Object Oriented Analysis and Design
With Rebus Learning & Flipped Classroom

In object oriented modeling we will view the software system as a collection of interacting objects. Related objects are grouped together into classes. So, every software system or application consists of classes and these classes collaborate with each other in some manner to achieve the functionality.

Modeling is all about creating a simplification or abstraction of the end software system that is going to be developed. Modelling consists of creating different diagrams.

Every diagram can be thought of as a graph containing vertices joined with edges. In UML, these vertices are replaced with things and the edges are replaced with relationships.

To better understand the system, there are different diagrams in UML. Each diagram provides different information about the system.

Every software system contains structural (static) aspects as well as behavioral (dynamic) aspects. To model the static aspects of a software system, UML provides the following diagrams:

Class diagram

Object diagram

Component diagram

Deployment diagram

To model the dynamic aspects of a software system, UML provides the following diagrams:

- Use case diagram
- Sequence diagram
- Collaboration diagram
- Statechart diagram
- Activity diagram

Structural Diagrams

UML's four structural diagrams allow us to model the static aspects of the software system. Static aspects of the system represent the skeleton of the system. The static aspects consist of classes, interfaces, collaborations, components and nodes.

UML's four structural diagrams are organized around the major groups of things we will find when modeling a system:

1. Class diagram	Classes, interfaces, and collaborations
2. Object diagram	Objects
3. Component diagram	Components
4. Deployment diagram	Nodes

Behavioral Diagrams

UML's five behavioral diagrams allow us to model the dynamic aspects of the software system. The dynamic aspects can be thought of as the changing parts in the system.

The dynamics of a system can be seen as messages between objects or components moving from one place to another. UML's behavioral diagrams are organized around the major ways you can model the dynamics of a system:

1. Use case diagram	Organizes the behaviors of the system
2. Sequence diagram	Focused on the time ordering of messages
3. Collaboration diagram	Focused on the structural organization of objects that send and receive messages
4. Statechart diagram	Focused on the changing state of a system driven by events
5. Activity diagram	Focused on the flow of control from activity to activity

All the diagrams in UML can be summarized as shown below:

Diagram	Elements	Purpose	View
Class Diagram	Class, Interface, Collaboration and Relationships	To model the structure/skeleton of the software system	Static design view Static process view (active classes)
Object Diagram	Objects and Relationships	To model the structure of the software system	Static design view Static process view (active classes)
Component Diagram	Components and Relationships	To model the components in the software system	Implementation view
Deployment Diagram	Nodes and Relationships	To model the runtime structure of the software system	Deployment view
Use Case Diagram	Use cases, actors and relationships	To model the behavior of the system	Static use case view
Sequence Diagram	Objects and messages	To model the time ordering of messages	Dynamic view
Collaboration Diagram	Objects and messages	To model the structural organization of the objects	Dynamic view
Statechart Diagram	States, transitions, events and activities	To model the behavior as a sequence of state changes	Dynamic view
Activity Diagram	Activities, transitions and other control flow symbols	To model the behavior as a set of activities	Dynamic view

Common Modeling Techniques

Modeling different views of a system

To model a system from different views:

1) Choose the suitable views for modeling the software system.
2) Create the UML diagrams for each view you have chosen.
3) Decide which diagrams will be reviewed and used in the documentation of the project.
4) Allow room for the diagrams that are discarded.

For example, for a simple standalone centralized application may be use case view, design views are enough. But, for a complex distributed application we need all the views.

Modeling complex views

To model complex views,

1) First, hide the unnecessary details using the notations of UML.
2) Still if the diagrams are complex, consider grouping the related elements using packages.
3) Still, if the diagrams are complex, consider using notes and different colors for the elements to draw the attention of the readers.

LIBRARY MANAGEMENT SYSTEM:DESIGN AND IMPLEMENTATION

A library database needs to store information pertaining to its users (or customers), its workers, the physical locations of its branches, and the media stored in those locations. We have decided to limit the media to two types: books and videos.

The library must keep track of the status of each media item: its location, status, descriptive attributes, and cost for losses and late returns. Books will be identified by their ISBN, and movies by their title and year. In order to allow multiple copies of the same book or video, each media item will have a unique ID number.

