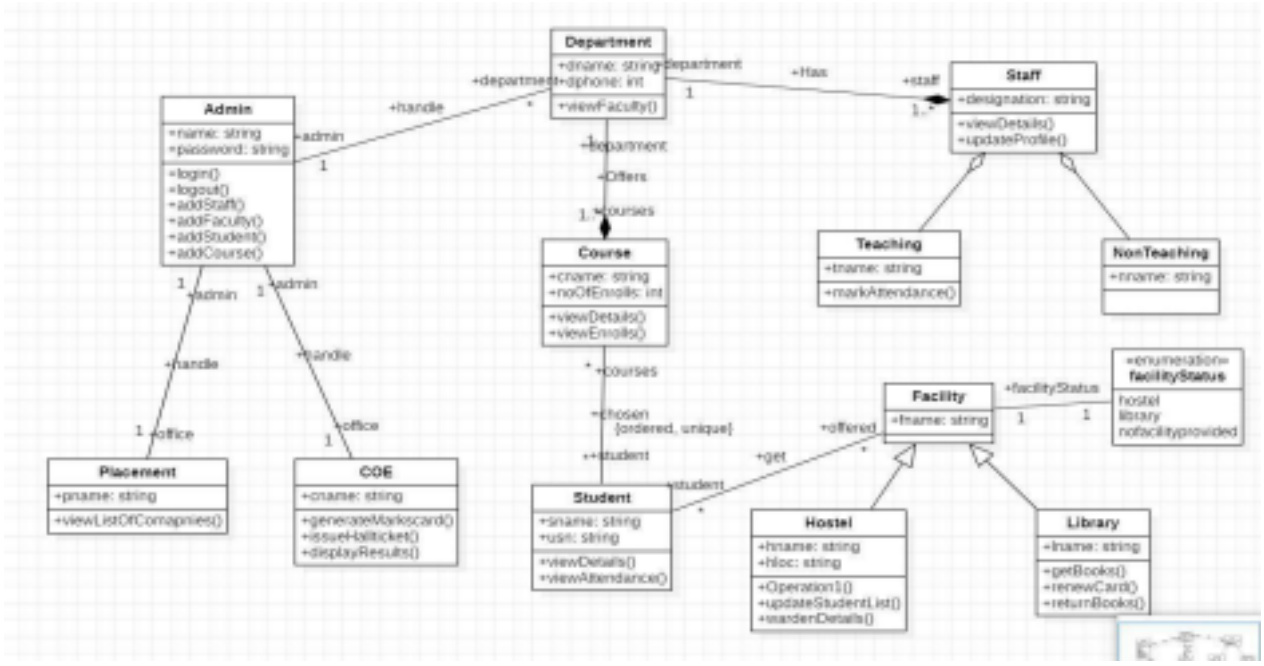

Fig 1.1

State Diagram:

1. Simple state diagram

The simple state diagram has a initial state (Authorization) and final state (Logout) with many other simple states which are: Login, Register (if not a member),then check with the user level, the fork used to split the states whether it's a student or admin or faculty, then join is used after

the operations performed by each state, then other states; verification (edit state if required) and changes saved.

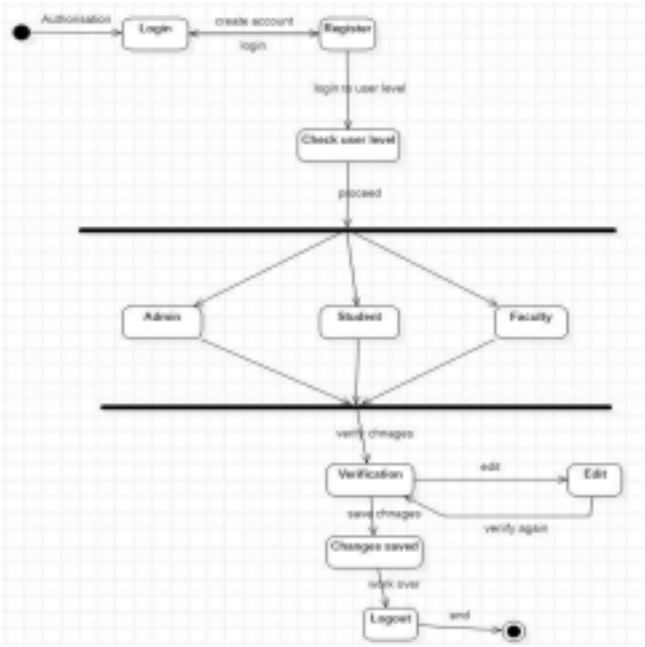

Fig 1.2

2. 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.

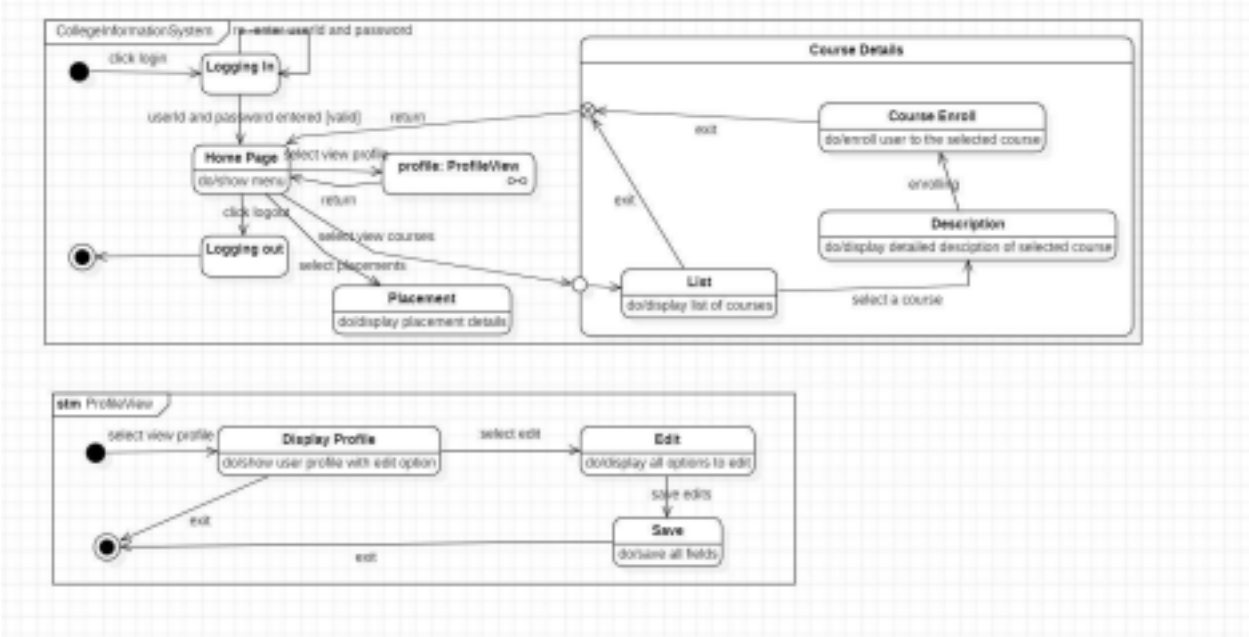

Fig 1.3

Use Case Diagram:

1. Simple use case diagram

The use case diagram for hostel management system has the following:
Actors: Administrator , Student, Event Manager, Faculty, Librarian.
Use cases: Manage Depts, Manage courses, Manage events, Manage books, Create courses, View events, View courses, View events, View students, add student, remove student, add books, remove books, add events, remove events.

Fig 1.4

2. Advance use case diagram

The advanced use case diagram has extra functionalities which includes extends, includes and generalization. The show available books use case extends view books use case, view events use case includes add events and remove events, issue books use case includes verify student and check availability of book.

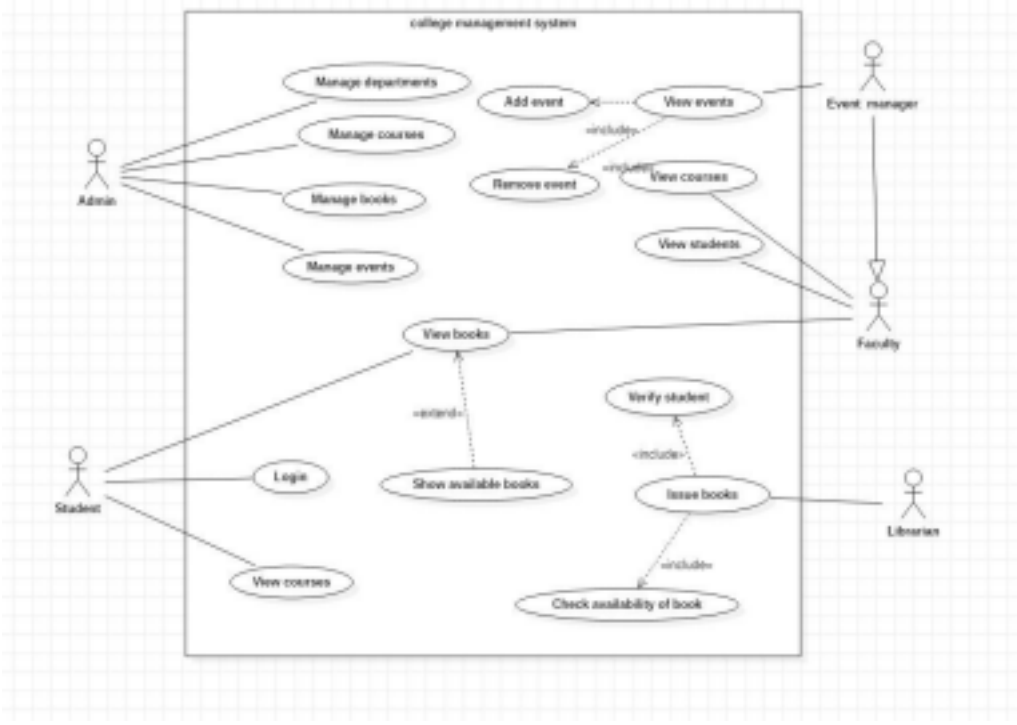

Fig1.5

Sequence Diagram:

1. Simple sequence diagram

The actors are: Student, management system, database, COE database.
This scenario of management system and database resolving issues contains messages between objects as well as activities performed by these objects.

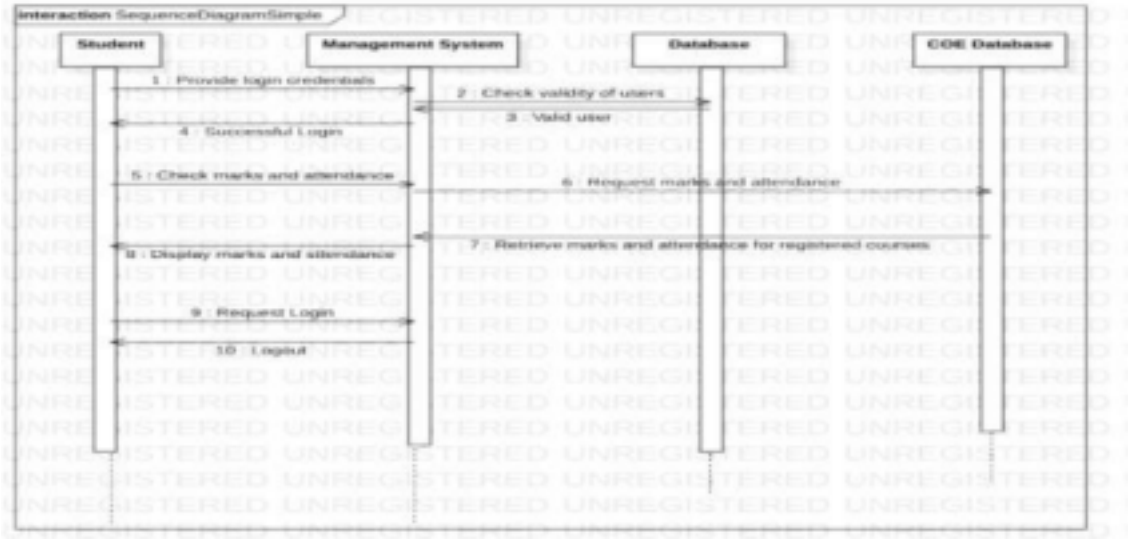

Fig 1.6

2. Advance sequence diagram

The lifeline is the dotted line and the rectangles represent the period of time the object is executing and is hence called activation.
The recursive function of verify is shown by double activation rectangle of verify payment and successful message.

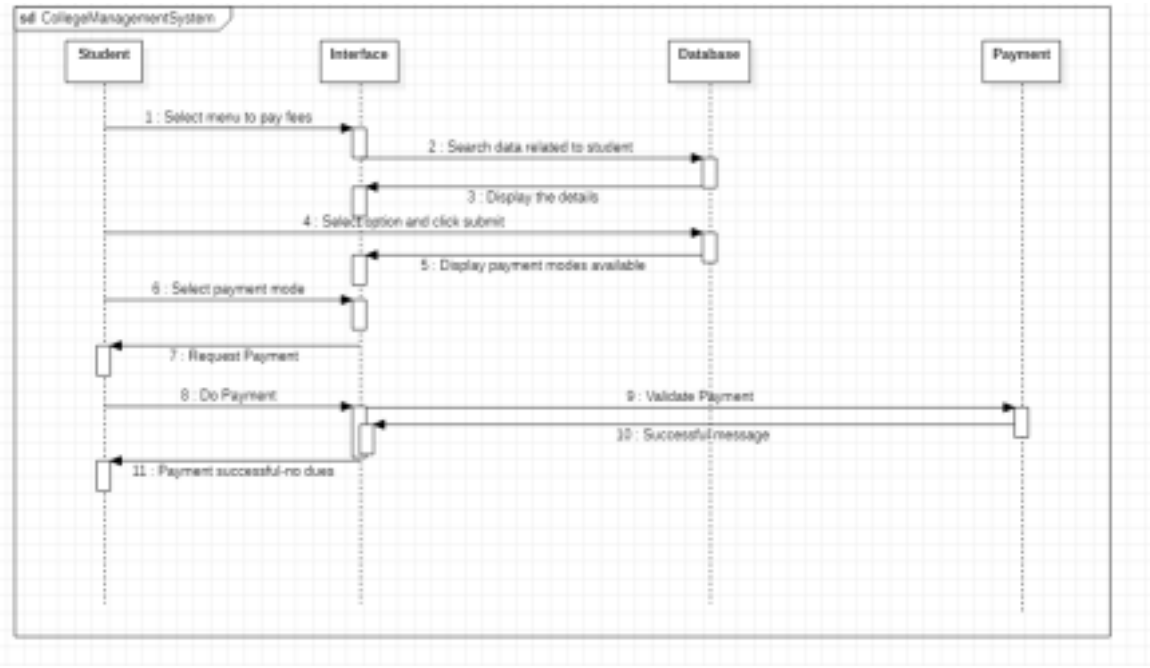

Fig 1.7

Activity Diagram:

1. Simple activity diagram

In the activity diagram, the control flows from the initiation to the login activity. Then the login credentials are validated in the validate activity. A condition is added to check whether the validation is successful or not; if the validation is successful, the control flows to the activity where Course details and options are displayed whereas on failure control returns to the login activity. A condition is then added to check the user's selection which can be to register the courses successfully and then the control flows to the corresponding activities. Then the control flows to the logout activity and then termination.

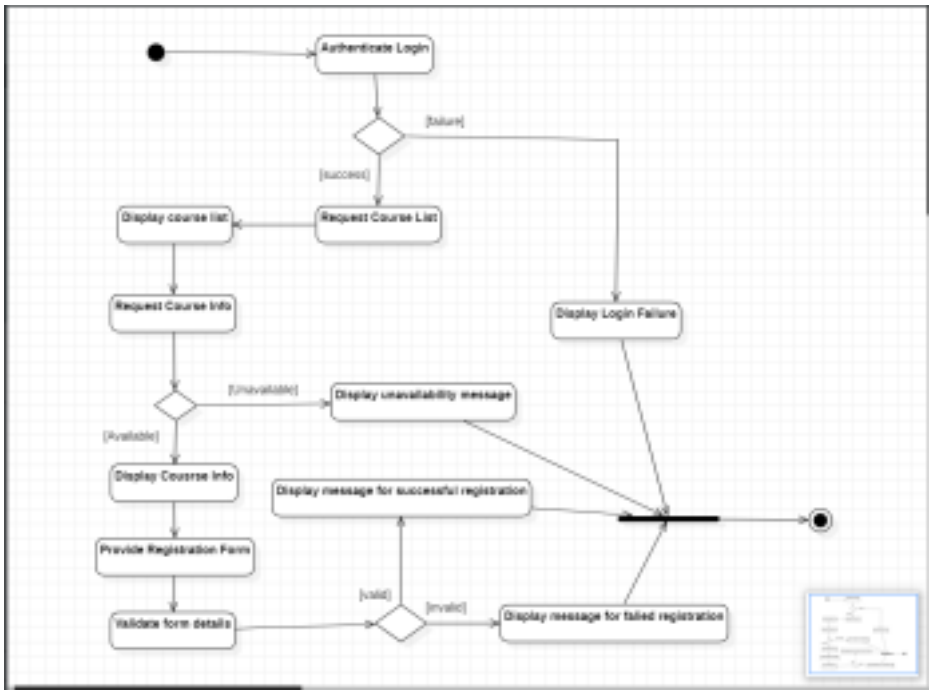

Fig 1.8

2. Advance activity diagram:

The advanced activity diagram starts from initiation and then user login activity where a signal is sent to the network for request validation and upon confirmation the control flows to profile and then fee payment activity. There are three swimlanes for Payment gateway, Database manager and accountant where validate payment, update database and generate receipt respectively. Then the control flows to the home page and then termination activities.

