1. Demonstration of rational rose 98, AGRO UML and IBM RSA Tools.

Rational Rose 98:
Rational Rose, a design tool from Cupertino, Calif.-based Rational Software Corp., achieves visual modeling through use of the Unified Modeling Language (UML) and component-based development. Rose supports multiple programming languages (C++, Visual Basic and Java) in the same model, and is extensible by way of add-ins and partner products available from third-party firms. Rose also supports the COM/DCOM (ActiveX), JavaBean and Corba component standards.

AGRO UML:
ArgoUML is an open source Unified Modeling Language (UML) modeling tool created in 1998. It includes support for all standard UML 1.4 diagrams. ArgoUML supports other open standards like XMI, SVG or OCL. ArgoUML runs on any Java platform and is available in ten languages

RSA Tools:
Rational Software Architect is a tool that enables software architects to model and design the architecture of their applications. The content that can be created within Rational Software Architect includes all kinds of UML 2.2 diagrams. It also includes features for automatic code generation starting from the models developed within the application. All content created in Rational Software Architect can be published to HTML and deployed to Web servers for distributed viewing. Rational Software Architect can also be connected to a number of other Rational lifecycle process tools in order to be fully used into the software process.

Unified Modeling Language (UML) is a general purpose modelling language. The main aim of UML is to define a standard way to **visualize** the way a system has been designed. It is quite similar to blueprints used in other fields of engineering.
UML is **not a programming language**, it is rather a visual language. We use UML diagrams to portray the **behavior and structure** of a system. UML helps software engineers, businessmen and system architects with modelling, design and analysis. The Object Management Group (OMG) adopted Unified Modelling Language as a standard in 1997. Its been managed by OMG ever since. International Organization for Standardization (ISO) published UML as an approved standard in 2005. UML has been revised over the years and is reviewed periodically.

- Complex applications need collaboration and planning from multiple teams and hence require a clear and concise way to communicate amongst them.
- Businessmen do not understand code. So UML becomes essential to communicate with non programmers essential requirements, functionalities and processes of the system.
- A lot of time is saved down the line when teams are able to visualize processes, user interactions and static structure of the system.

UML is linked with **object oriented** design and analysis. UML makes the use of elements and forms associations between them to form diagrams. Diagrams in UML can be broadly classified as:

1. **Structural Diagrams** – Capture static aspects or structure of a system. Structural Diagrams include: Component Diagrams, Object Diagrams, Class Diagrams and Deployment Diagrams.

2. **Behavior Diagrams** – Capture dynamic aspects or behavior of the system. Behavior diagrams include: Use Case Diagrams, State Diagrams, Activity Diagrams and Interaction Diagrams.

Structural UML Diagrams –

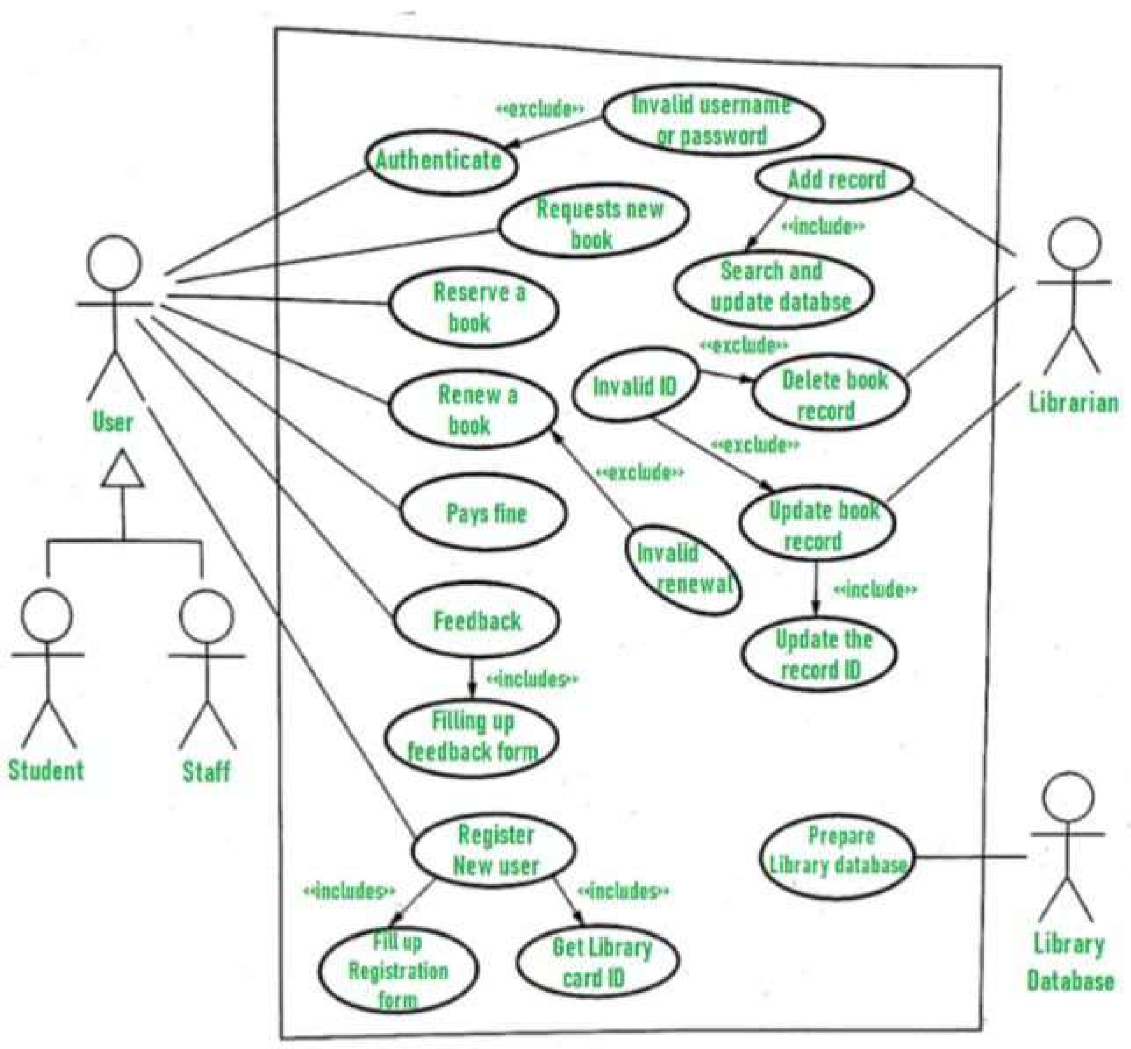
1. **Class Diagram** – The most widely use UML diagram is the class diagram. It is the building block of all object oriented software systems. We use class diagrams to depict the static structure of a system by showing system's classes,their methods and attributes. Class diagrams also help us identify relationship between different classes or objects.
2. **Composite Structure Diagram** – We use composite structure diagrams to represent the internal structure of a class and its interaction points with other parts of the system. A composite structure diagram represents relationship between parts and their configuration which determine how the classifier (class, a component, or a deployment node) behaves. They represent internal structure of a structured classifier making the use of parts, ports, and connectors. We can also model collaborations using composite structure diagrams. They are similar to class diagrams except they represent individual parts in detail as compared to the entire class.
3. **Object Diagram** – An Object Diagram can be referred to as a screenshot of the instances in a system and the relationship that exists between them. Since object diagrams depict behaviour when objects have been instantiated, we are able to study the behaviour of the system at a particular instant. An object diagram is similar to a class diagram except it shows the instances of classes in the system. We depict actual classifiers and their relationships making the use of class diagrams. On the other hand, an Object Diagram represents specific instances of classes and relationships between them at a point of time.
4. **Component Diagram** – Component diagrams are used to represent the how the physical components in a system have been organized. We use them for modelling implementation details. Component Diagrams depict the structural relationship between software system elements and help us in understanding if functional requirements have been covered by planned development. Component Diagrams become essential to use when we design and build complex systems. Interfaces are used by components of the system to communicate with each other.
5. **Deployment Diagram** – Deployment Diagrams are used to represent system hardware and its software. It tells us what hardware components exist and what software components run on them .We illustrate system architecture as distribution of software artifacts over distributed targets. An artifact is the information that is generated by system software. They are primarily used when a software is being used, distributed or deployed over multiple machines with different configurations.
6. **Package Diagram** – We use Package Diagrams to depict how packages and their elements have been organized. A package diagram simply shows us the dependencies between different packages and internal composition of packages. Packages help us to organise UML diagrams into meaningful groups and make the diagram easy to understand. They are primarily used to organise class and use case diagrams.