Customers will provide their name, address, phone number, and date of birth when signing up for a library card. They will then be assigned a unique user name and ID number, plus a temporary password that will have to be changed. Checkout operations will require a library card, as will requests to put media on hold. Each library card will have its own fines, but active fines on any of a customer's cards will prevent the customer from using the library's services.

The library will have branches in various physical locations. Branches will be identified by name, and each branch will have an address and a phone number associated with it. Additionally, a library branch will store media and have employees.

Employees will work at a specific branch of the library. They receive a paycheck, but they can also have library cards; therefore, the same information that is collected about customers should be collected about employees.

Functions for customers:

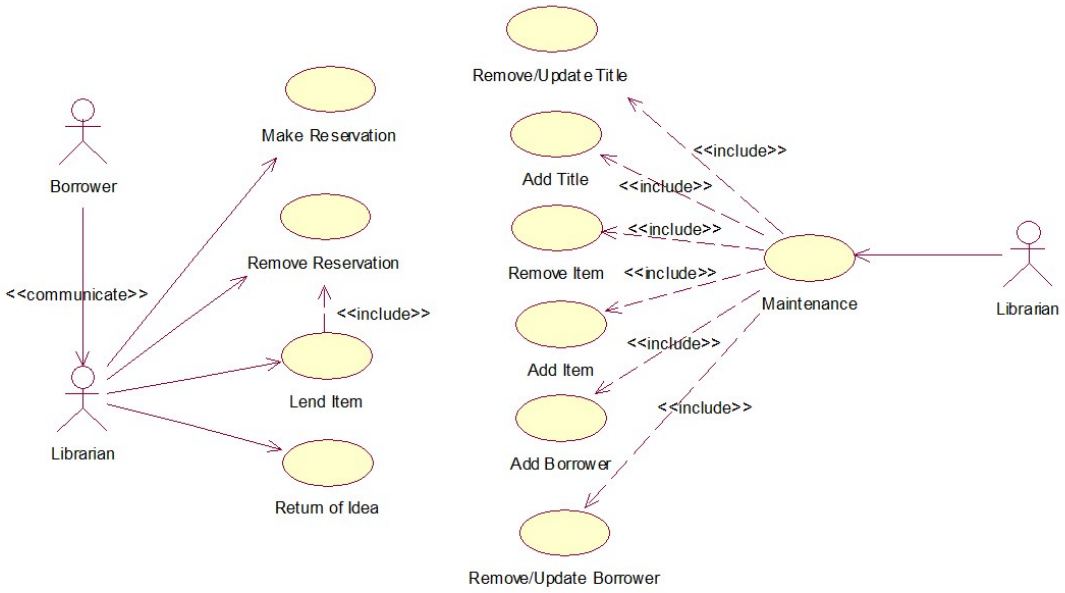
- Log in
- Search for media based on one or more of the following criteria:
    - type (book, video, or both)
    - title
    - author or director(3)
    - year
- Access their own account information:
    - Card number(s)
    - Fines
    - Media currently checked out
    - Media on hold
- Put media on hold
- Pay fines for lost or late items

- Update personal information:
    - Phone numbers
    - Addresses
    - Passwords