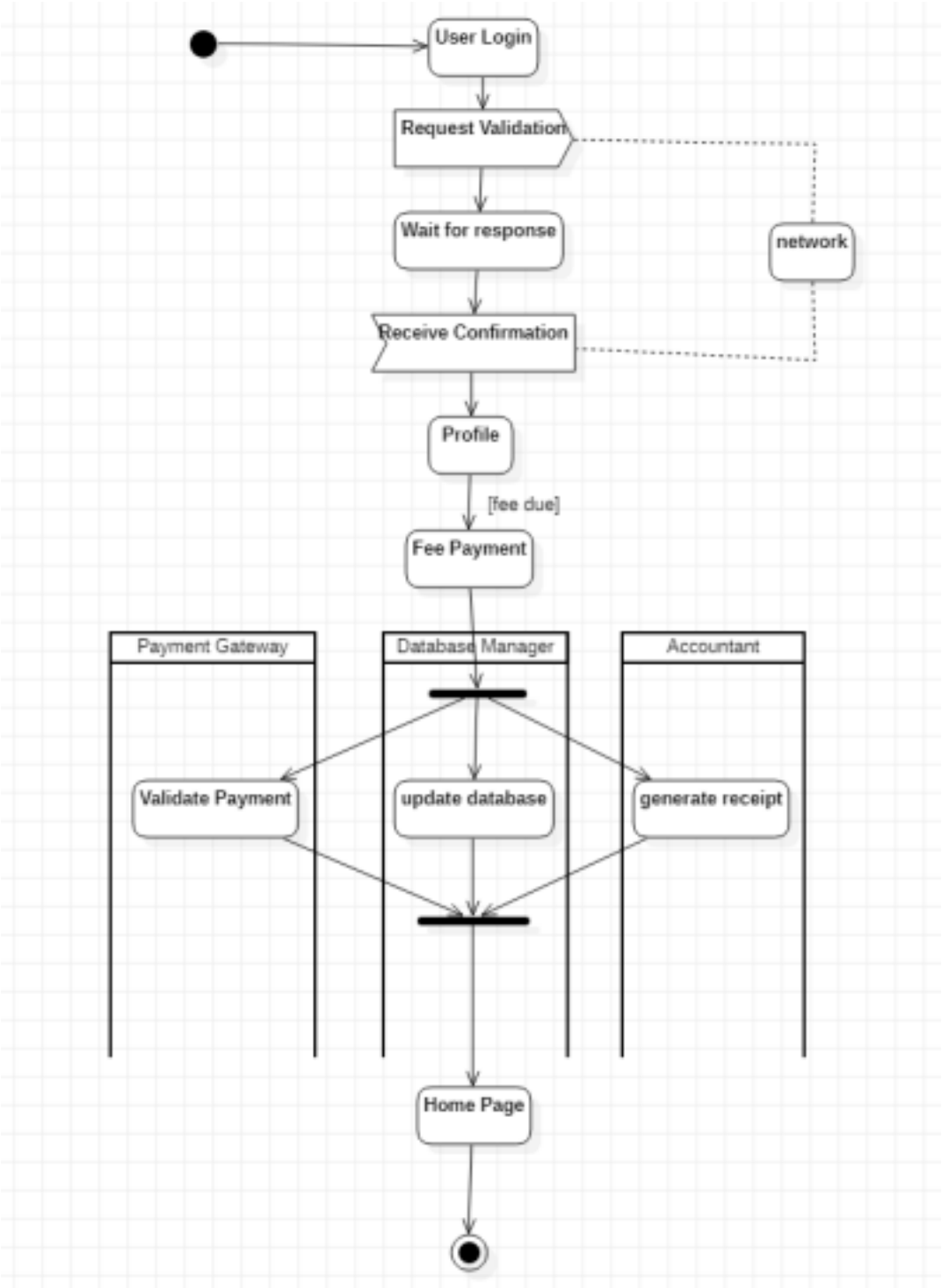

Fig 1.9

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: Warden manages Student, Student stays_in Hostel, Student is allocated to Room_allocation, Administrator decides_room Room_allocation, Administrator generates Receipt_generation, .
Generalization: Student, Administrator and Warden are generalized to Person class.
Aggregation: Rooms class (composed of) with Bed class.
Composition: Hostel has (or is composed of) Rooms.
Enumeration: RoomType.

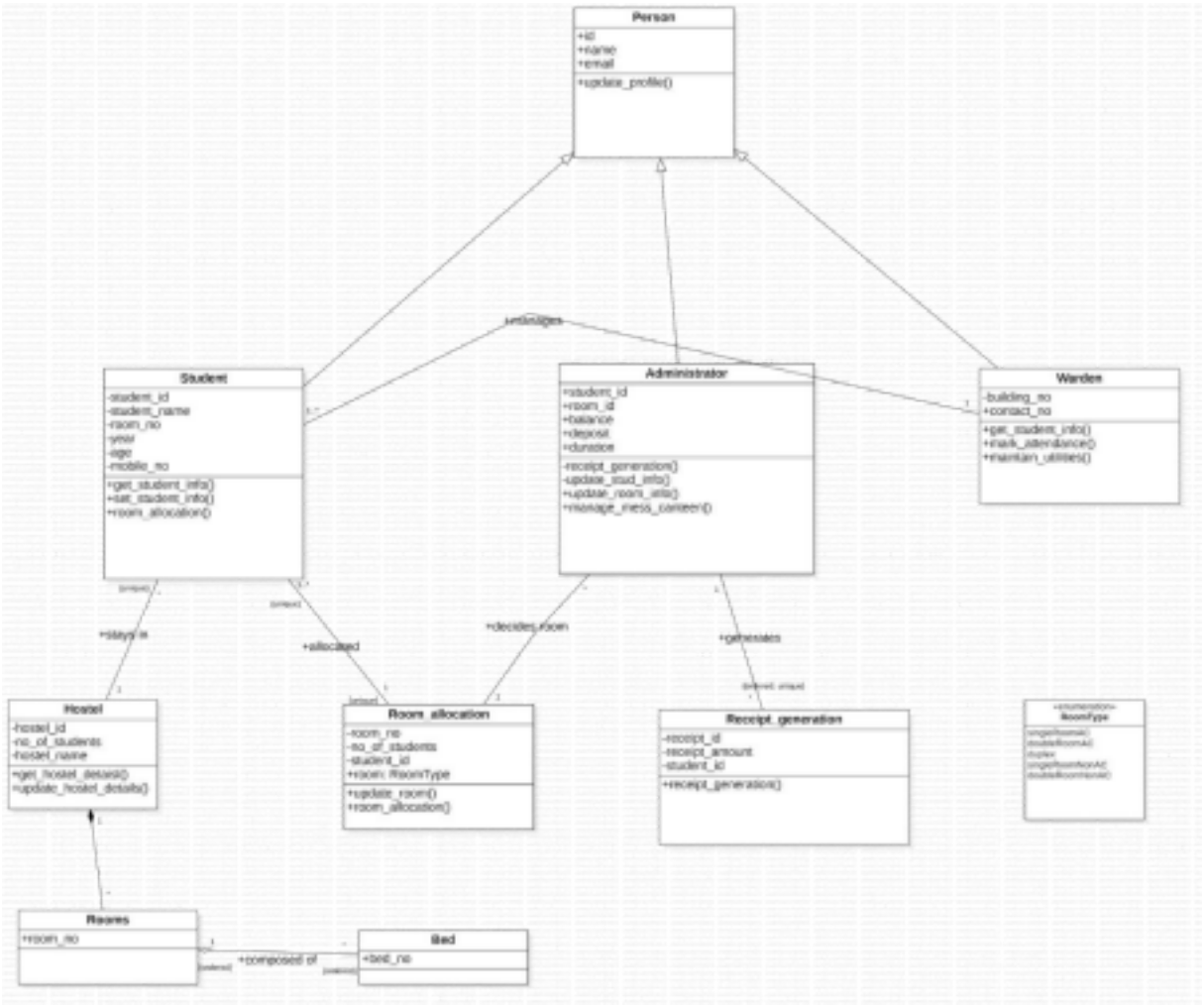

Fig 2.1

State Diagram:

1. Simple state diagram

The simple state diagram has a initial state (Authorization) and final state (Logout) with many other simple states which are: Login, Create account (if not a member),then check with the user level, the fork used to split the states whether it's a hostel student or hostel management, then join is used after the operations performed by each state, then other states; available rooms, room freezing, payment, payment confirmed, room allotted and changes saved.

Fig 2.2

2. 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.

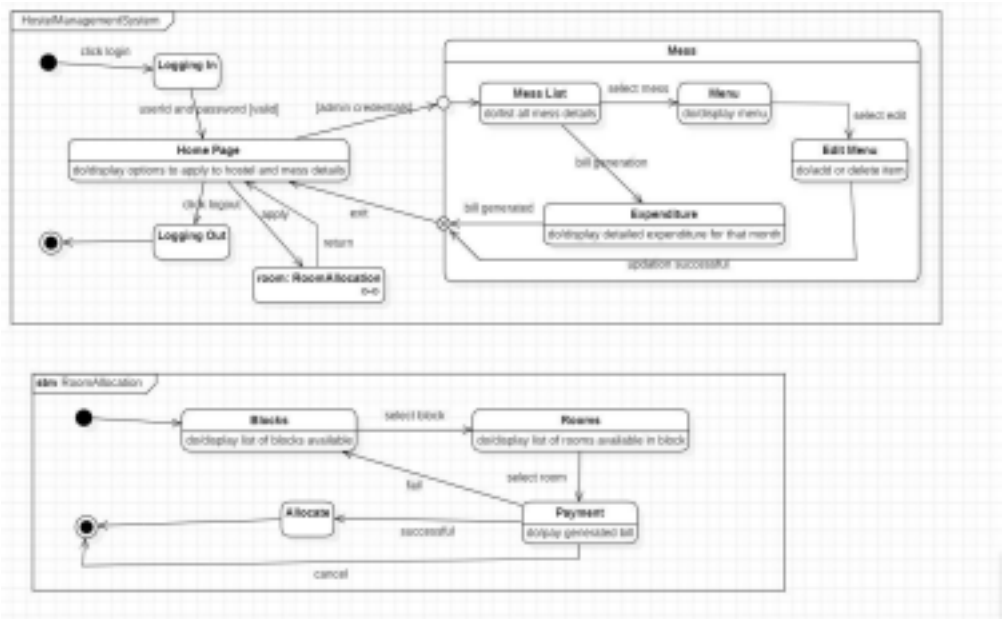

Fig 2.3

Use Case Diagram:

1. Simple use case diagram

The use case diagram for hostel management system has the following: Actors: Admin, Student, Warden, Mess staff
Use cases: collect fees, maintain mess, maintain attendance, approve room allotment, act on complaint, edit hostel info, add room, delete room, change mess, book room, change hostel, make complaint, fee payment, change room, change room, give feedback, create menu, check hostel facilities, search rooms, give permissions.

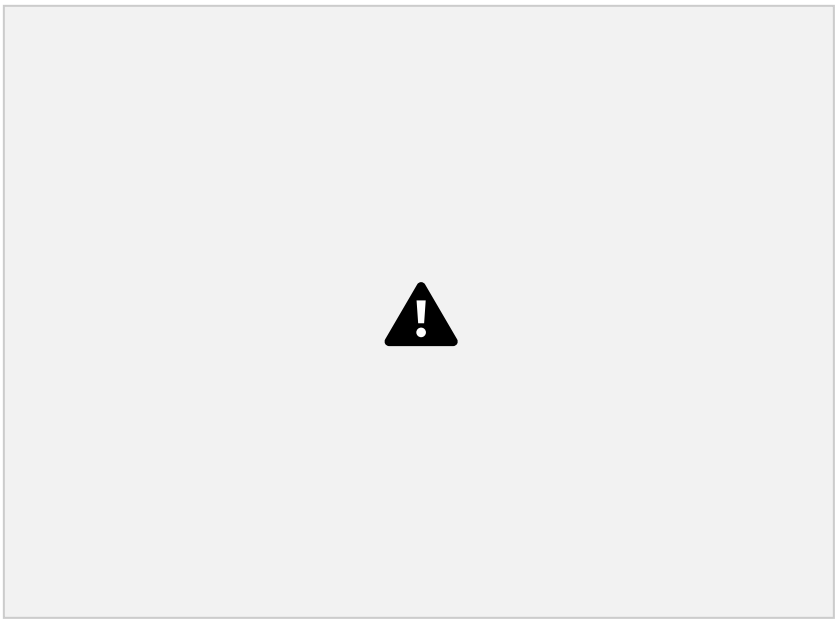

Fig 2.4

2. Advance use case diagram

The advanced use case diagram has extra functionalities which includes extends, includes and generalization. The edit hostel info use case extends add room use case, collect fee use case includes verify student, add room use case includes delete room use case.

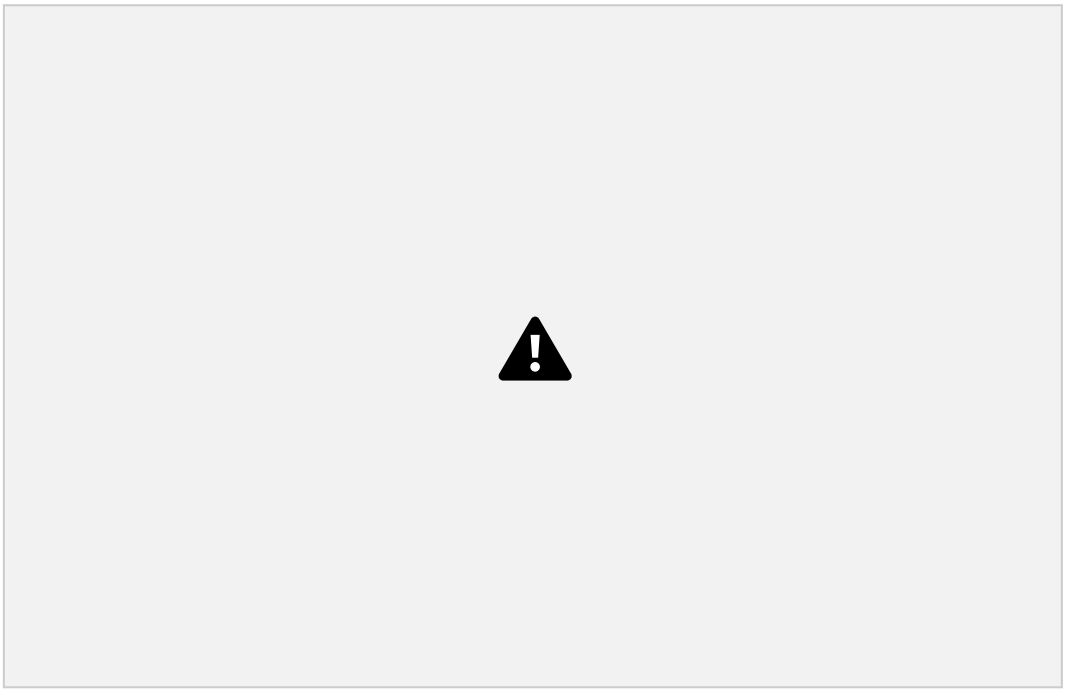

Fig 2.5

Sequence Diagram:

1. Simple sequence diagram

The actors are: Admin, login database, building database, hostel database. This scenario of hostel database management and student booking resolving issues contains messages between objects as well as activities performed by these objects.

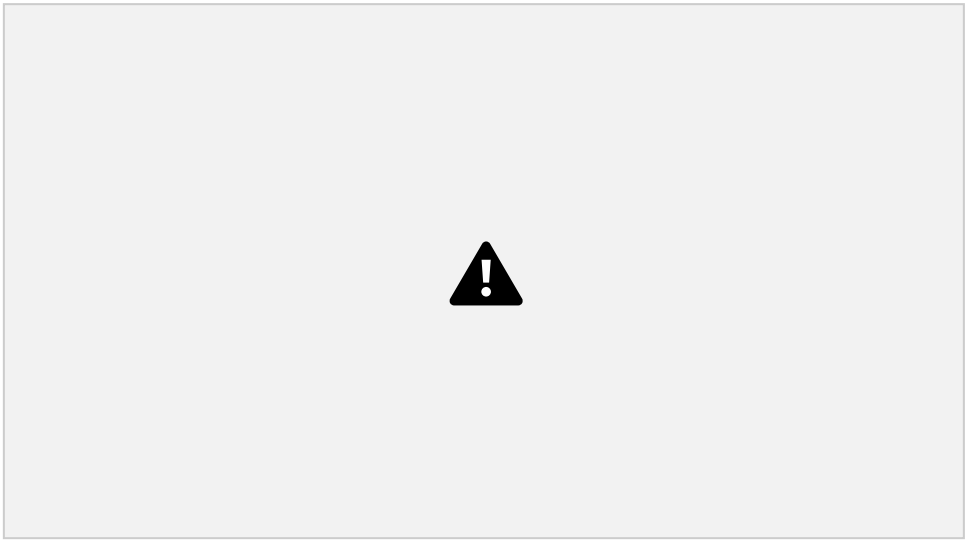

Fig 2.6

2. Advance sequence diagram

The lifeline is the dotted line and the rectangles represent the period of time the object is executing and is hence called activation.
The Login actor has self-message to check with the registration of the student. Async and sync signal replies (dotted line) are used to reply back with specificity to the object.

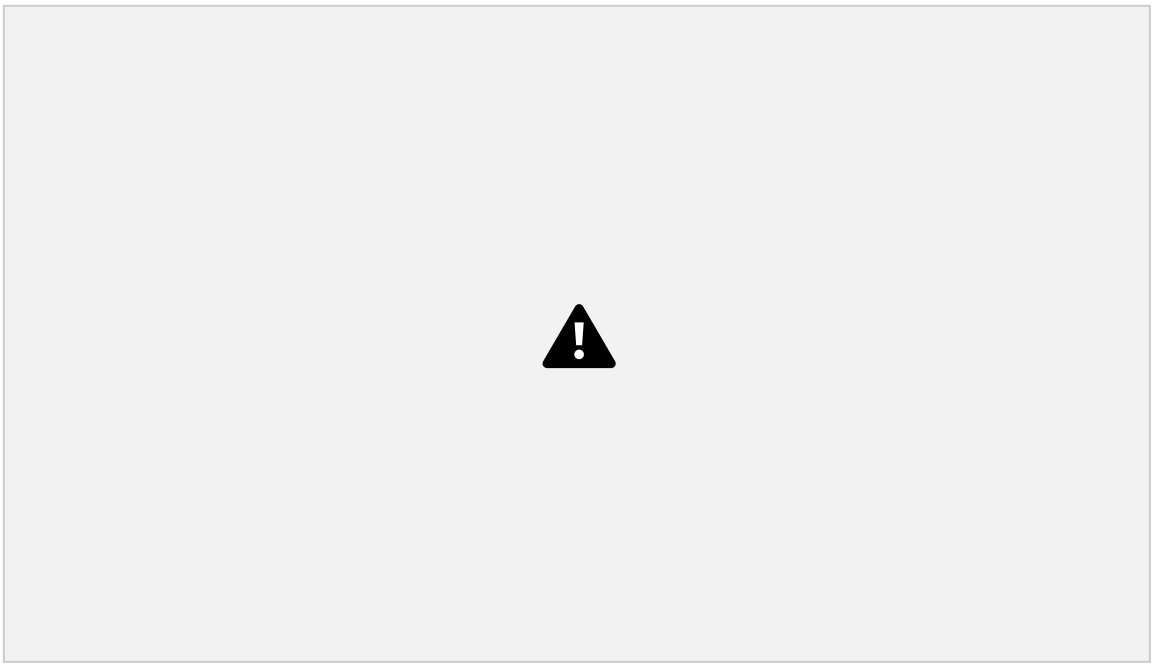

Fig 2.7

Activity Diagram:

1. Simple activity diagram

In the activity diagram, the control flows from the initiation to the login activity. Then the login credentials are validated in the validate activity. A condition is added to check whether the validation is successful or not; if the validation is successful, the control flows to the activity where room availability details and options are displayed whereas on failure control returns to the login activity. A condition is then added to check the user's selection which confirms the payment of the hostel booking and then the control flows to the corresponding activities. Then the control flows to the logout activity and then termination.