2. Draw class diagram and use case diagram for Library Management System.

Use case diagram: It is referred as a behaviour model or diagram. It simply describes and displays the relation or interaction between the users or customers and providers of application service or the system. It describes different actions that a system performs in collaboration to achieve something with one or more users of the system. Use case diagram is used a lot nowadays to manage the system.

Here, we will understand the designing use case diagram for the library management system. Some scenarios of the system are as follows:

1. User who registers himself as a new user initially is regarded as staff or student for the library system. For the user to get registered as a new user, registration forms are available that is needed to be fulfilled by the user. After registration, a library card is issued to the user by the librarian. On the library card, an ID is assigned to cardholder or user.
2. After getting the library card, a new book is requested by the user as per there requirement.
3. After, requesting, the desired book or the requested book is reserved by the user that means no other user can request for that book.
4. Now, the user can renew a book that means the user can get a new due date for the desired book if the user has renewed them.
5. If the user somehow forgets to return the book before the due date, then the user pays fine. Or if the user forgets to renew the book till the due date, then the book will be overdue and the user pays fine.
6. User can fill the feedback form available if they want to.
7. Librarian has a key role in this system. Librarian adds the records in the library database about each student or user every time issuing the book or returning the book, or paying fine.
8. Librarian also deletes the record of a particular student if the student leaves the college or passed out from the college. If the book no longer exists in the library, then the record of the particular book is also deleted.
9. Updating database is the important role of Librarian.



Use case diagram for Library Management System

Class Diagram for Library Management System:

Class diagrams are generally used for conceptual modelling of static view of a software application, and for modelling translating models into programming code in a detailed manner. At time of developing or construction software systems, a class diagram is widely used. They are also used for data modelling. It is used to show classes, relationships among them, interface, association, etc. Class in a class diagram simply is a blueprint of an object. It simply describes and explains different type of objects in system, and different types of relationships that exist between them.

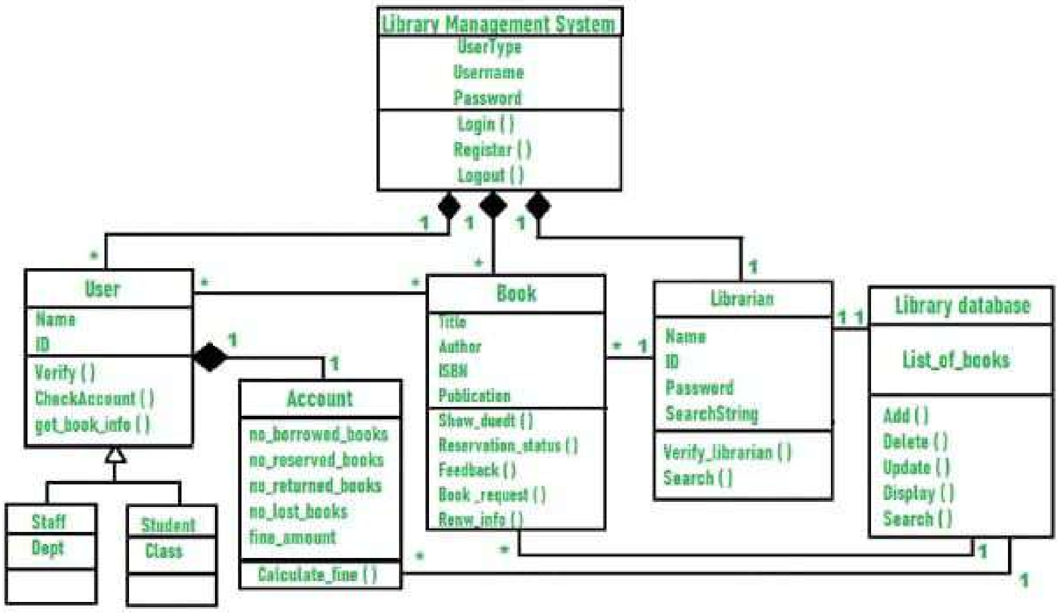
Aggregation and Multiplicity are two important points that need to take into consideration while designing a Class Diagram. Let us understand in detail.

1. Aggregation

   Aggregation simply shows a relationship where one thing can exist independently of other thing. It means to create or compose different abstractions together in defining a class. Aggregation is represented as a part of relationship in class diagram. In diagram given below, we can see that aggregation is represented by an edge with a diamond end pointing towards super class. The "Library Management System" is super class that consists of various classes.

These classes are User, Book, and Librarian as shown in diagram. Further, for "Account" class, "User" is a super class. All of these share a relationship and these relationships are known as aggregate relationships.

2. Multiplicity:
   Multiplicity means that number of elements of a class is associated with another class. These relations can be one-to-one, many-to-many, and many-to-one or one-to-many. For denoting one element we use 1, for zero elements we use 0, and for many elements we use *. We can see in diagram; many users are associated with many books denoted by * and this represents a many-to-many type of relationship. One user has only one account that is denoted by 1 and this represents a one-to-one type of relationship.



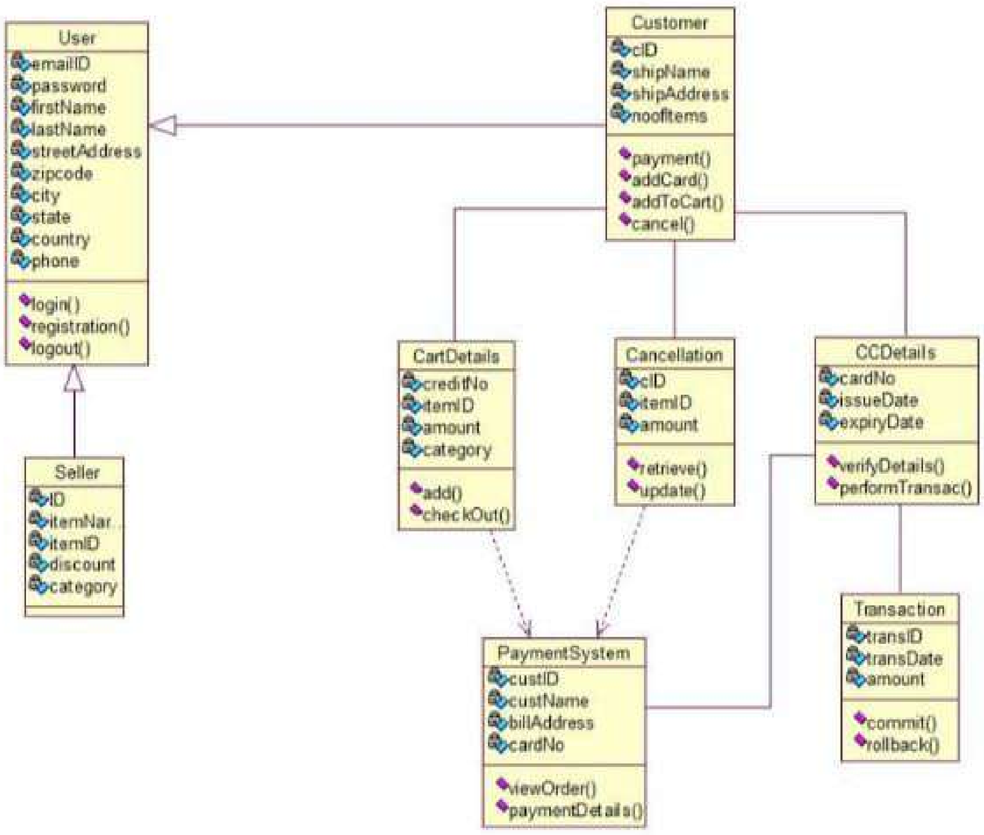
CLASS DIAGRAM FOR LIBRARY MANAGEMENT SYSTEM