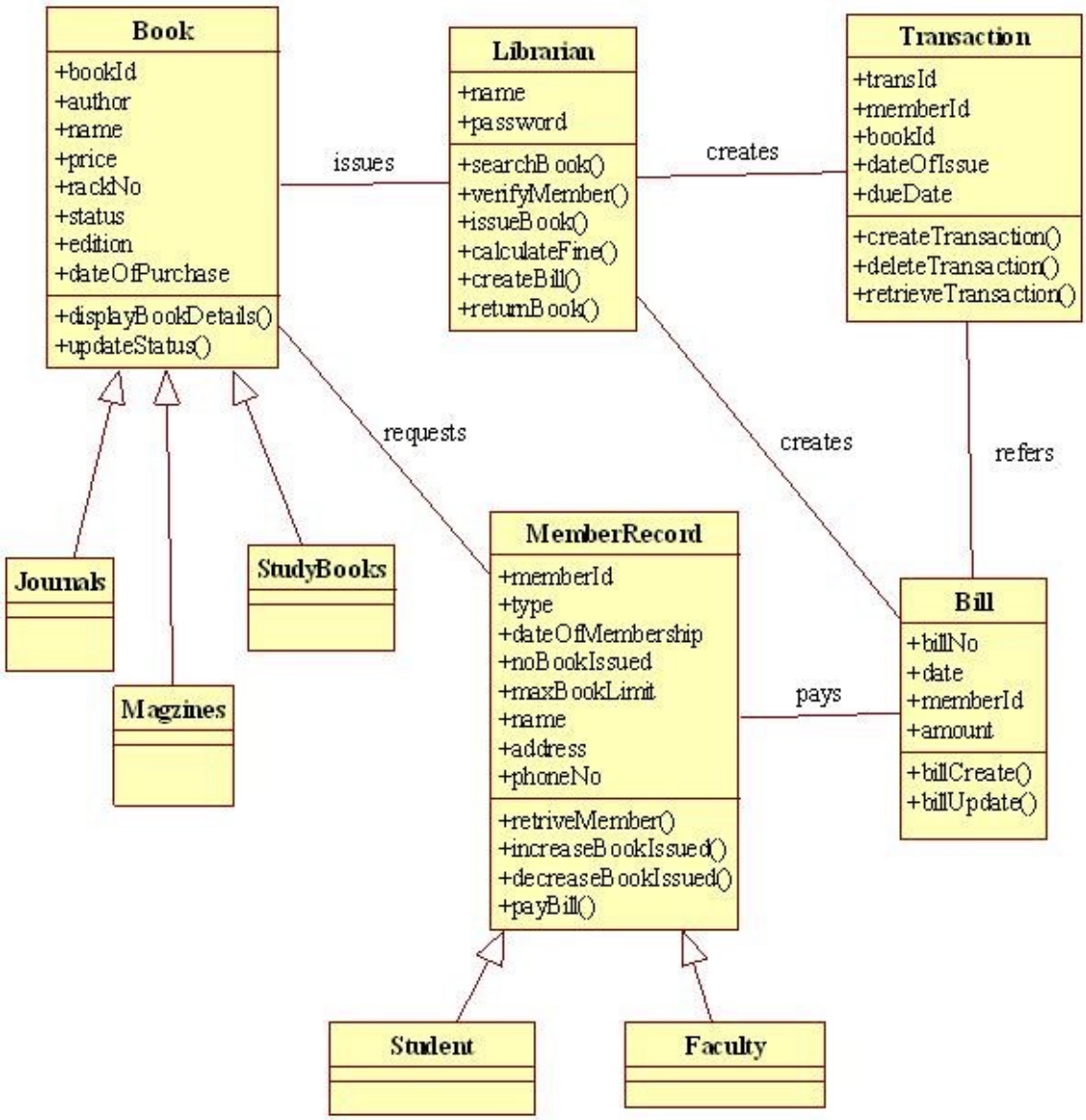
Functions for librarians are the same as the functions for customers plus the following:

- Add customers
- Add library cards and assign them to customers
- Check out media
- Manage and transfer media that is currently on hold
- Handle returns
- Modify customers' fines
- Add media to the database
- Remove media from the database
- Receive payments from customers and update the customers' fines
- View all customer information except passwords