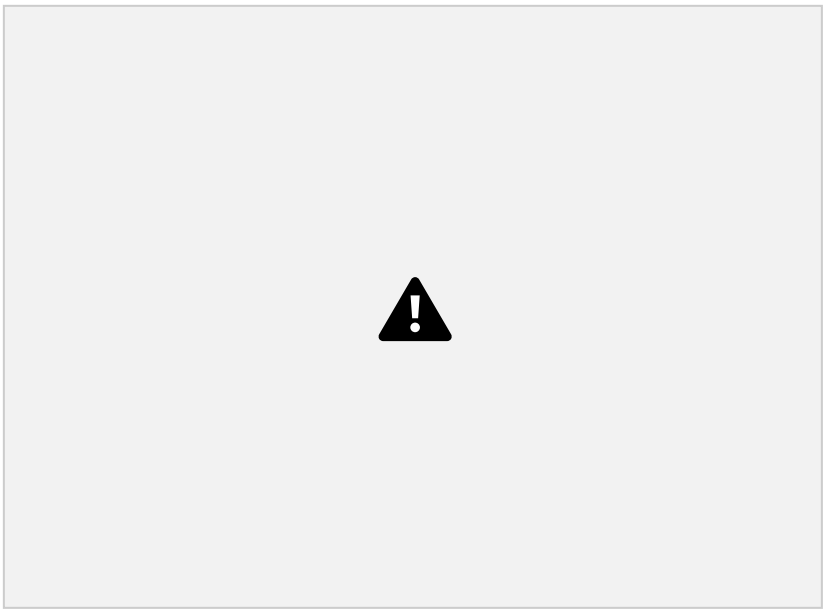

Fig 2.8

2. Advance activity diagram

The advanced activity diagram starts from initiation and then in the student swimlane, student login activity where a signal is sent to the network for request validation and upon confirmation the control flows to profile and then book room activity. There are three swimlanes namely student, database, payment where validate student, update database and confiem payment respectively. Then the control flows to the home page and then termination activities.,

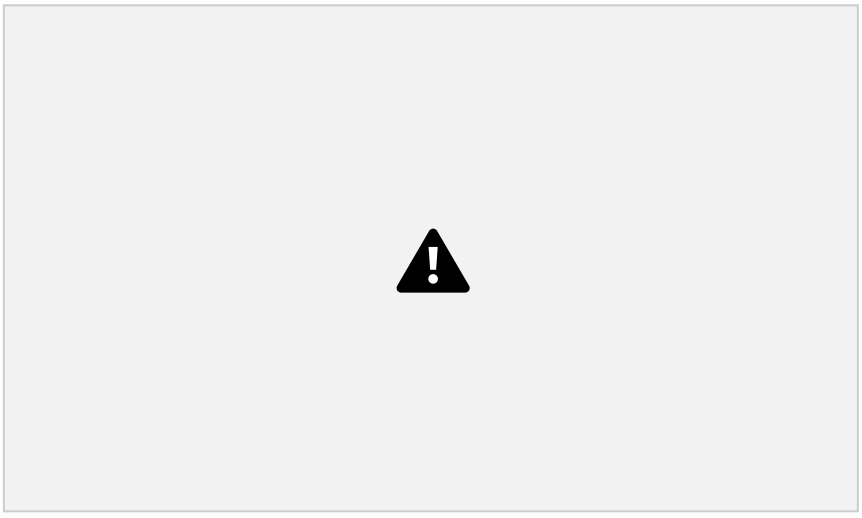

Fig 2.9

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.
Association: Customer buys Product, Customer buys Stocks, Customer pays Payment.
Generalization: User is generalized to Permission class and User is Generalized to Role.
Aggregation: Stock class, Product class, Store class, Customer class are (composed of) with Permission class.
Composition: Payment needs (or is composed of) Permission and Stock (or is composed of) has Product.

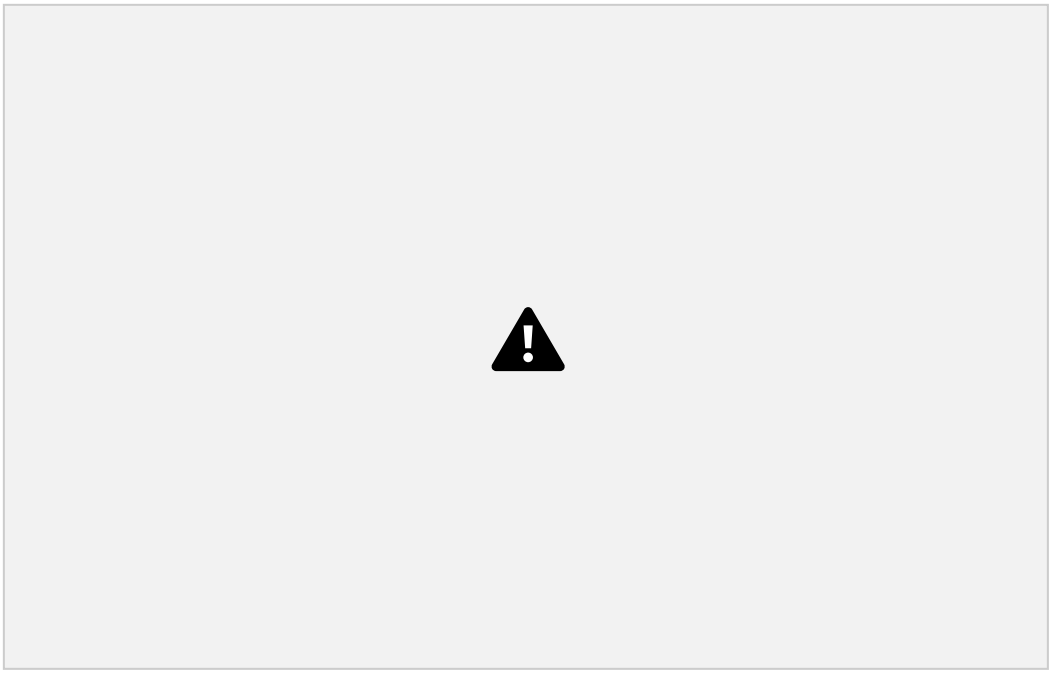

Fig 3.1

State Diagram:

1. Simple state diagram

The simple state diagram has a initial state (Authorization) and final state (Logout) with many other simple states which are: Login, Create Account (if not a member), then identify the user level, the fork used to split the states whether it's a supplier management or owner or customer management, then join is used after the operations performed by each state, then other states; save changes and update.

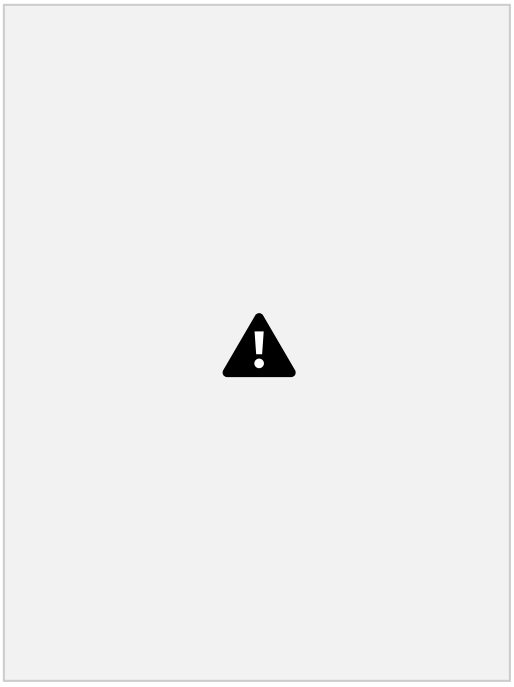

Fig 3.2

2. 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.

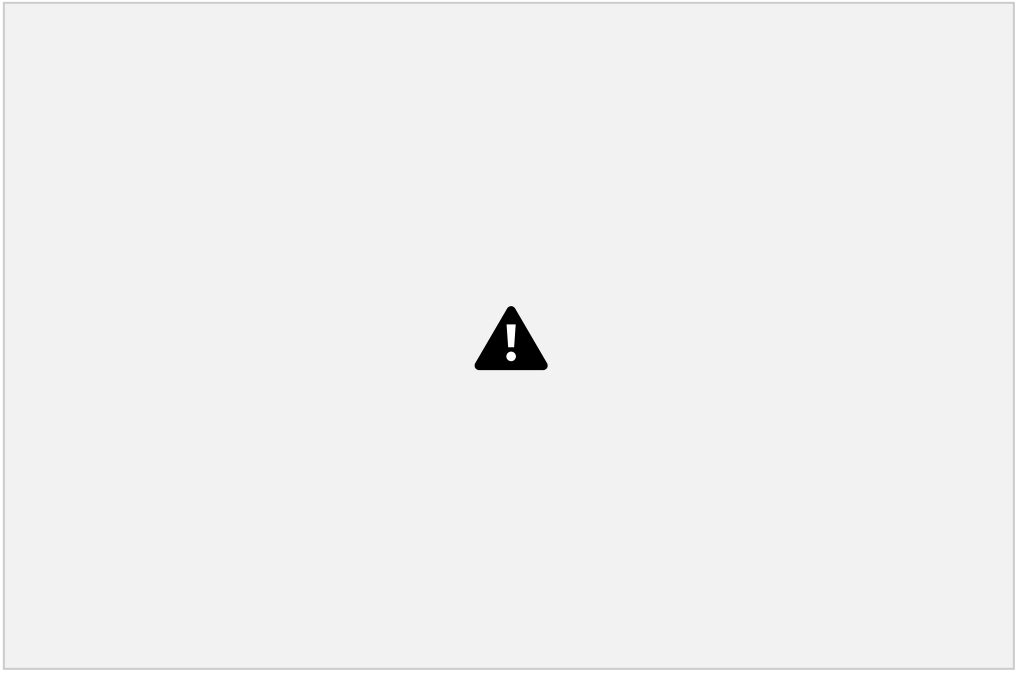

Fig 3.3

Use Case Diagram:

1. Simple use case diagram

The use case diagram for hostel management system has the following:
Actors: Manager, Company owner, supplier, customer.
Use cases: Calculate exp stock, calculate req stock, order product, view returns, find invoice, find factory, modify product, pay taxes, pay bill, track order, report quality issues, search stock, receive payment, check order, search product.

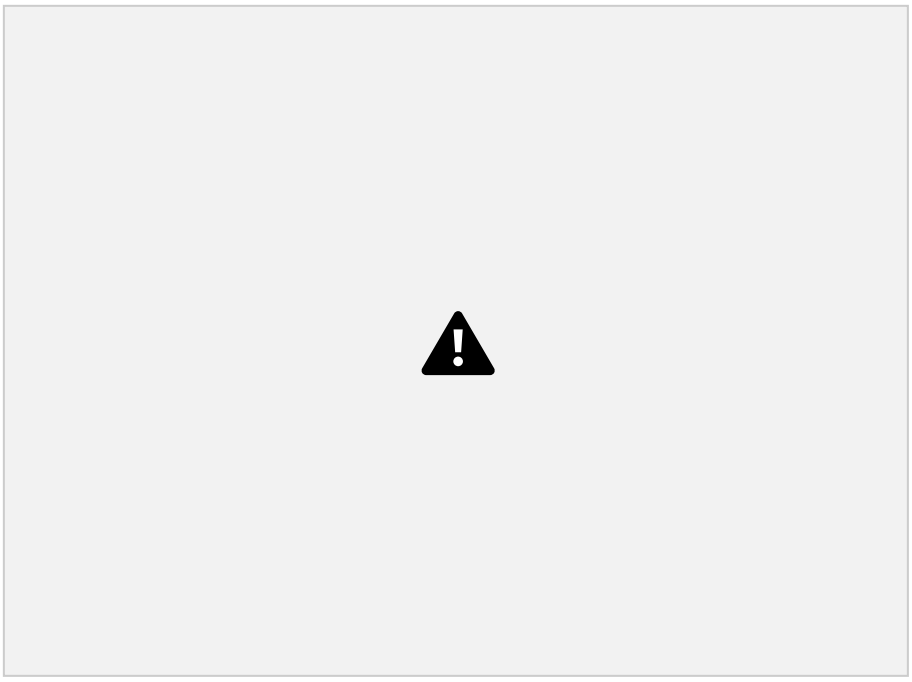

Fig 3.4

2. Advance use case diagram

The advanced use case diagram has extra functionalities which includes extends, includes and generalization. The stock level use case extends place order use case, detective shipment use case extends check quality criteria use case , shipment error use case extends receive shipment with bill use case, pay bill use case includes track order use case.

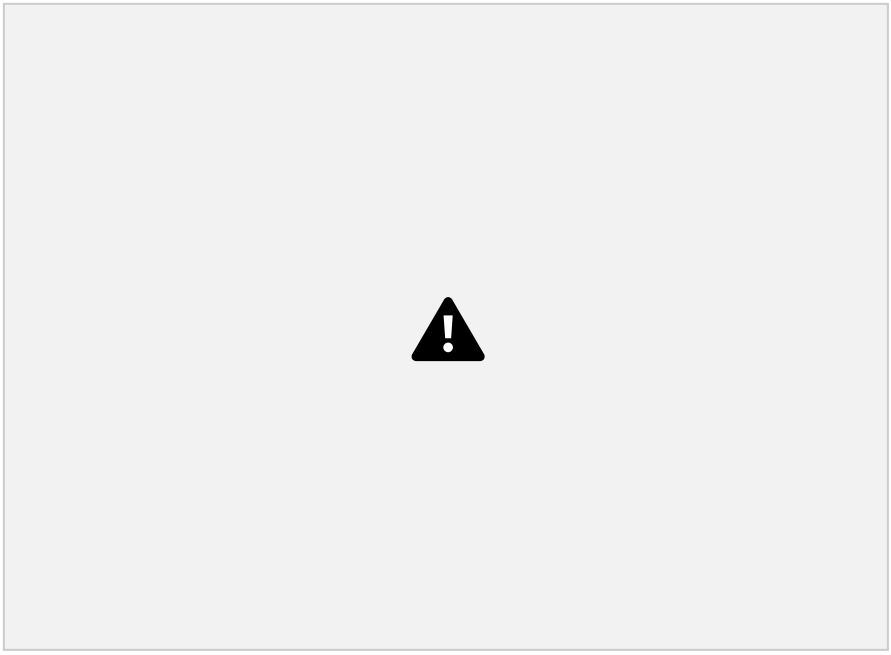

Fig 3.5

Sequence Diagram:

1. Simple sequence diagram

The actors are: Customer, inventory manager, suppliers, accountant, billing. This scenario of inventory manager, suppliers and account resolving issues contains messages between objects as well as activities performed by these objects.

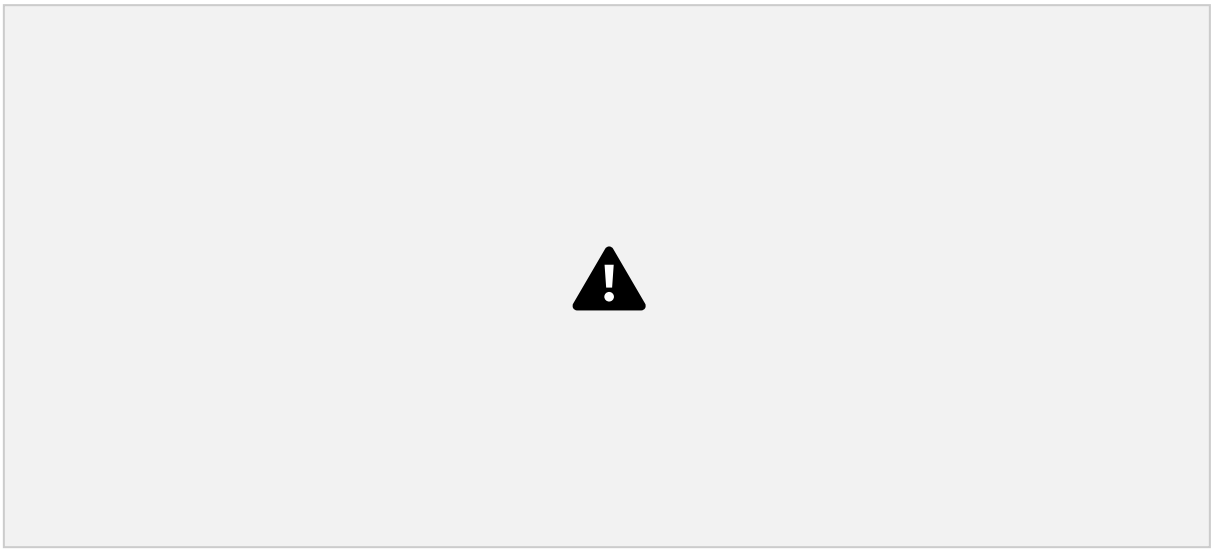

Fig 3.6

2. Advance sequence diagram

The lifeline is the dotted line and the rectangles represent the period of time the object is executing and is hence called activation.
Create message signal is used to indicate the display of failure in any failure situation.