[illegible]

3. Draw Class diagram and Use Case Diagram of Online Book Shop.

Class diagram for Online Book Shop: Class diagrams offer a number of benefits for any organization. Use UML class diagrams to:

- Illustrate data models for information systems, no matter how simple or complex.
- Better understand the general overview of the schematics of an application.
- Visually express any specific needs of a system and disseminate that information throughout the business.
- Create detailed charts that highlight any specific code needed to be programmed and implemented to the described structure.
- Provide an implementation-independent description of types used in a system that are later passed between its components.

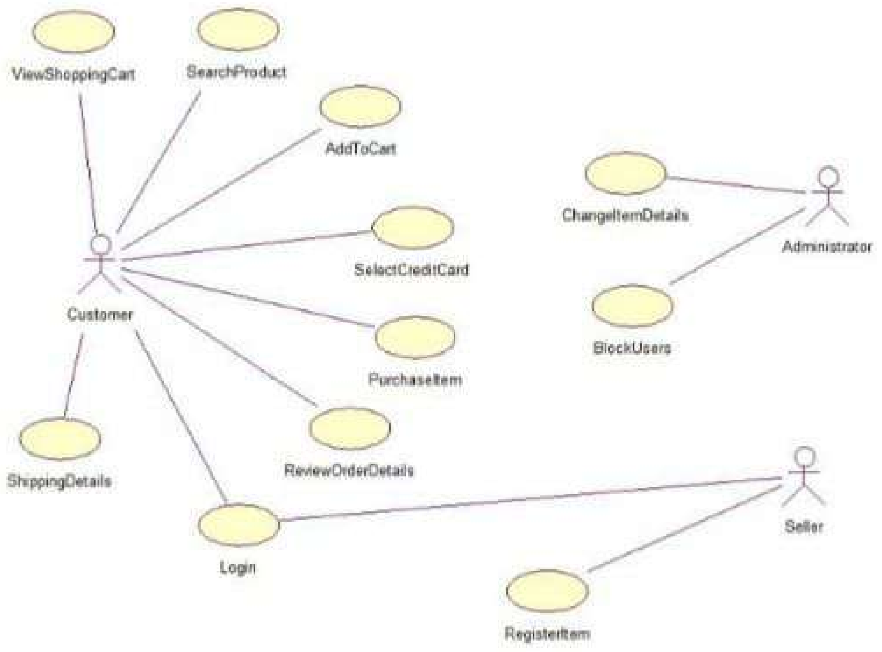


Class diagram for Online Book Shop

Use Case Diagram of Online Book Shop:

The main objective of proposed system, is used to provide Online Book Shopping solution to consumers and vendors. It will automate some of the basic operations of an online store. Scope would be to provide basic functionalities using a web application so that those manual process can be automated. It will include providing administration access to vendors and admins and user specific access to customers.

The system will ease the shopping operations for customer of online store. It will provide vendor or administration functionality to manage categories and products. Consumer will be able to browse and search products under different categories. Selected products or items selected for purchase would be added into the virtual shopping cart. Which can be managed separately by customer. It can be examined at any time by customer for selected products, their quantity & price. These would be main functionalities apart from some usual operations such as login

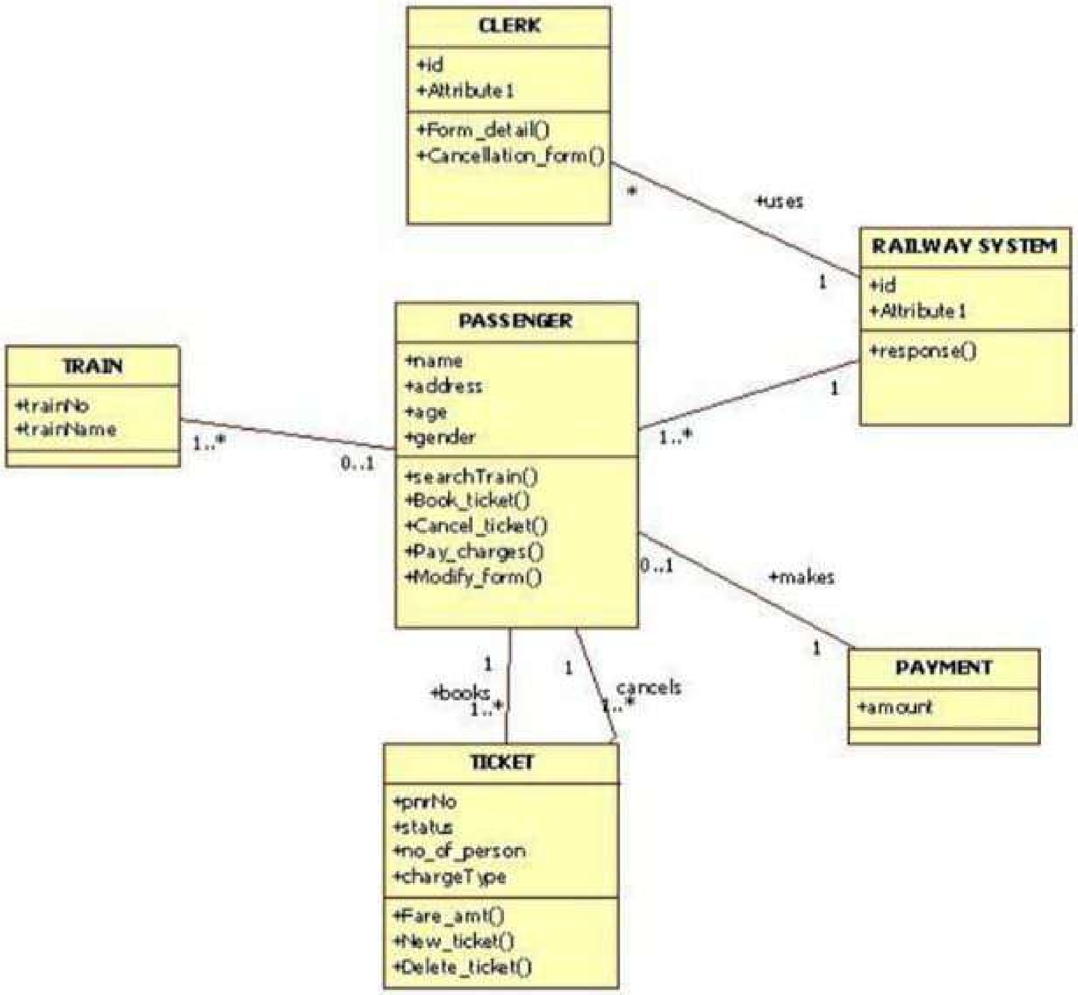


Use Case Diagram of Online Book Shop

4. Draw Class diagram and Use Case Diagram of Railway Reservation System

Class diagram for Railway Reservation System:

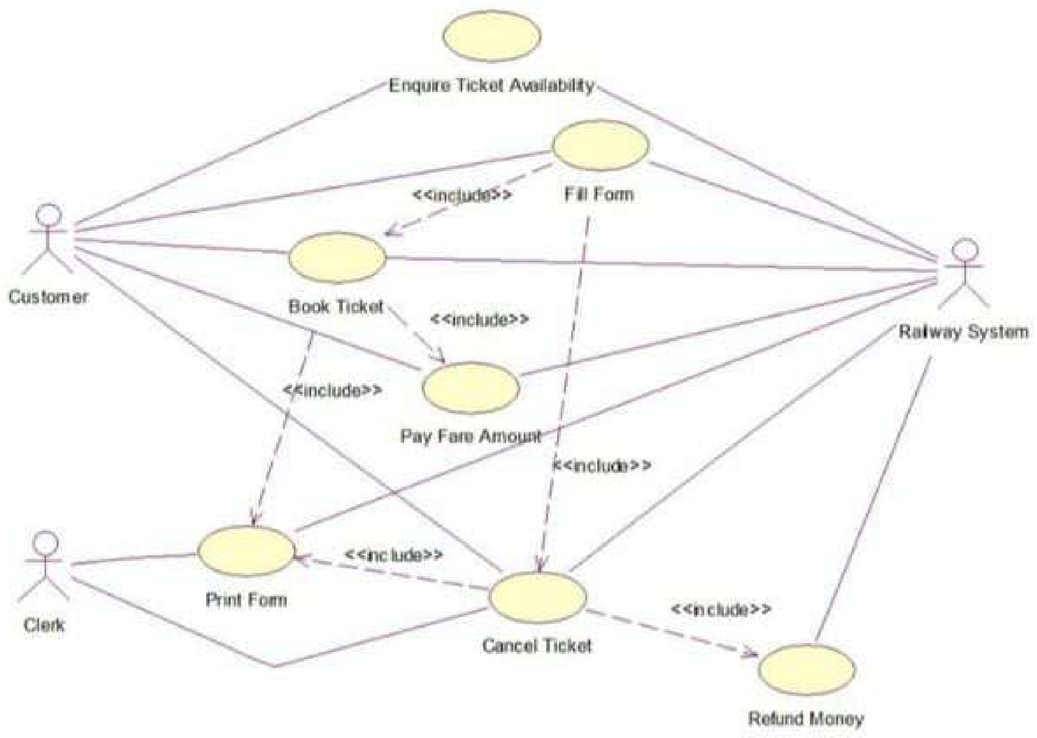
In software engineering, a class diagram in the Unified Modelling Language (UML) is **a type of static structure diagram** that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects. These are used to define different classes in Railway Reservation System. Each one has own importance to define all classes in reservation of railways.



Class diagram for Railway Reservation System

Use Case Diagram of Railway Reservation System:

- The first step is to provide the total number of passengers and submit all the necessary details of the passengers.
- The next step is to enter the source and destination.
- A list of available trains will appear. Among them, the user has to choose one.
- The ticket value will be evaluated. The system will ask to enter the seat choice by showing the seat matrix. At last, a receipt will be generated on the screen.



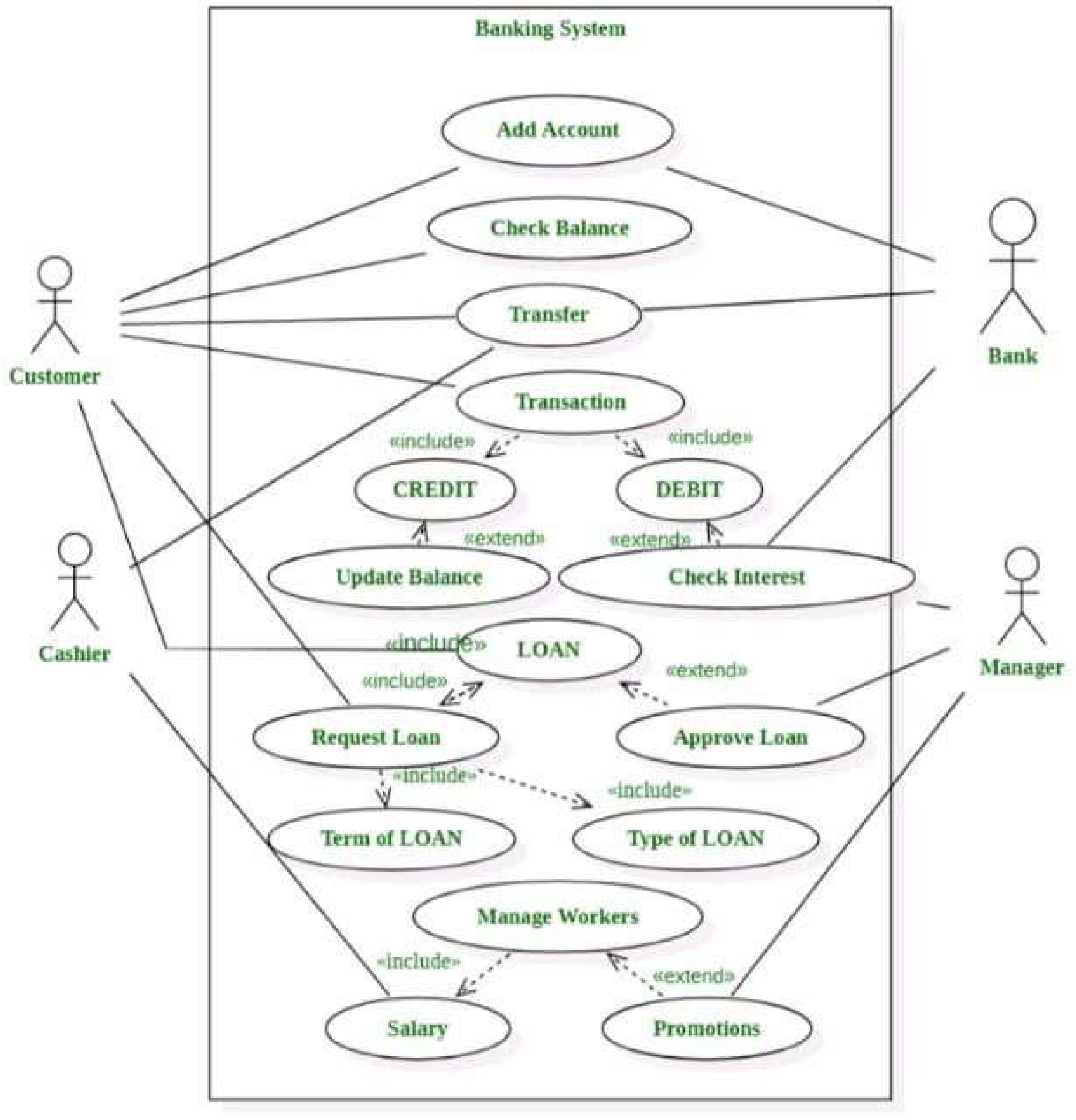
Use Case Diagram of Railway Reservation System

5. Draw Class diagram and Use Case Diagram of Banking System

Use Case Diagram for Banking System:

A user must provide appropriate details to securely Login. Software must check and verify the details at every attempt to Login. Here LOGIN is the Base Use Case and AUTHENTICATE is the **Included Use Case**.
If a user enters appropriate details , user is allowed to Login. However if the details entered by the user are incorrect, software must be able to catch and display problem to the user and allow the user to re-enter details. LOGIN is hence a complete use case. However under certain situations it might use action corresponding to INVALID PASSWORD. Here LOGIN is the Base Use Case and INVALID PASSWORD is **Extended Use Case**.