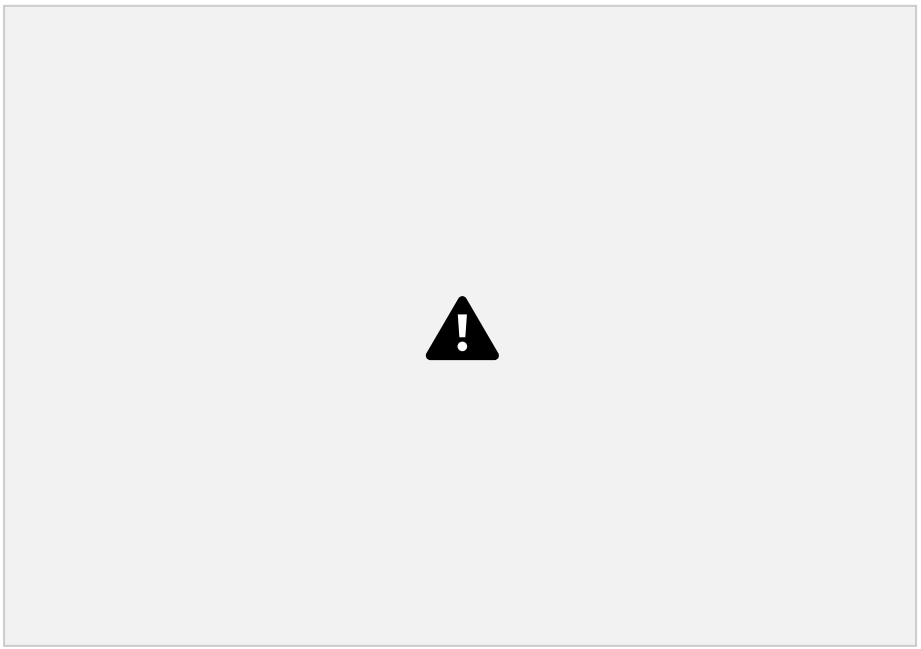

Fig 3.7

Activity Diagram:

1. Simple activity diagram

In the activity diagram, the control flows from the initiation to the login activity. Then the login credentials are validated in the validate activity. A condition is added to check whether the validation is successful or not; if the validation is successful, the control flows to the activity where search product details and options are displayed whereas on failure control returns to the login activity. A condition is then added to check the user's selection which checks with the supplier, delivery and payment process and then the control flows to the corresponding activities. Then the control flows to the logout activity and then termination.

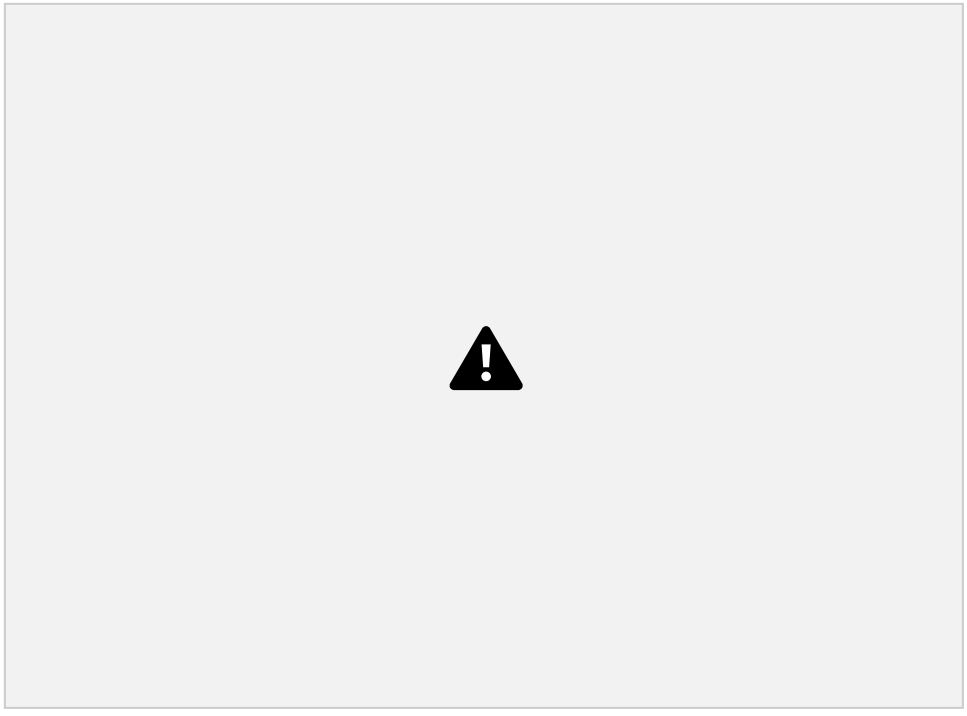

Fig 3.8

2. Advance activity diagram

The advanced activity diagram starts from initiation and then user login activity where a signal is sent to the network for request validation and upon confirmation the control flows to order received and then check inventory activity. There are three swimlanes namely inventory manager, accountant and sale agent where update inventory, update payment and generate bill respectively. Then the control flows to the home page and then termination activities.

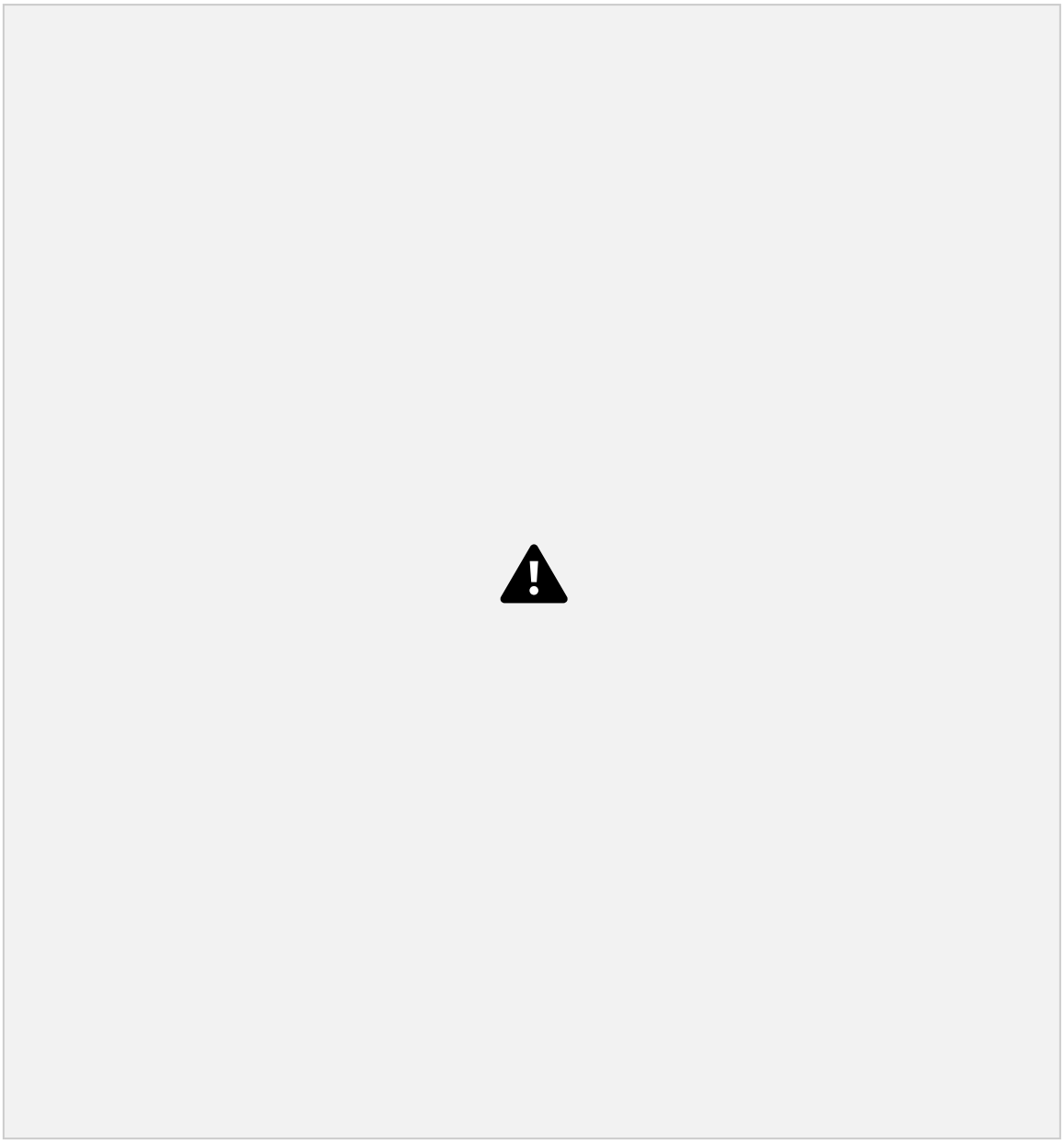

Fig 3.9

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

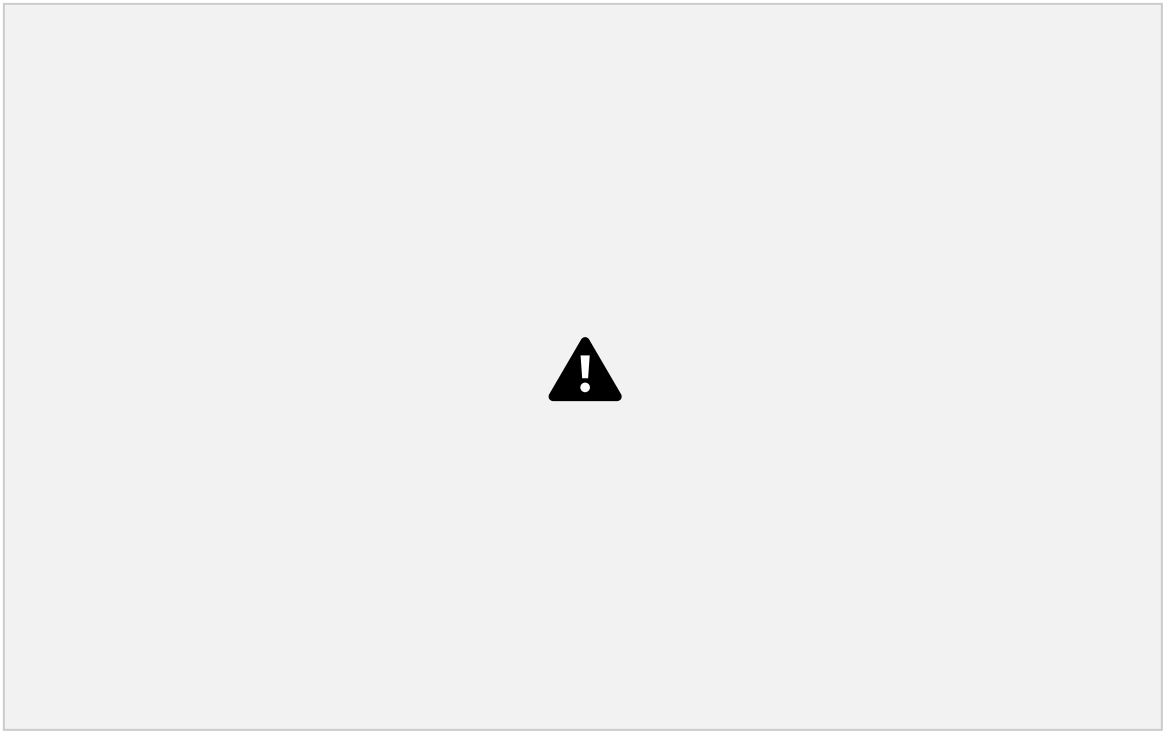

Fig 4.1

State Diagram:

1. Simple state diagram

The simple state diagram has a initial state (Authorization) and final state (Logout). The machine operates depending on the number of customers then the goes to Menu state checks with the raw materials and proceeds towards payment and authentication state, prepare coffee, dispense coffee and beep is any error occurred.

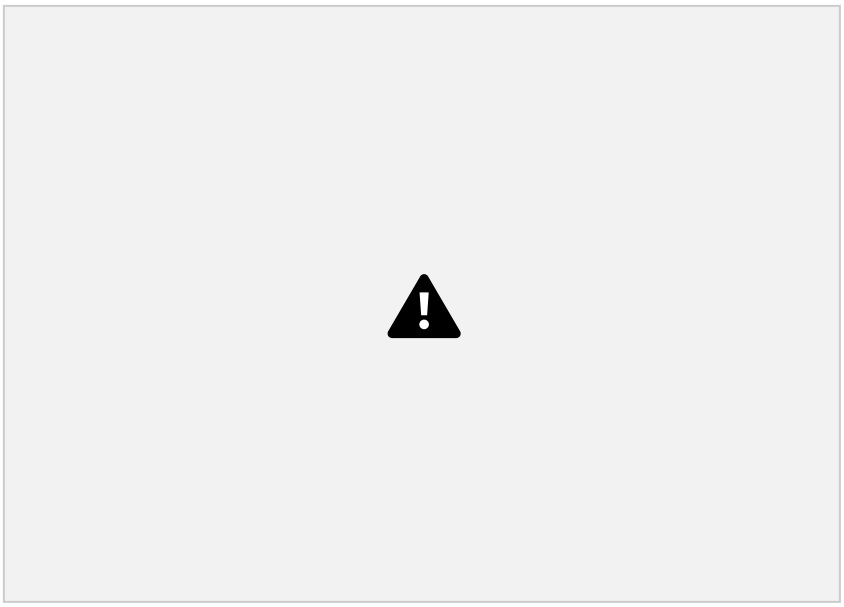

Fig 4.2

2. 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.

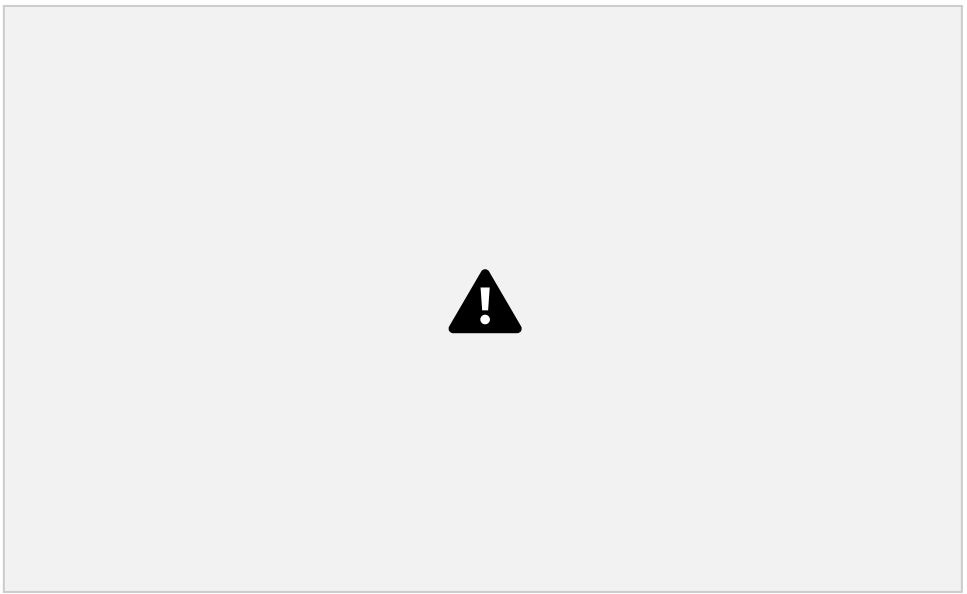

Fig 4.3

Use Case Diagram:

1. Simple use case diagram

The use case diagram for hostel management system has the following:
Actors: Customer, coffee vending, stock clerk
Use cases: Display payment details, request coffee, prepare coffee, make payment, dispense change, dispense coffee, load ingredients.

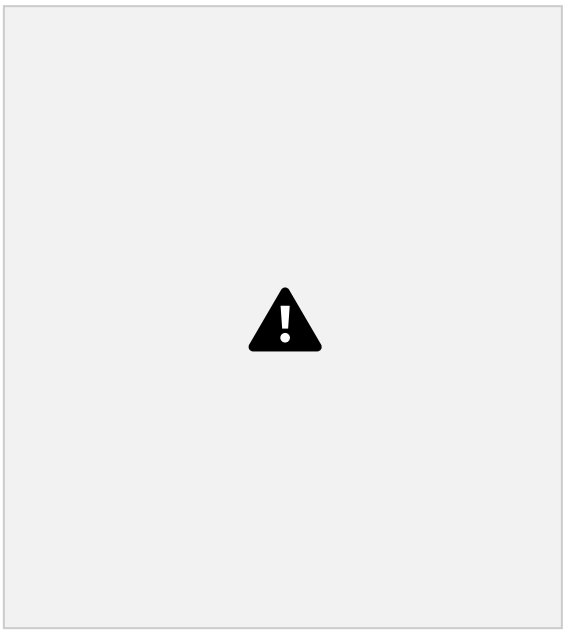

Fig 4.4

2. Advance use case diagram

The advanced use case diagram has extra functionalities which includes extends, includes and generalization. The dispense change use case extends payment use case, payment use case extends buy item use case, buy item use case includes choose item and take item use case. Capucchino dispense and American dispense is generalized to super class dispense coffee.

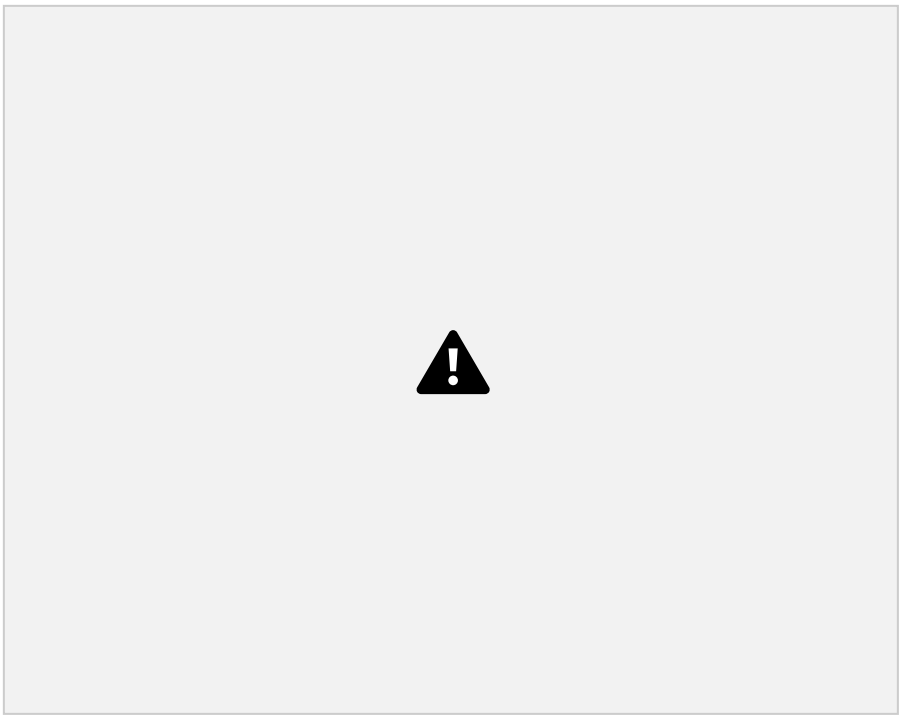

Fig 4.5

Sequence Diagram:

1. Simple sequence diagram

The actors are: User, Coffee Vending Machine and Maintenance System. This scenario of User ordering Coffee and Maintenance System resolving issues contains messages between objects as well as activities performed by these objects.