1. A Customer is required to create an account to avail services offered by Bank. Bank verifies detail and creates new account for each new customer. Each customer is an actor for the Use-Case Diagram and the functionality offered by Online Banking System to Add Account is Use-Case.
2. Each customer can check the balance in bank account and initiate request to transfer an account across distinct branches of Bank. Cashier is an employee at bank who supports service to the customer.
3. A customer can execute cash transactions where the customer must either add cash value to bank account or withdraw cash from account. Either of two or both that is credit as well as debit cash, might be executed to successfully execute one or multiple transactions.
4. After each successful transaction customer might or might not want to get details for action. Manager can check interest value for each account corresponding to transaction to ensure and authenticate details.
5. A customer can also request loan from bank where customer must add request for loan with the appropriate details.
6. The type of loan in accordance with purpose or the need for loan and term or duration to pay back the loan must be provided by customer.
7. The manager of each branch of bank has choice to either accept or approve loan to initiate process further or just reject request for loan based on terms and conditions.
8. The record for each employee of bank is maintained by bank and bank manages all employees of each branch of bank. The manager of each branch has choice to offer bonus to employees. Note here that each employee is paid as part of management of staff but promotion or bonus might or might not be offered certainly to each employee.



Use Case Diagram for Banking System

Class diagram Banking System:

A bank has many branches. In each zone, one branch is designated as the zonal head office that supervises the other branches in that zone. Each branch can have multiple accounts and loans. An account may be either a savings account or a current account. A customer may open both a savings account and a current account. However, a customer must not have more than one savings account or current account. A customer may also procure loans from the bank.

Classes in the system

Bank, Branch, Account, Savings Account, Current Account, Loan, and Customer.

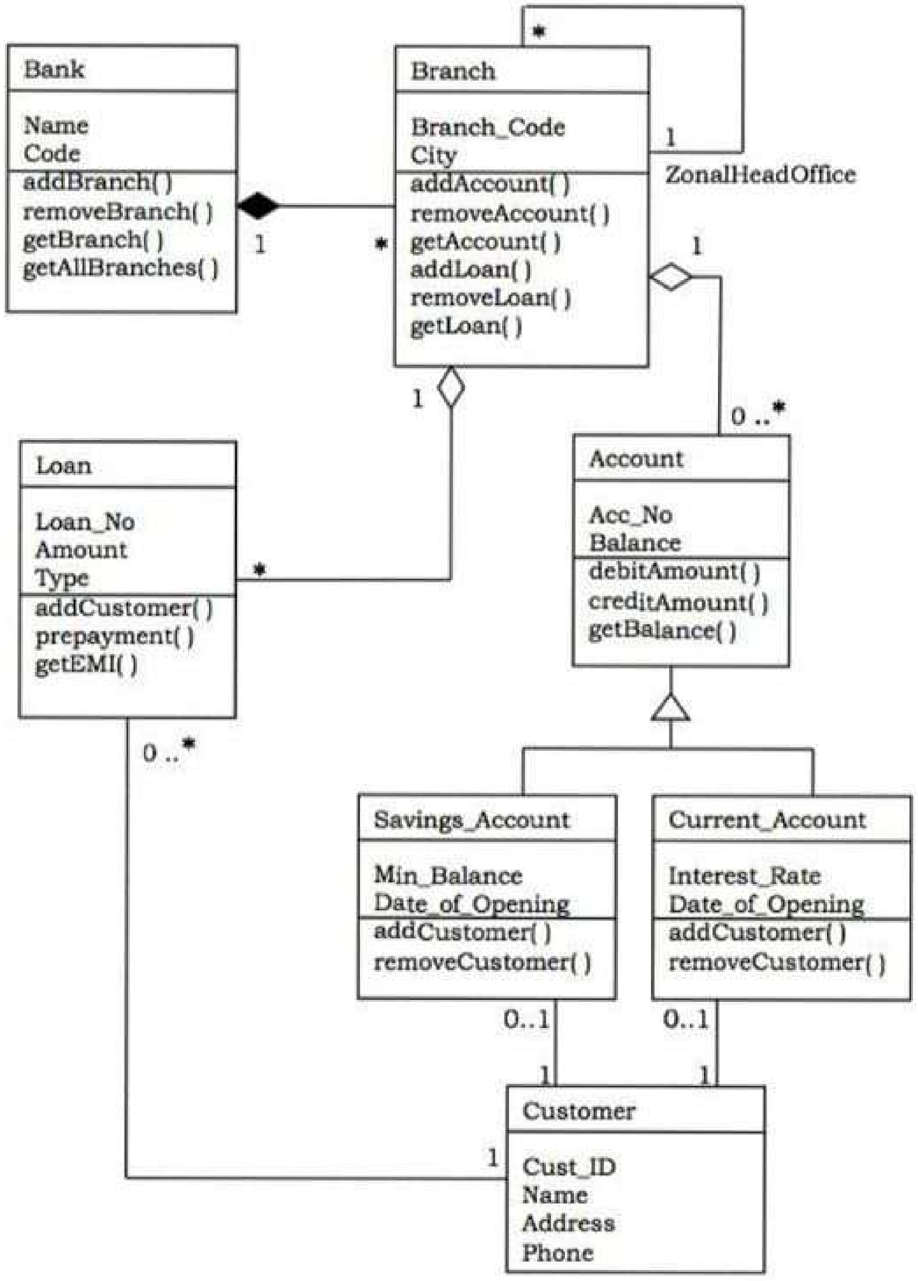
Relationships

- **A Bank "has–a" number of Branches** − composition, one–to–many
- **A Branch with role Zonal Head Office supervises other Branches** − unary association, one–to-many

- **A Branch "has–a" number of accounts** – aggregation, one–to–many

From the class Account, two classes have inherited, namely, Savings Account and Current Account.

- **A Customer can have one Current Account** – association, one–to–one
- **A Customer can have one Savings Account** – association, one–to–one
- **A Branch "has–a" number of Loans** – aggregation, one–to–many
- **A Customer can take many loans** – association, one–to–many



Class diagram Banking System

6. Draw Class diagram and Use Case Diagram for Hotel Management system.

Class diagram is a type of UML diagram which shows the properties and relationships among various objects.

Class:

The classes used in this system are,

- **Hotel Management:** This class depicts the entire hotel and says whether the hotel is opened or closed.
- **Employees :** It contains the details of the Employee. There are two kinds of employees, Server and the chef. This employee class is the parent class of two subclass – Server and Chef
- **Server :** It contains the details of the server, the table to which they are assigned, the order which is currently serving, etc.
- **Chef :** It contains the details of the chef working on a particular order.
- **Customer :** It contains the details of the customer.
- **Table :** It contains the table details like table number and the server who are assigned to that table.
- **Menu :** Menu contains all the food items available in the restaurant, their availability, prize, etc.
- **Order :** Order depicts the order associated with a particular table and the customer.
- **Bill :** Bill is calculated using the order and menu.
- **Payment :** This class is for doing payment. The payment can be done in two ways either cash or card. So payment is the parent class and cash and card are subclasses.
- **Cash :** Payment can be done by cash
- **Card :** Payment can be done by card or online

Attributes :

- **Hotel Management** – HotelName, NumberOfEmployees
- **Employees** – EmployeeId, EmployeeName, EmployeeSalary
- **Server** – ServerId, OrderId
- **Chef** – Chef_Id, OrderId
- **Customer** – CustomerId, CustomerName, Bill_Id, OrderId, PaymentId
- **Table** – TableNumber, OccupiedStatus, ServerId, CustomerId
- **Menu** – ItemId, ItemName, Amount
- **Order** – OrderId, ItemId, ItemName, Quantity, CustomerId, ServerId
- **Bill** – Bill_Id, OrderId, TotalBill
- **Payment** – PaymentId, Bill_Id

Methods:

1. Hotel Management :

- **open()** -This is used to indicate if the hotel is functioning or not.

2. Employees :

- **employee details()** – This method contains the details of the employee.

3. Customer :

- **customer_details()** – This depicts the details of the customer.
- **ordered_items()** – This method contains the items which are ordered by the customer.
- **payment_status()** -This says whether the customer paid or not.

4. Table :

- **table_details()** – This method contains the details of the table along with the customer and no of seats.
- **availability_status()** – This method says whether the table is occupied or not.

5. Menu :

- **items()** – This method displays the menu items, their availability and their price.

6. Order :

- **order_items()** – This method orders the items selected by the user from the menu.

7. Bill :

- **calculate_bill()** – This method calculates the bill for a particular table.

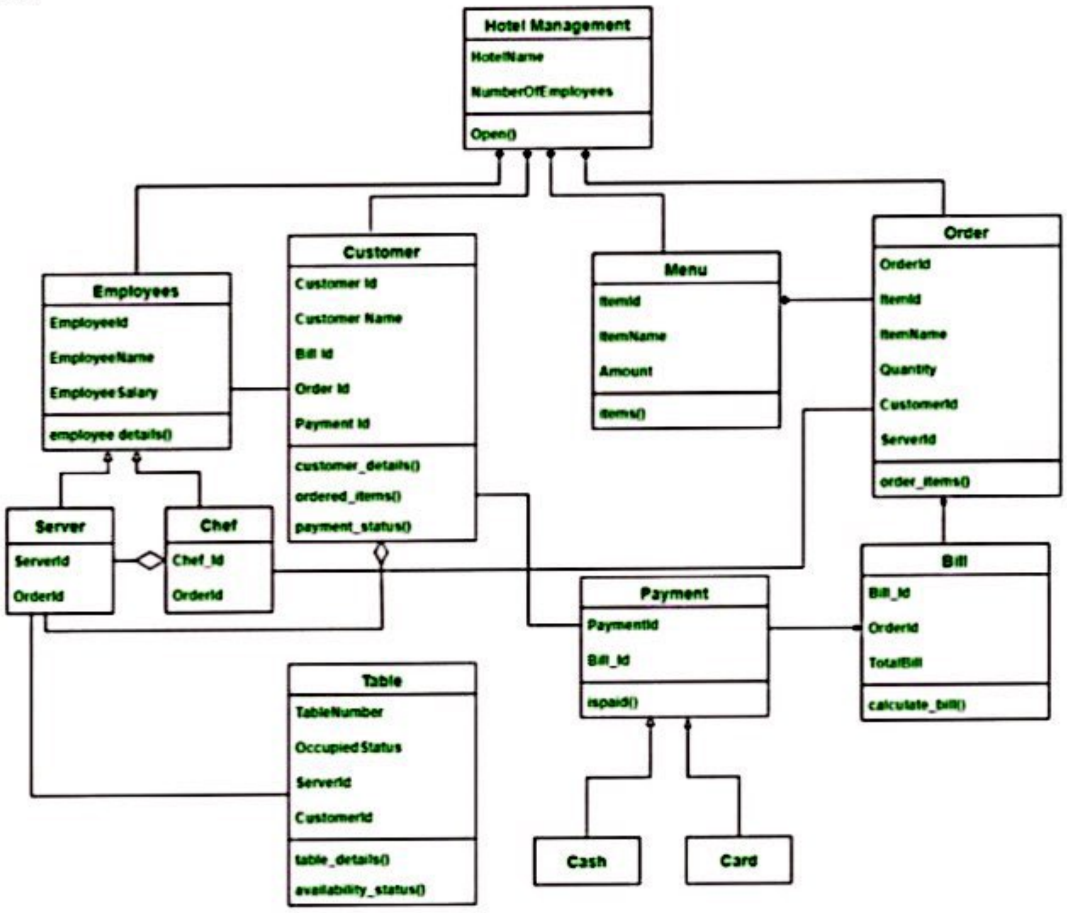
8. Payment :

- **ispaid()** – It shows whether payment is successful or not.

Relationships :

Inheritance :

Inheritance is **"is a relationship"**. It has a parent class and its corresponding child classes. The child class inherits the methods and attributes which are required for it from the parent class.

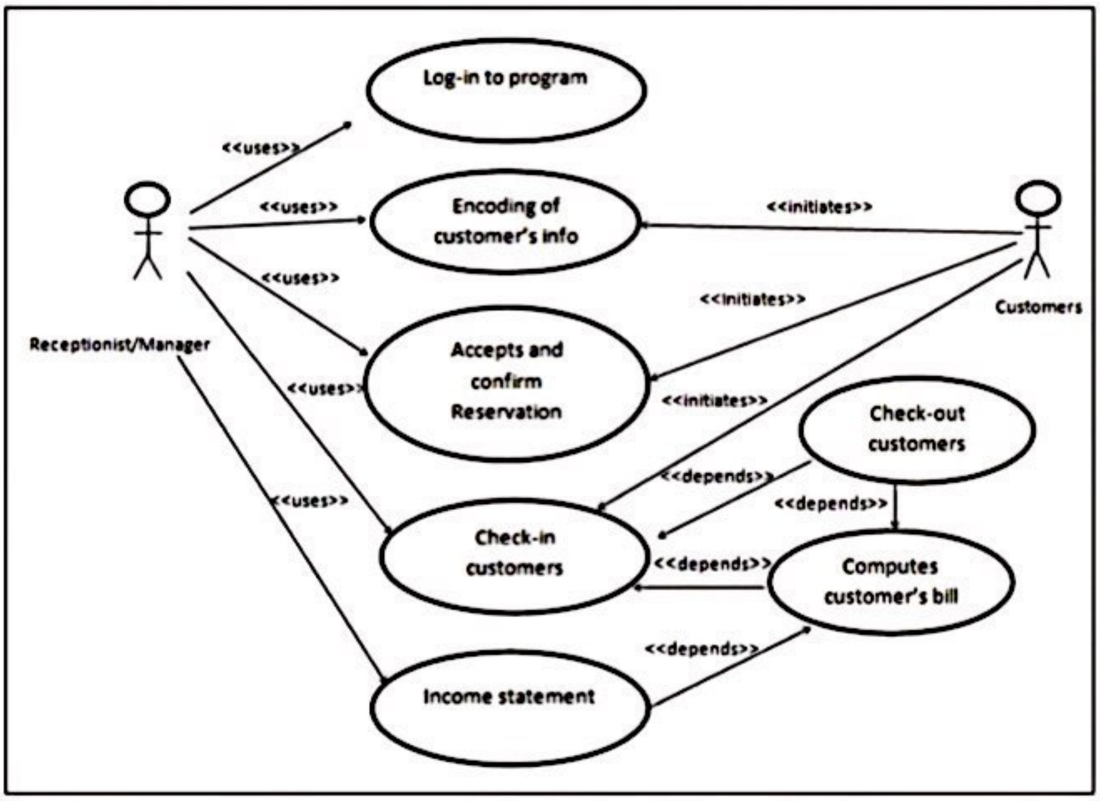


Class diagram for Hotel Management system

Use case diagram for Hotel Management system:

In the use case diagram, you can see how the manager or staff and the customers are linked together. It's important for the user or the actor to be someone who can talk about how things work at an event. It helps a lot for the programmer to figure out how the system works, which helps them, write the code.

The Hotel Management System Use Case Diagram purposes in the people who work in hotels use Hotel Management System, which is a type of properly management system. It helps people run hotel operations and functions like front office, sales, planning and accounting; it also helps people keep track of how much money they make. Use-case diagrams show how a system works and what it can do at a high level. These diagrams also show how the system and its actors work together. Use-case diagrams show what the system does and how the actors use it, but they don't show how the system works on the inside.