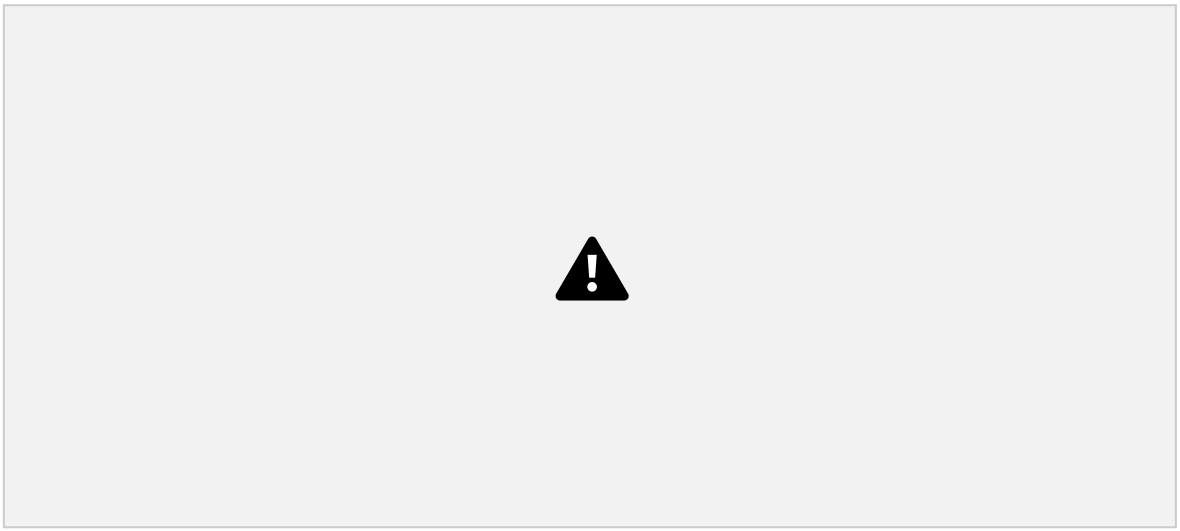

Fig 4.6

2. Advance sequence diagram

The lifeline is the dotted line and the rectangles represent the period of time the object is executing and is hence called activation.
The recursive function of customise is shown by double activation rectangle of customise and verify coins.
The passive object Printer is created when the customer asks for printing and is destroyed (turned off) after sending the receipt.
A time constraint of 1 to 10 seconds is given for depositing coins by the customer in the vending machine.

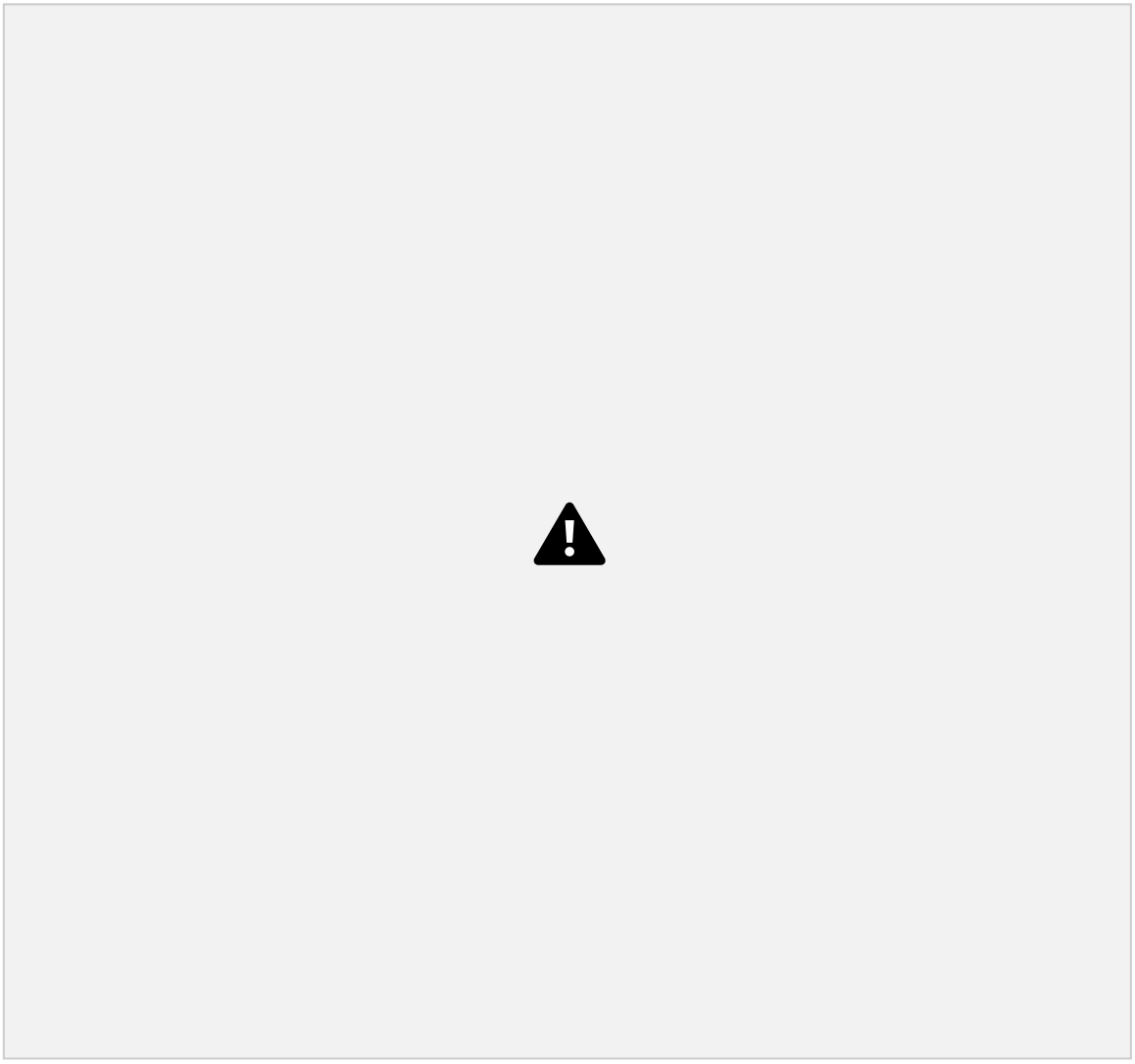

Fig 4.7

Activity Diagram:

1. Simple activity diagram

In the activity diagram, the control flows from the initiation to the login activity. Then the login credentials are validated in the validate activity. A condition is added to check whether the validation is successful or not; if the validation is successful, the control flows to the activity where beverage details and options are displayed whereas on failure control returns to the login activity. A condition is then added to check the user's selection where he/she can see order, get the desired product with coin exchange and dispense coffee then the control flows to the corresponding activities. Then the control flows to the logout activity and then termination.

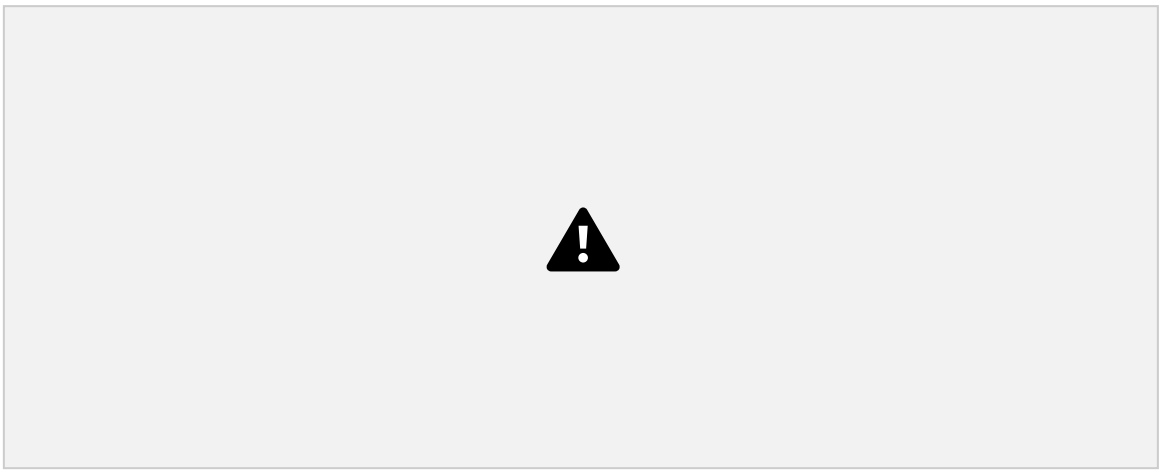

Fig 4.8

2. Advance activity diagram

The advanced activity diagram starts from initiation and in the customer swimlane, customer login activity where a signal is sent to the network for request validation and upon confirmation the control flows to order received and then check inventory activity. There are three swimlanes namely customer, coffee dispenser and payment where customer perform operations like order coffee, dispenses coffee and collect coins respectively. Then the control flows to the home page and then termination activities.

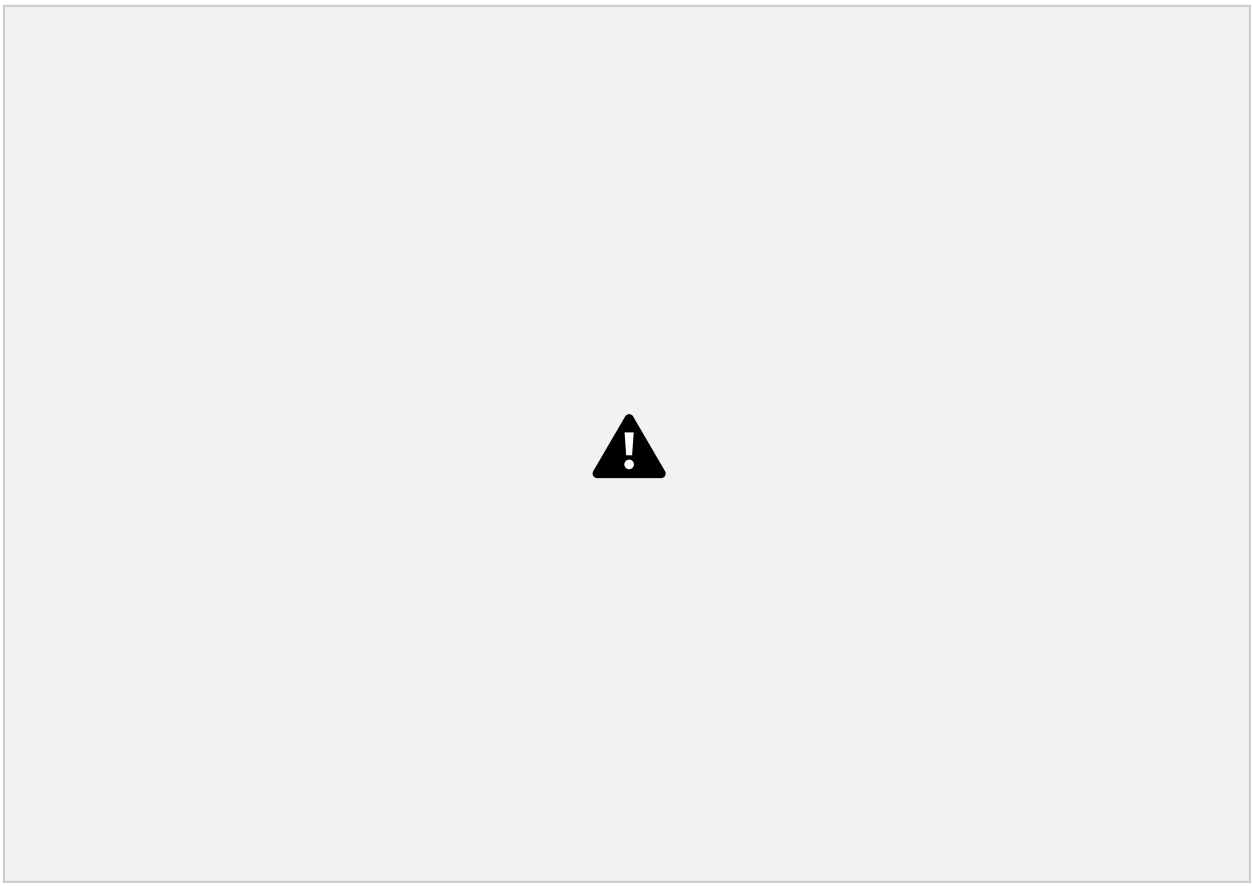

Fig 4.9

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

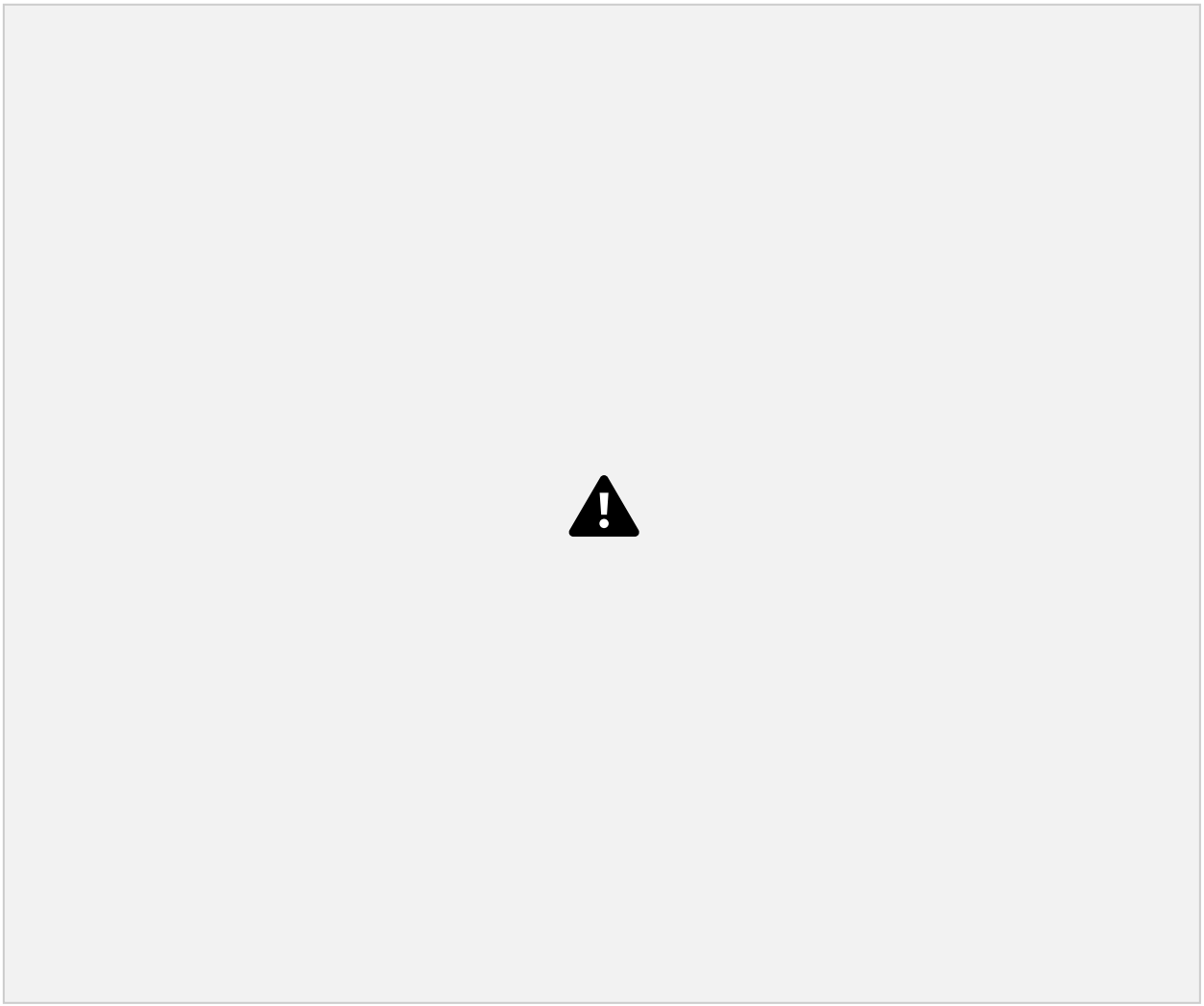

Fig 5.1

State Diagram:

1. Simple state diagram

The simple state diagram has a initial state (Authorization) and final state (Logout). The system operates once the customer logs in and if not then register, search for items, view item, add to cart, update cart if changes, checkout, confirm order and proceed to payment.

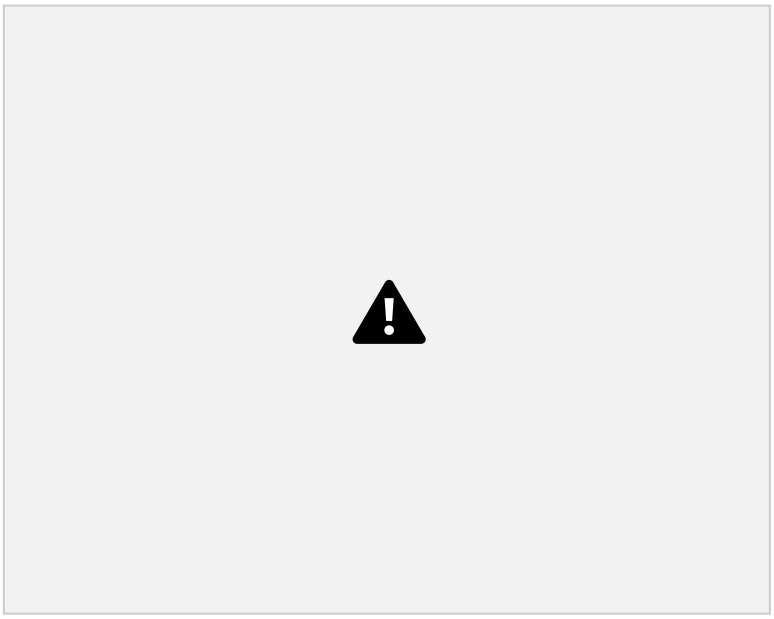

Fig 5.2

2. 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.

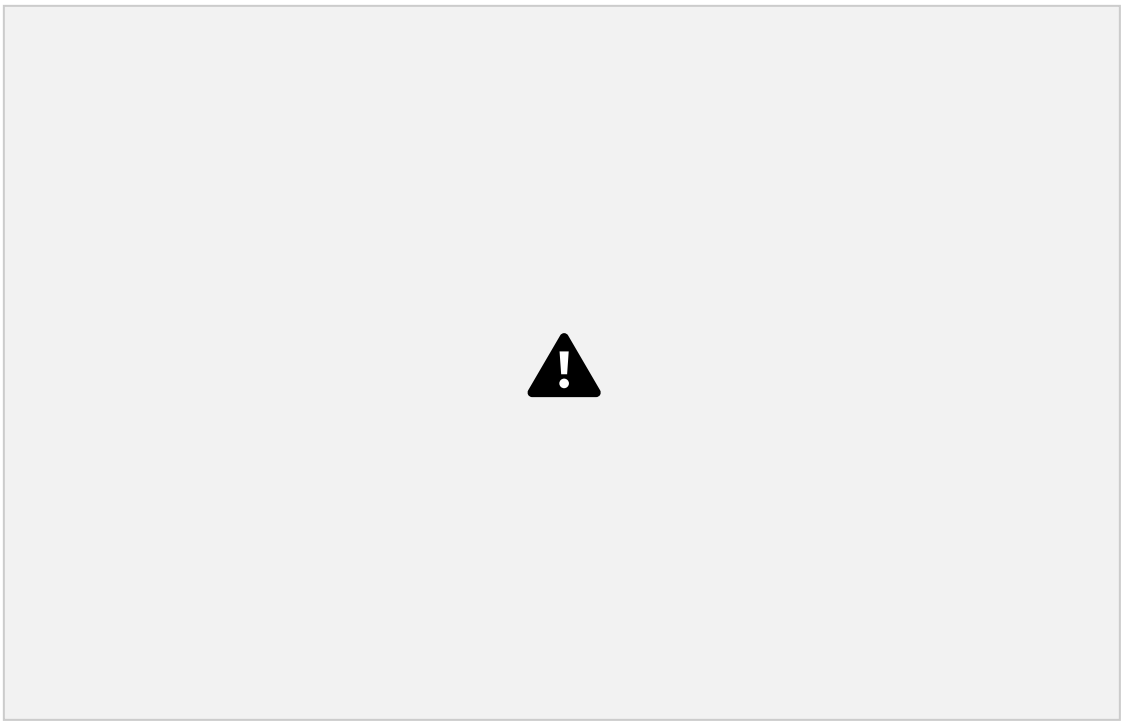

Fig 5.3

Use Case Diagram:

1. Simple use case diagram

The use case diagram for hostel management system has the following: Actors: Customer, supplier, Payment system, Delivery executive.
Use cases: Login, view product details, supply product, place order, make payment, Deliver product, Rate and review product.

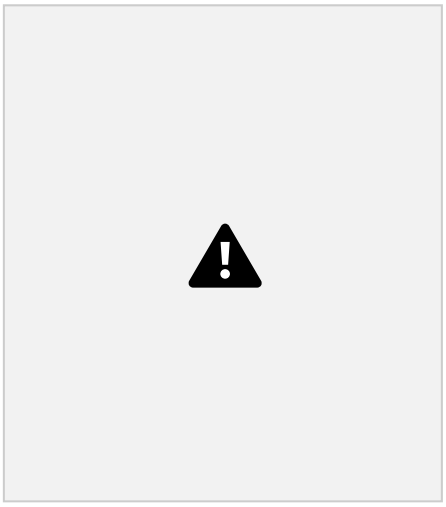

Fig 5.4

2. Advance use case diagram

The advanced use case diagram has extra functionalities which includes extends, includes and generalization. The delete from cart use case extends add item to cart use case, place order use case includes make payment use case, check authentication use case includes make finalized payment and add tax use case. Add item, remove item and update quantity is generalized to super class maintain stock.

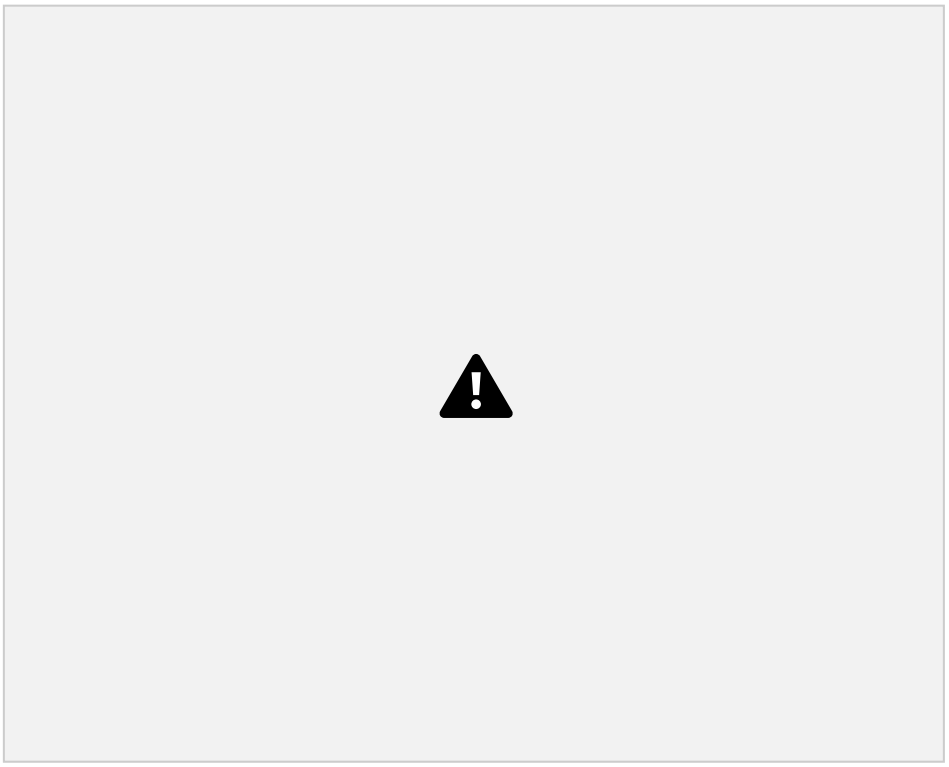

Fig 5.5

Sequence Diagram:

1. Simple sequence diagram

The actors are: Customer and online interface.
This scenario of customer shopping and online maintainance resolving issues contains messages between objects as well as activities performed by these objects.

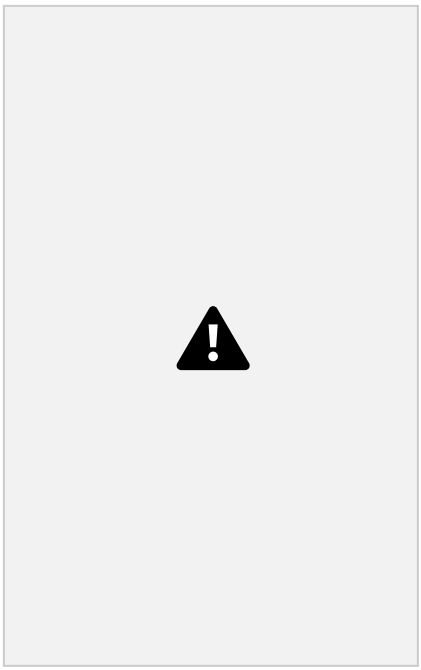

Fig 5.6

2. Advance sequence diagram

The lifeline is the dotted line and the rectangles represent the period of time the object is executing and is hence called activation.

Reply message is used to to return back to lifelines with the required message.

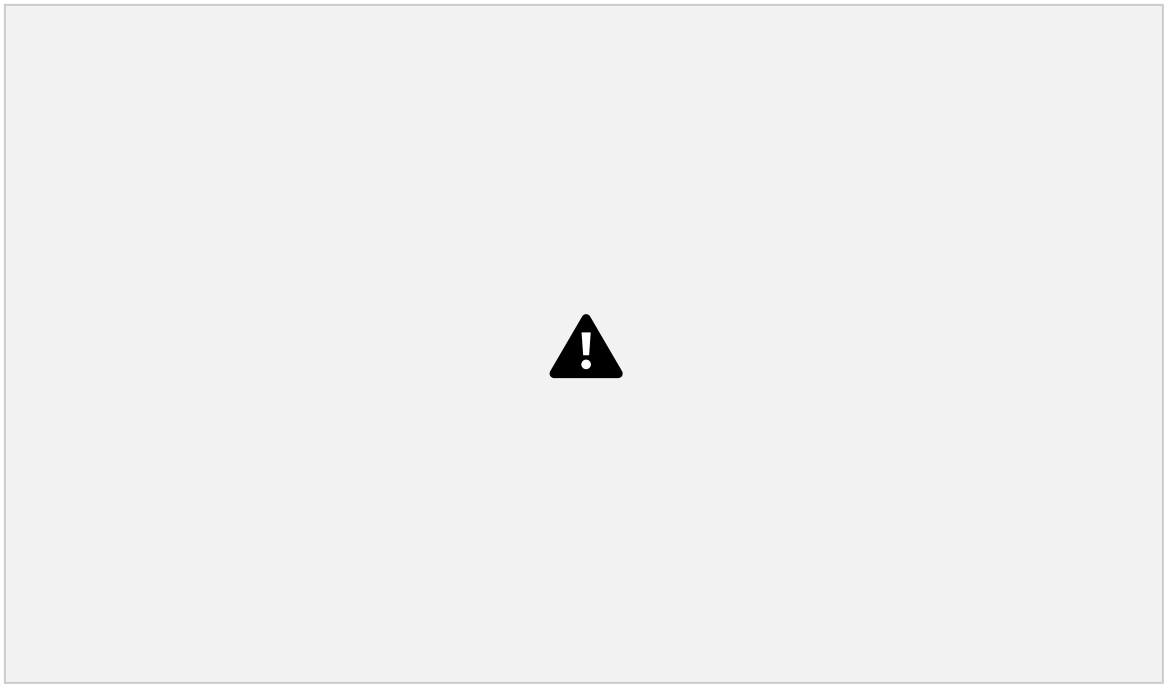

Fig 5.7

Activity Diagram:

1. Simple activity diagram

In the activity diagram, the control flows from the initiation to the login activity. Then the login credentials are validated in the validate activity. A condition is added to check whether the validation is successful or not; if the validation is successful, the control flows to the activity where search product details and options are displayed whereas on failure control returns to the login activity. A condition is then added to check the user's selection which can be to select product, add to cart and then the control flows to the corresponding activities. Then the control flows to the logout activity and then termination.

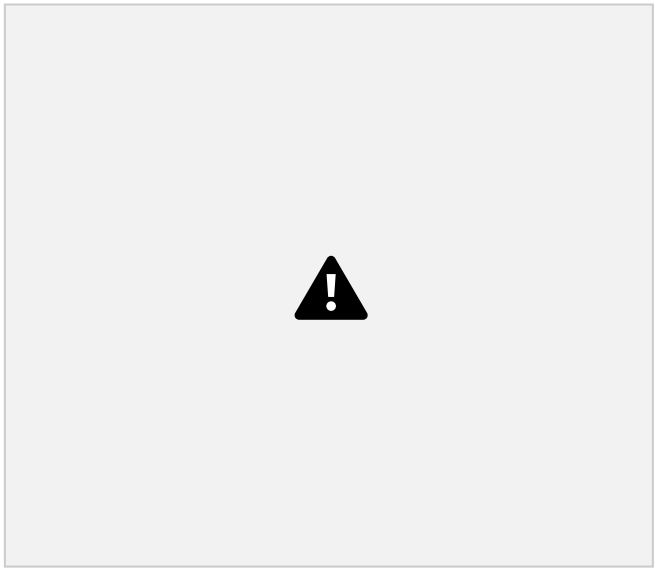

Fig 5.8

2. Advance activity diagram

The advanced activity diagram starts from initiation and in the customer swimlane, the customer login activity where a signal is sent to the network for request validation and upon confirmation the control flows to add product and checkout activity. There are two swimlanes namely customer and online shop where it confirms the order and delivery, payment process respectively. Then the control flows to the home page and then termination activities.

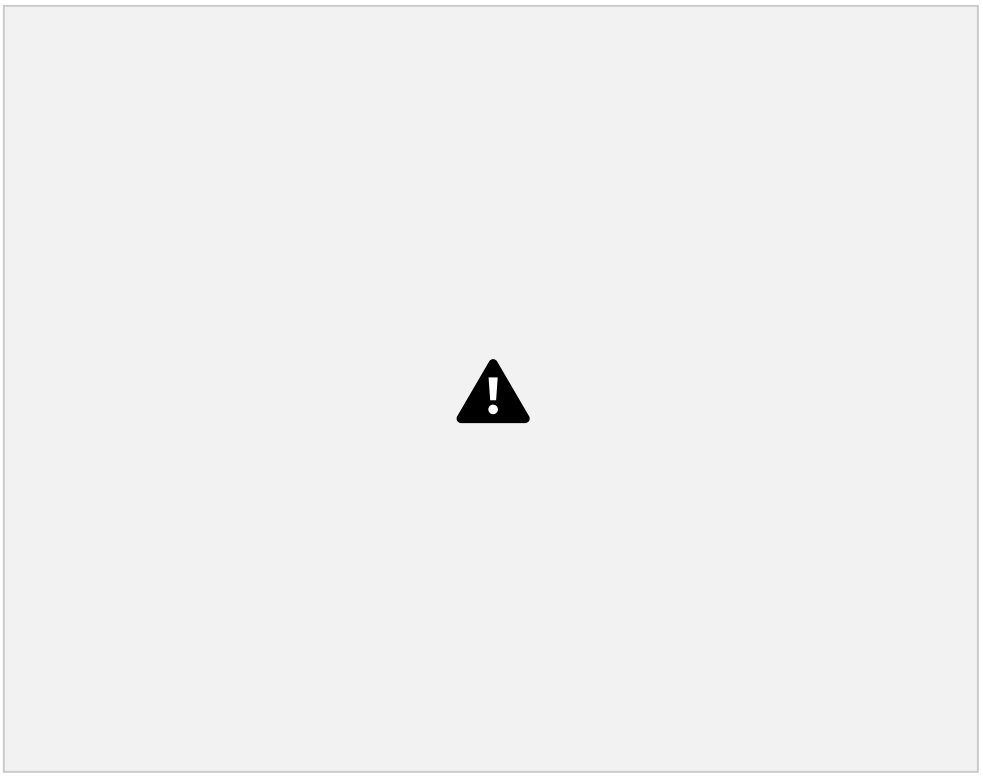

Fig 5.9

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.

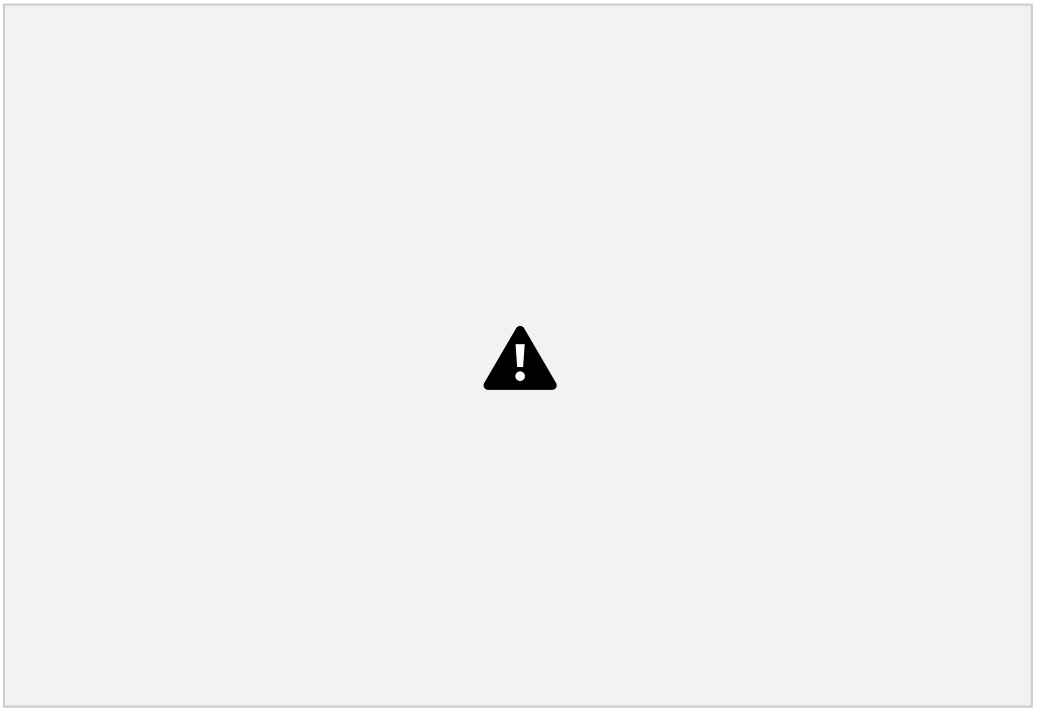

Fig 6.1

State Diagram:

1. Simple state diagram

The simple state diagram has a initial state (Authorization) and final state (Logout) with many other simple states which are: TicketCreation (for existing user), the fork used to verify train and verify passenger then join is used after the operations performed by each state, then other states; payment, ticket printing, ticket issued based on availability of ticket and close of ticket.