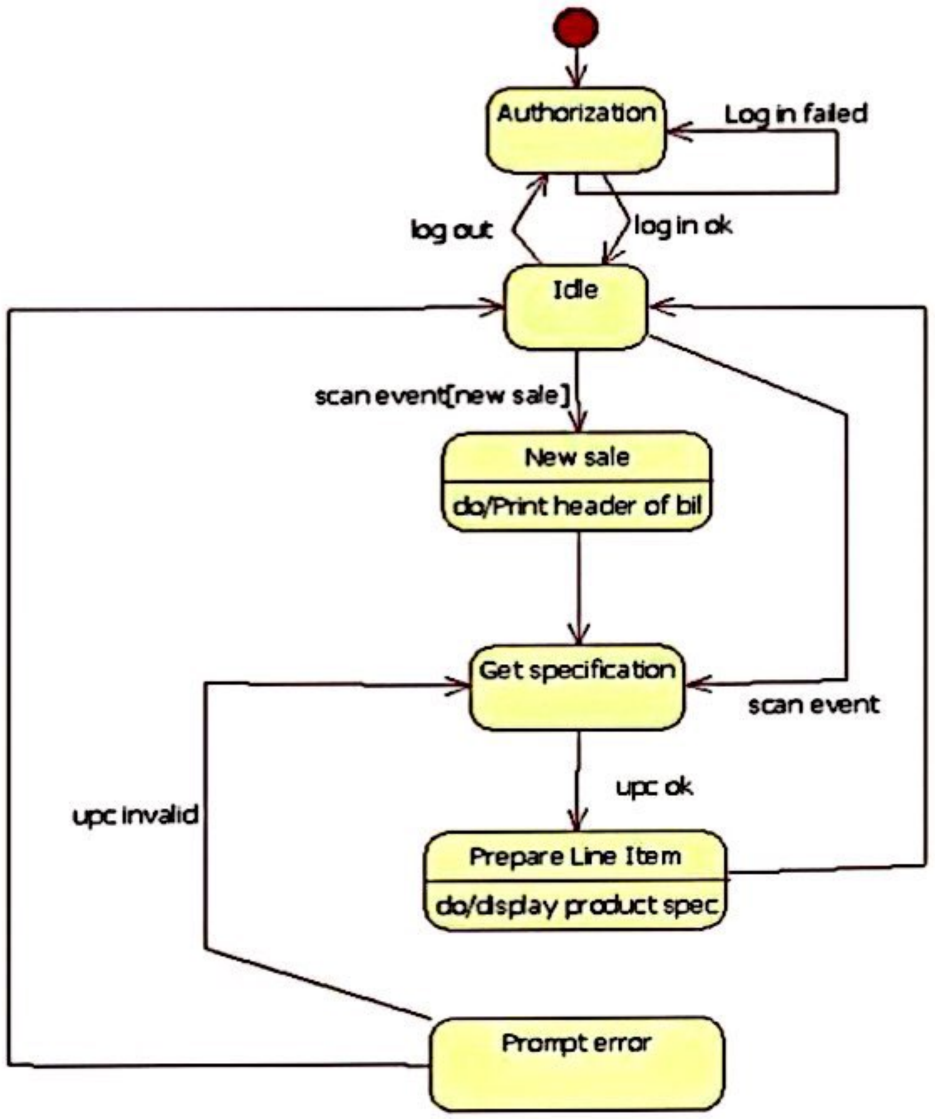


Use case diagram for Hotel Management system

7. Draw State Chart Diagram for Point Sale System.

A point of sale system, or POS, is the place where your customer makes a payment for products or services at your store. Simply put, every time a customer makes a purchase, they're completing a point of sale transaction.

The latest point of sale software goes beyond credit card processing to help retailers and restaurants incorporate mobile POS features and contactless payment options, ecommerce integration capabilities, and more. At Software Advice, our advisors help small business software buyers find the right retail POS software every day. We asked senior advisor Julia Morton, who helps POS software buyers, about the importance of a good point of sale system, and here's what she had to say:



State Chart Diagram for Point Sale System

8. Draw State Chart Diagram for Library Management System.

State chart diagram is one of the five UML diagrams used to model the dynamic nature of a system. They define different states of an object during its lifetime and these states are changed by events. State chart diagrams are useful to model the reactive systems. Reactive systems can be defined as a system that responds to external or internal events.

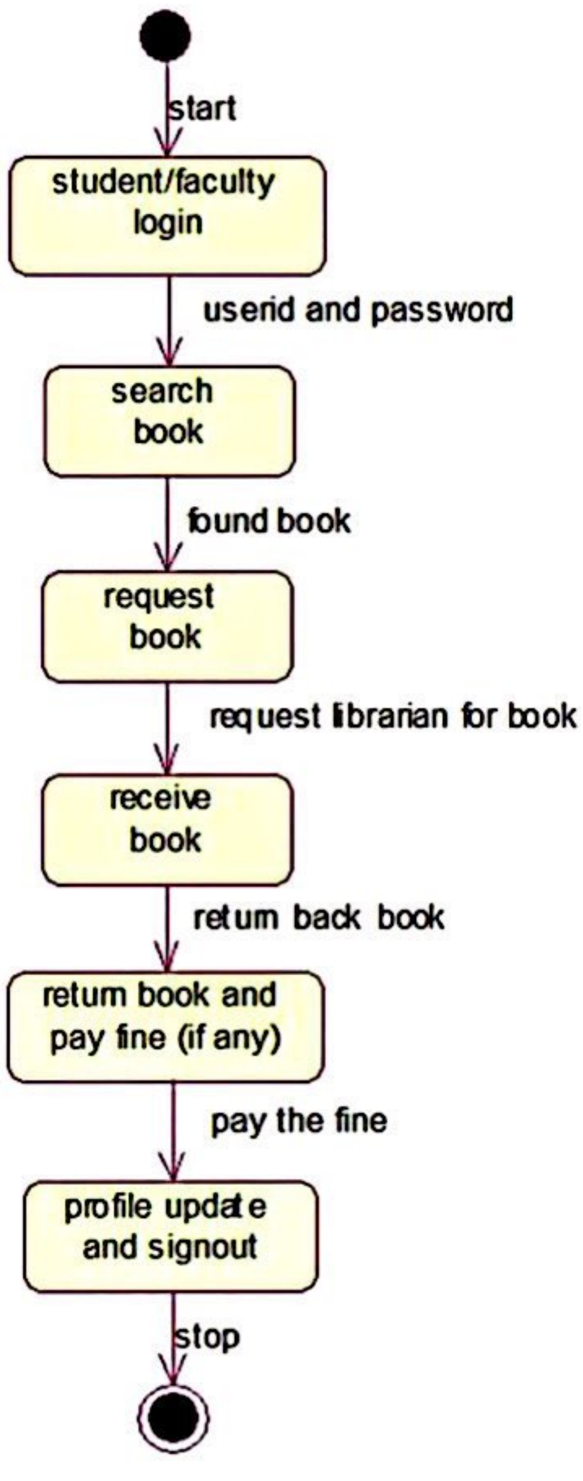
State chart diagram describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered. The most important purpose of State chart diagram is to model lifetime of an object from creation to termination.

State chart diagrams are also used for forward and reverse engineering of a system. However, the main purpose is to model the reactive system.

Following are the main purposes of using State chart diagrams –

- To model the dynamic aspect of a system.
- To model the life time of a reactive system.
- To describe different states of an object during its life time.
- Define a state machine to model the states of an object.

State chart diagram is also called as state machine diagram. The state chart diagram contains the states in the rectangular boxes and the states are indicated by the dot enclosed. The state chart diagram describes the behaviour of the system. The state chart diagram involves eight stages such as login, enter details, requesting for book, display book details, search book, issue book, and return book and logout.



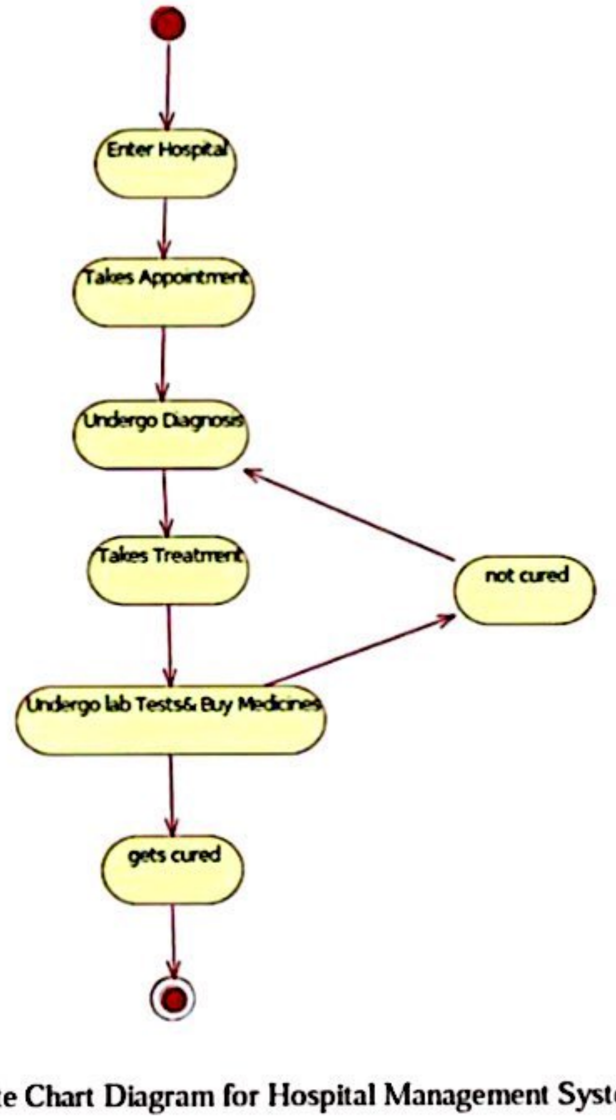
State Chart Diagram for Library Management System

9. Draw State Chart Diagram for Hospital Management System.

Any real time system is expected to be reacted by some kind of internal/external events. These events are responsible for state change of the system. State chart diagram is used to represent the event driven state change of a system. It basically describes the state change of a class, interface etc. State chart diagram is used to visualize the reaction of a system by internal/external factors.

HMS was introduced to solve the complications coming from managing all the paper works of every patient associated with the various departments of hospitalization with confidentiality. HMS provides the ability to manage all the paperwork in one place, reducing the work of staff in arranging and analyzing the paperwork of the patients. HMS does many works like:

- Maintain the medical records of the patient
- Maintain the contact details of the patient
- Keep track of the appointment dates
- Save the insurance information for later reference
- Tracking the bill payments.

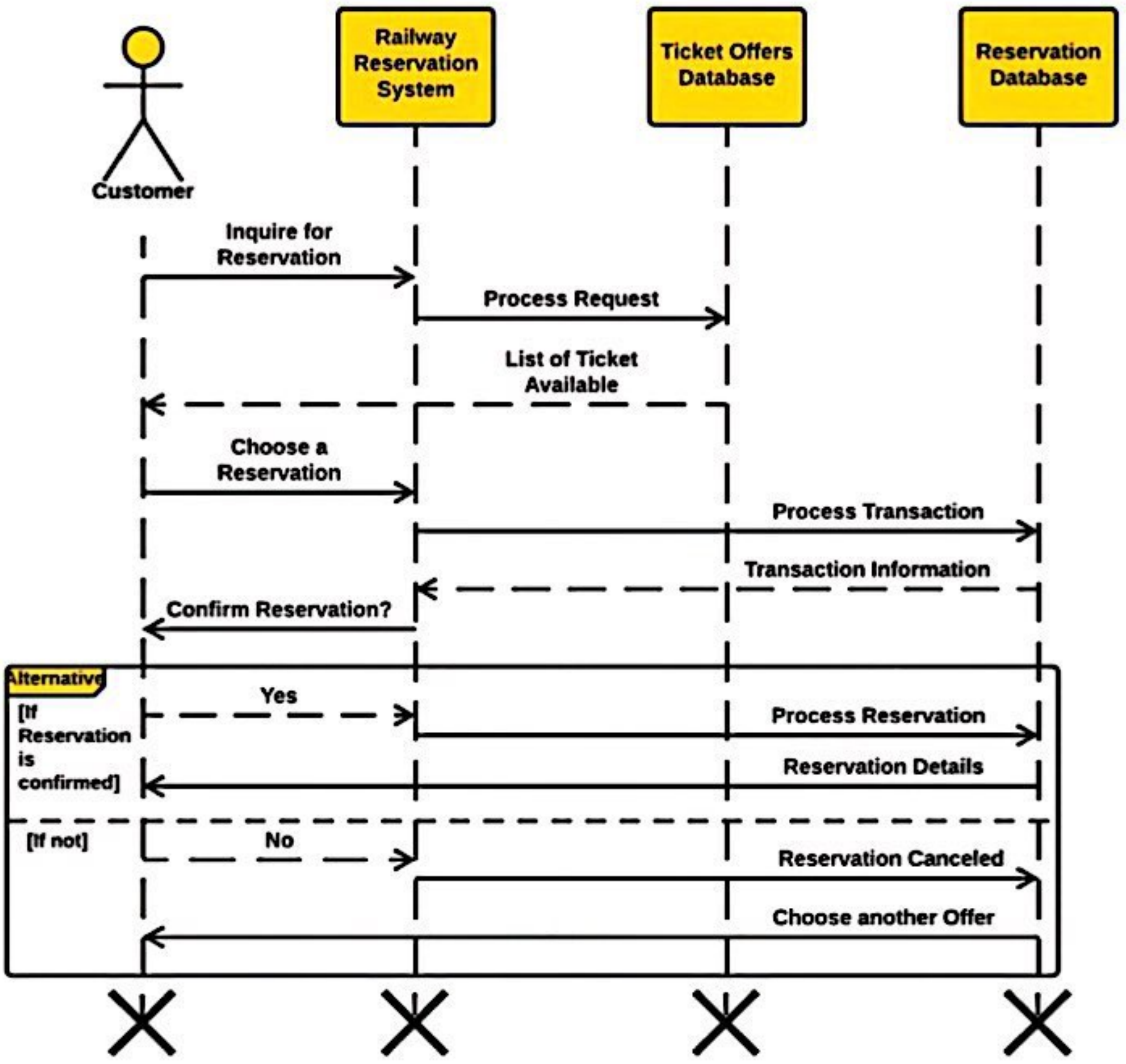


State Chart Diagram for Hospital Management System

10. Draw Interaction Diagrams for Railway Reservation System.

This design will enlighten you on how should the system or the actor approaches each other. This will also each you on how would you develop the system to achieve its desired behaviour. The design that I will be showing you is a detailed illustration of the sequence of events happening in Railway Reservation System. This designed sequence diagram is able to show programmers and readers about the sequence of messages between the actor and the objects.

As you can see through the illustration, the conditions and interactions are emphasized. These interactions are essential for the Railway Reservation System development. The series of messages are shown and labelled to guide you in building the Railway Reservation System. You can modify the design if you have more ideas. You can also add more features to this design and use it as your project blueprint. To understand and educate on how the Railway Reservation Management System works by creating a sequence diagram. Because it determines the needed objects, actors and messages or interactions.



Interaction Diagrams for Railway Reservation System