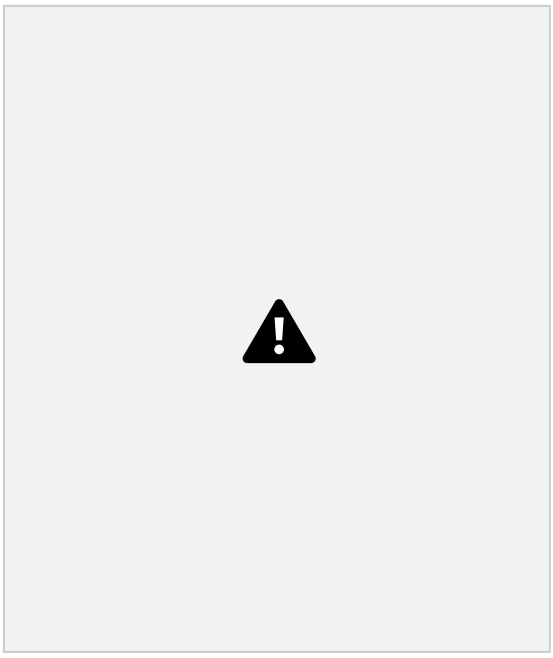

Fig 6.2

2. 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.

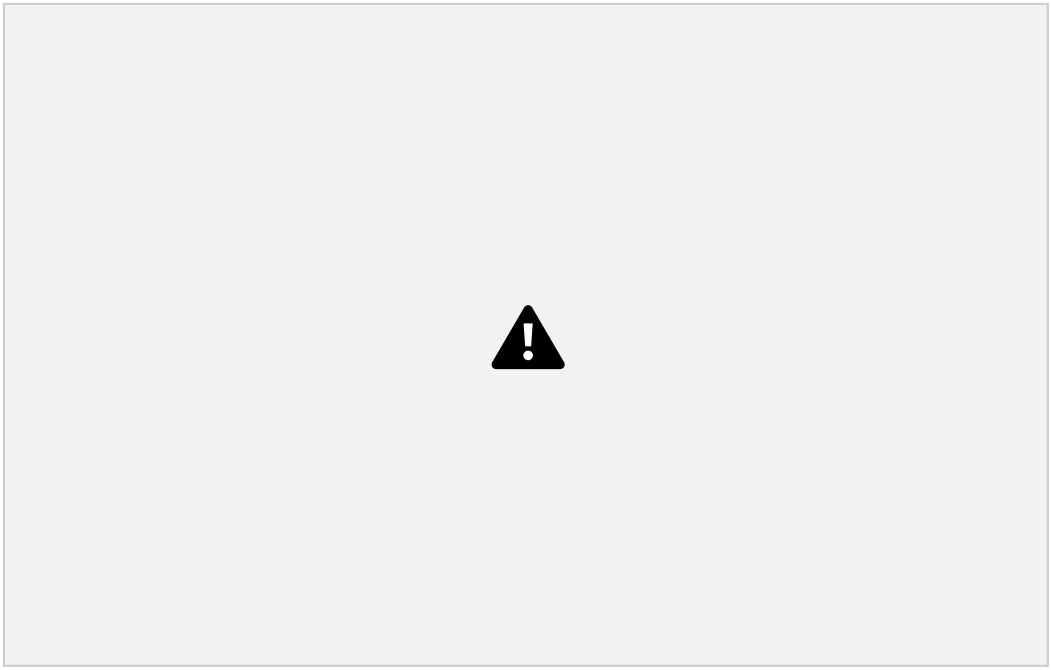

Fig 6.3

Use Case Diagram:

1. Simple use case diagram

The use case diagram for hostel management system has the following:
Actors: User, admin, railway system.
Use cases: Enter login details, register, book ticket, make payment, verify payment, cancel ticket, update train details, refund money, verify login credentials.

Fig 6.4

2. Advance use case diagram

The advanced use case diagram has extra functionalities which includes extends, includes and generalization. The cancel ticket use case extends refund money use case, check availability use case extends book ticket use case, book ticket use case includes fill details use case, fill details use case includes make payment. Online and offline is generalized to super class make payment.

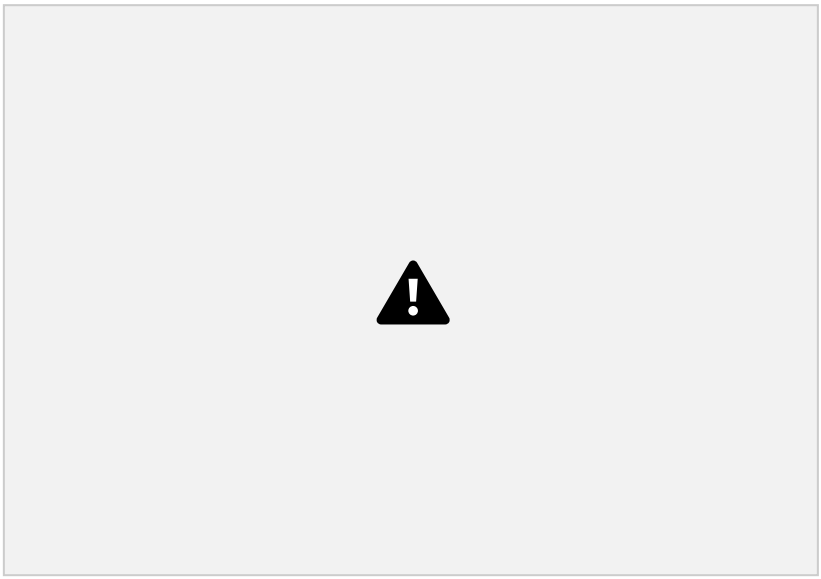

Fig 6.5

Sequence Diagram:

1. Simple sequence diagram

The actors are: User, railway system, admin, bank.
This scenario of User ticket booking and Maintenance System resolving issues contains messages between objects as well as activities performed by these objects.

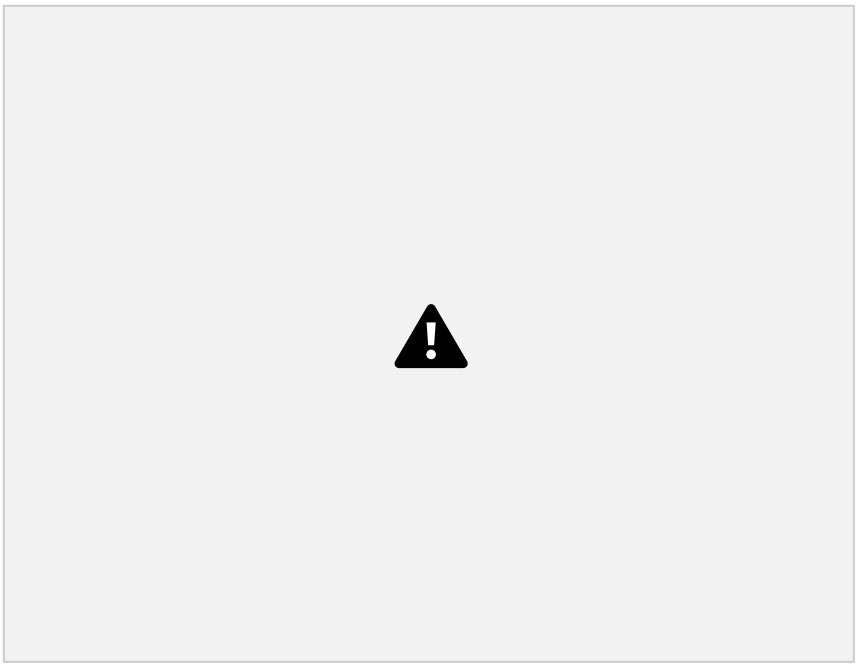

Fig 6.6

2. Advance sequence diagram

The lifeline is the dotted line and the rectangles represent the period of time the object is executing and is hence called activation.
The recursive function of validation is shown by double activation rectangle of validation with self-transition and verify user.Reply message is used to to return back to lifelines with the required message.

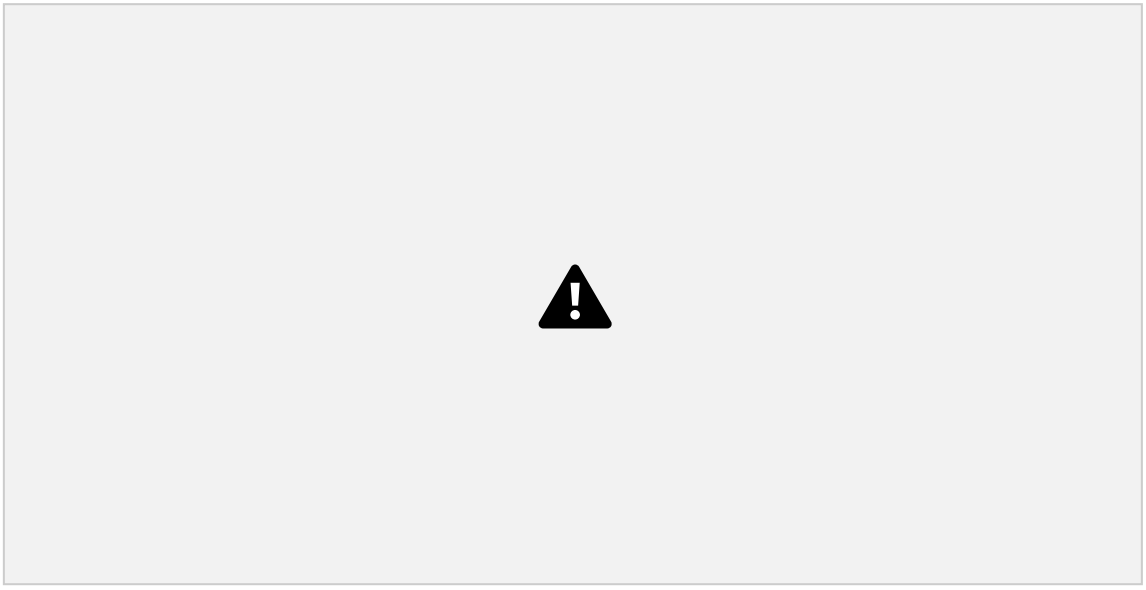

Fig 6.7

Activity Diagram:

1. Simple activity diagram

In the activity diagram, the control flows from the initiation to the login activity. Then the login credentials are validated in the validate activity. A condition is added to check whether the validation is successful or not; if the validation is successful, the control flows to the activity where ticket details and options are displayed whereas on failure control returns to the login activity. A condition is then added to check the user's selection which can be to confiem ticket and payment process with the control flows to the corresponding activities. Then the control flows to the logout activity and then termination.

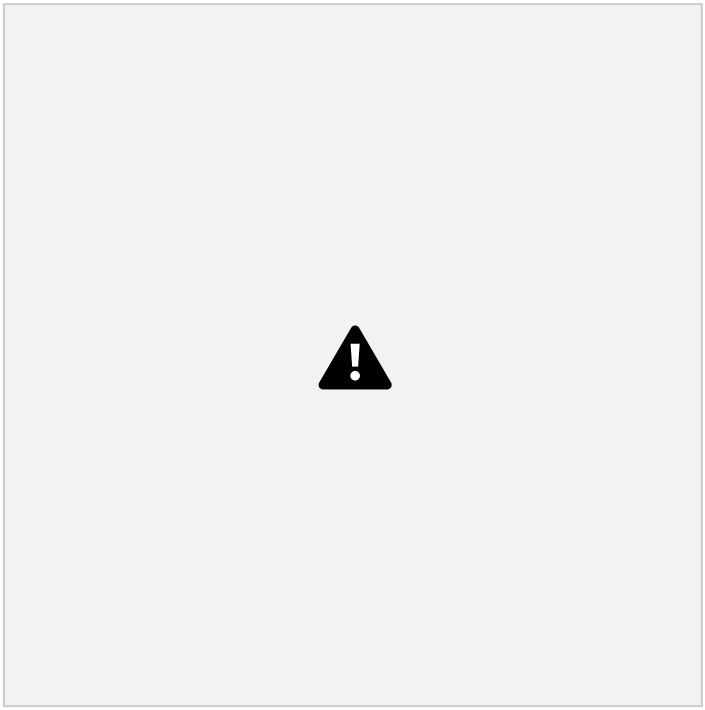

Fig 6.8

2. Advance activity diagram

The advanced activity diagram starts from initiation and in the passenger swimlane, the passenger login activity where a signal is sent to the network for request validation and upon confirmation the control flows to check seat availability activity. There are four swimlanes namely passenger, railway database, railway authority and bank where each one indicates the passenger operations, check seat availability, check validation, confirm payment respectively. Then the control flows to the home page and then termination activities.

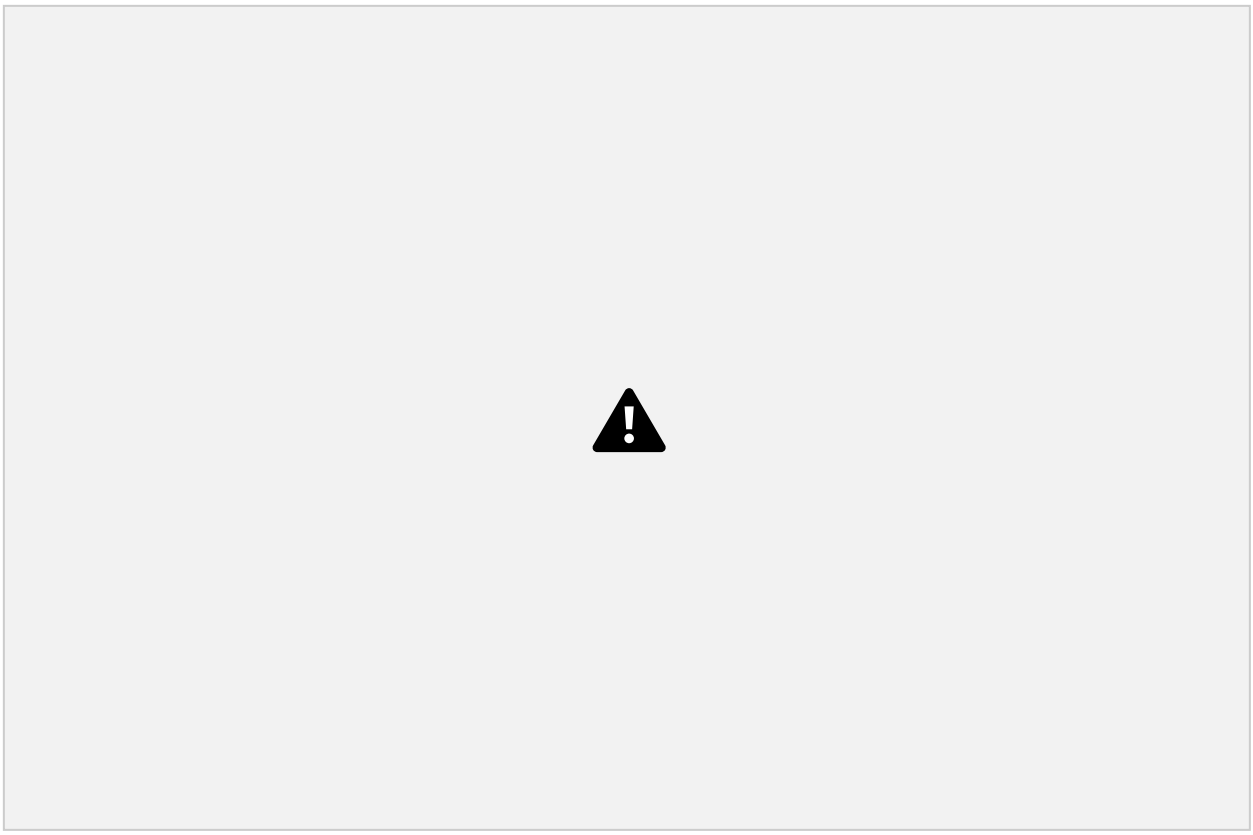

Fig 9.9

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, OneDimension, and TwoDimension are generalized to Object class. Point is generalized to ZeroDimension, Line and Arc are generalized to OneDimension, Circle, Rectangle and Ellipse are generalized to TwoDimension. Composition: GraphicsEditor madeof Document, Document containsOf Sheet, Sheet has Group, Group has Object.
Enumeration: Point

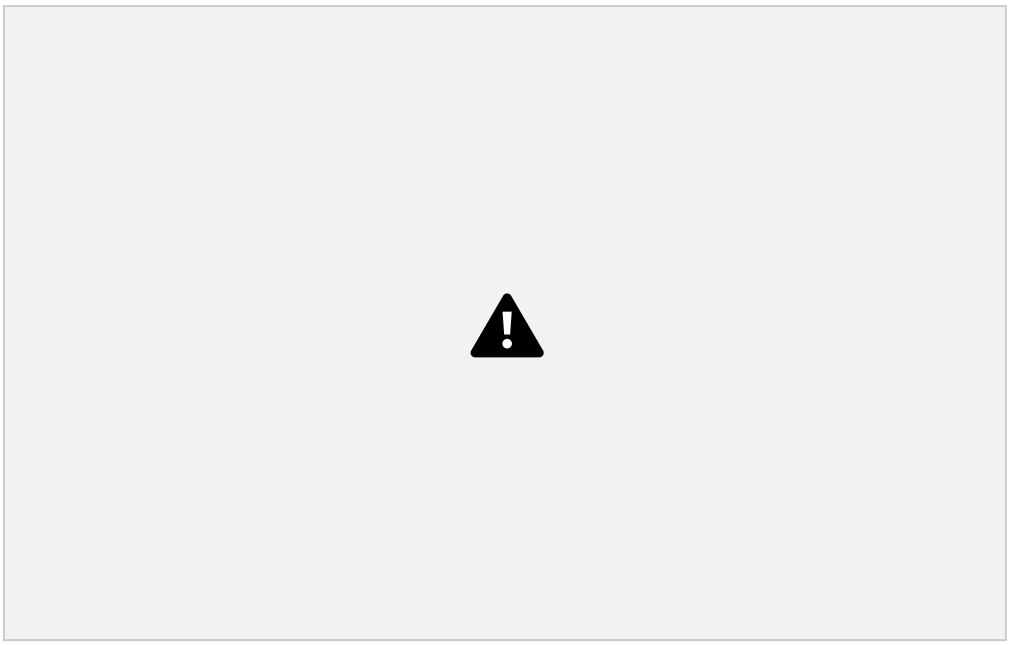

Fig 7.1

State Diagram:

1. Simple state diagram

The simple state diagram has a initial state (Authorization) and final state (Logout) with many other simple states which are: Login, Create Account (if not a member), and go to state open or create document, the fork used to split the states to whether create new doc or open existing doc, then join is used after the operations performed by each state, and start working and then use fork for different shapes after that review the doc file and save.

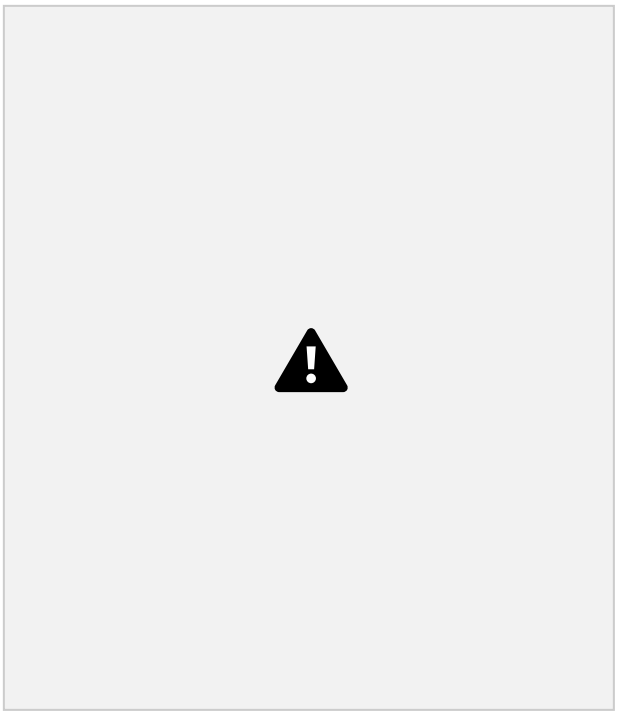

Fig 7.2

2. 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.

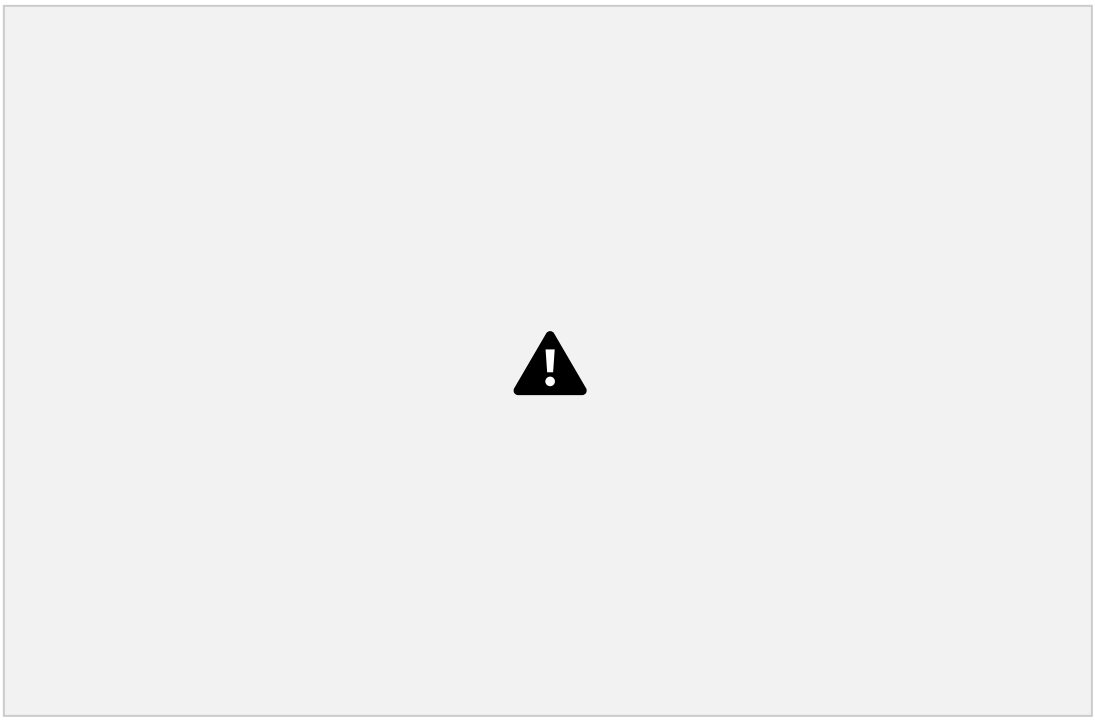

Fig 7.3

Use Case Diagram:

1. Simple use case diagram

The use case diagram for hostel management system has the following:
Actors: User, graphic editor.
Use cases: Edit document, new document, delete graphic tools, color graphic tools, insert graphic tool, save document, delete document.

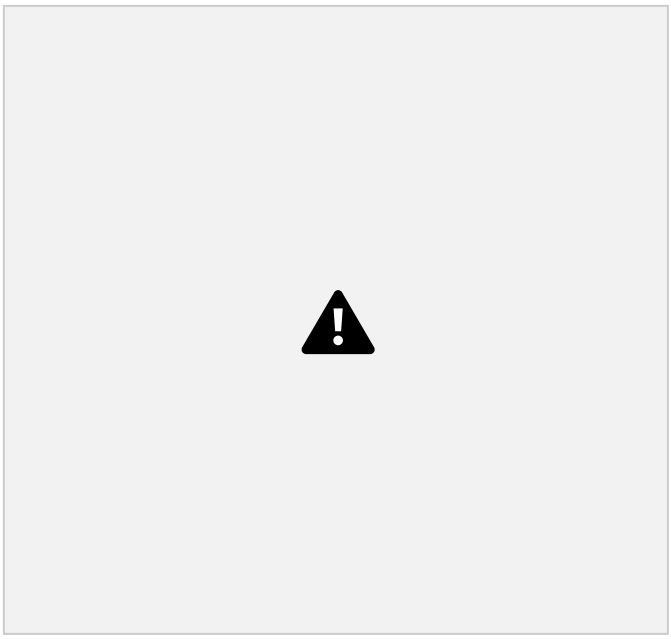

Fig 7.4

2. Advance use case diagram

The advanced use case diagram has extra functionalities which includes extends, includes and generalization. The edit document use case extends new document use case, delete document use case extends new document use case, graphic tools use case extends new document use case, new document use case includes save document use case. Insert, delete and color is generalized to super class graphics tools.

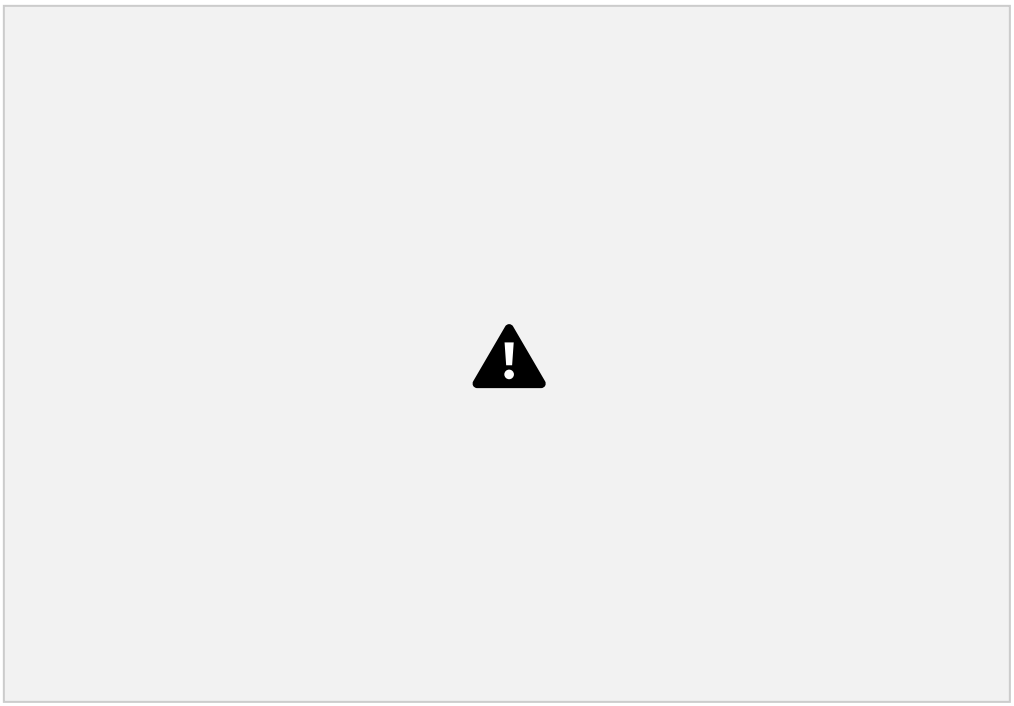

Fig 7.5

Sequence Diagram:

1. Simple sequence diagram

The actors are: User, graphics editor.
This scenario of graphic editing and resolving issues contains messages between objects as well as activities performed by these objects.

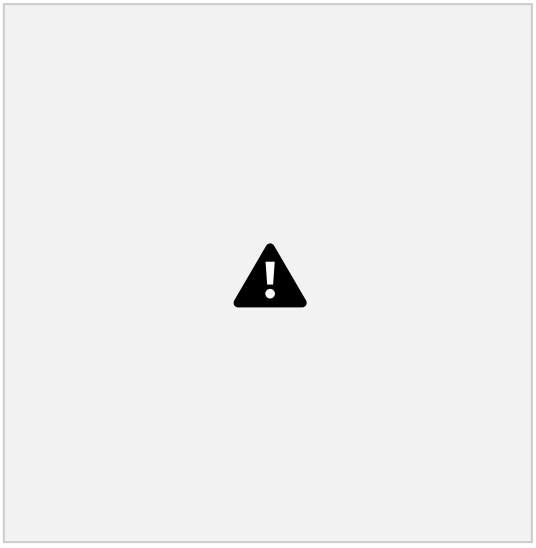

Fig 7.6

2. Advance sequence diagram

The lifeline is the dotted line and the rectangles represent the period of time the object is executing and is hence called activation.

Reply message is used to to return back to lifelines with the required message.

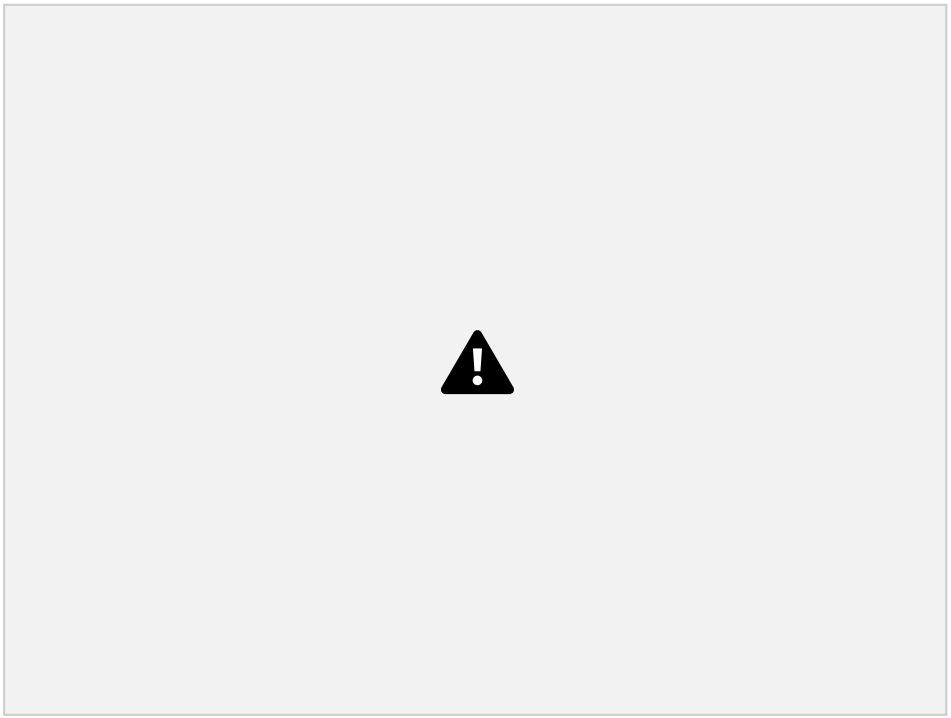

Fig 7.7

Activity Diagram:

1. Simple activity diagram

In the activity diagram, the control flows from the initiation to the login activity. Then the login credentials are validated in the validate activity. A condition is added to check whether the validation is successful or not; if the validation is successful, the control flows to the activity where graphic tool details and options are displayed whereas on failure control returns to the login activity. A condition is then added to check the user's selection which can be to format the document and save the document with the control flows to the corresponding activities. Then the control flows to the logout activity and then termination.

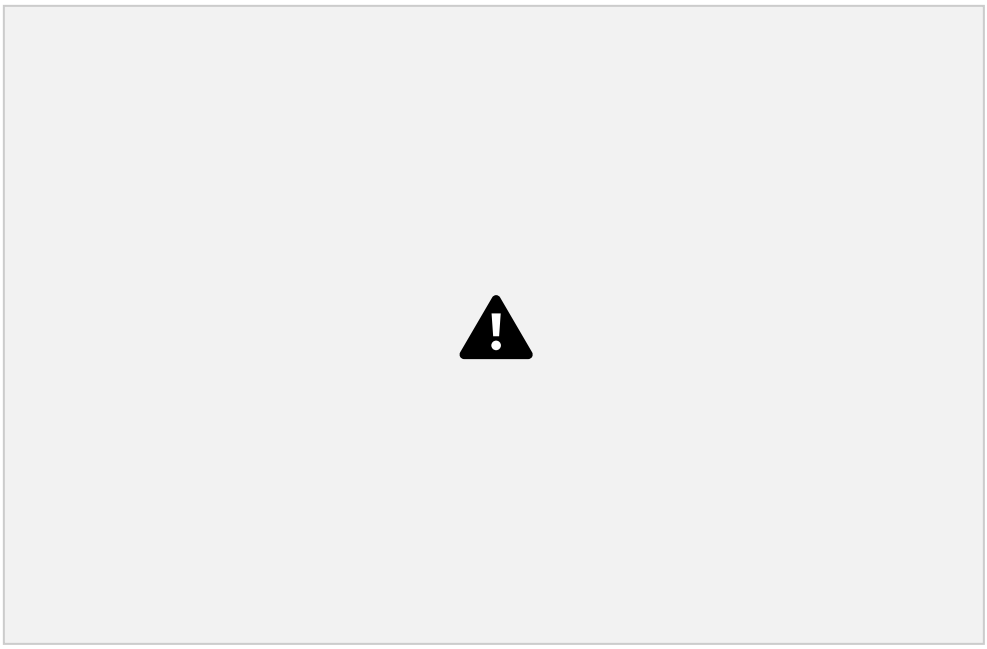

Fig 7.8

2. Advance activity diagram

The advanced activity diagram starts from initiation and in the user swimlane, the user login activity where a signal is sent to the network for request validation and upon confirmation the control flows to open file activity. There are two horizontal swimlanes namely user and editor where each one indicates the user operations and drawing a diagram respectively. Then the control flows to the close file activity and then termination activities.

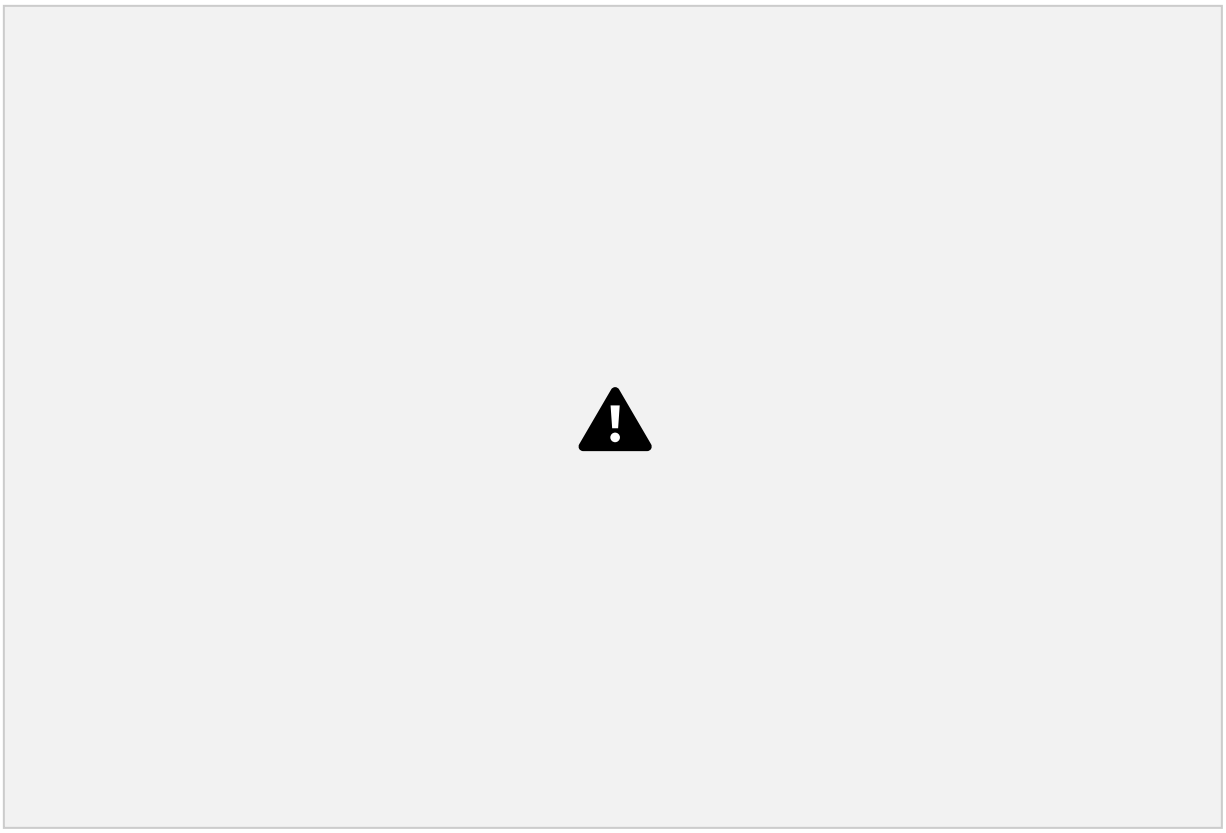

Fig 7.9