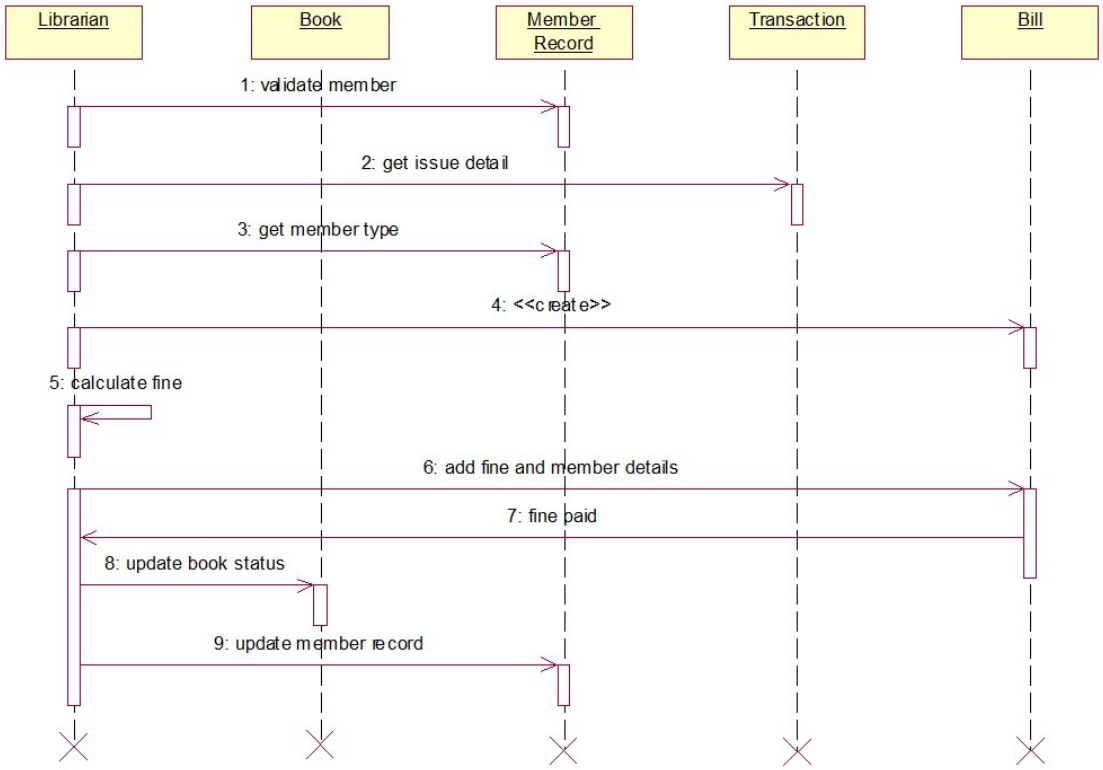
Use case diagram



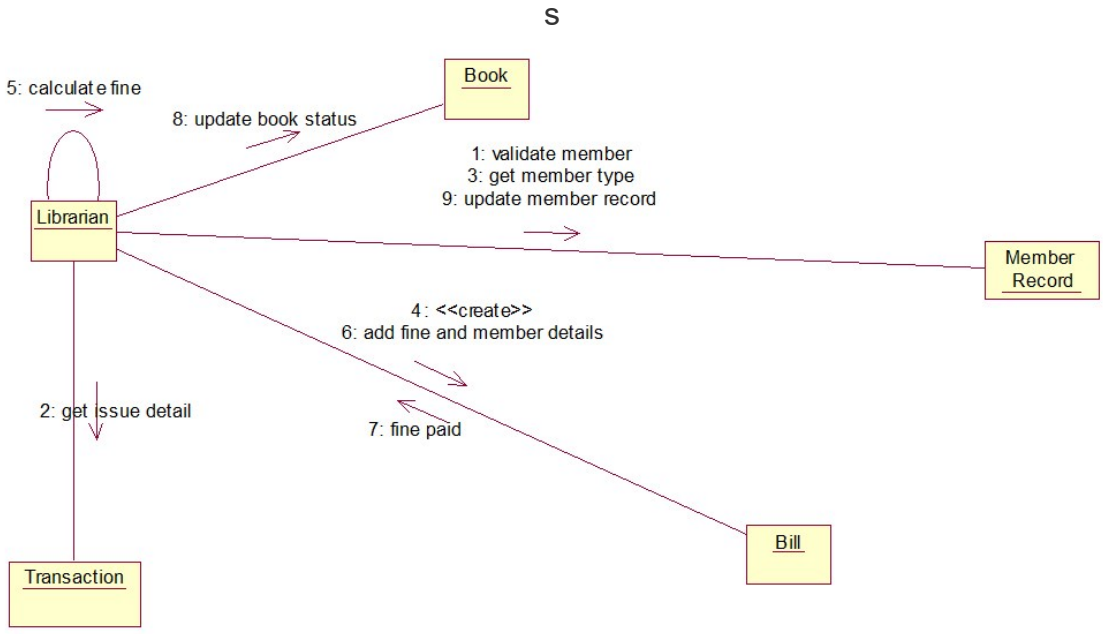
Class diagram



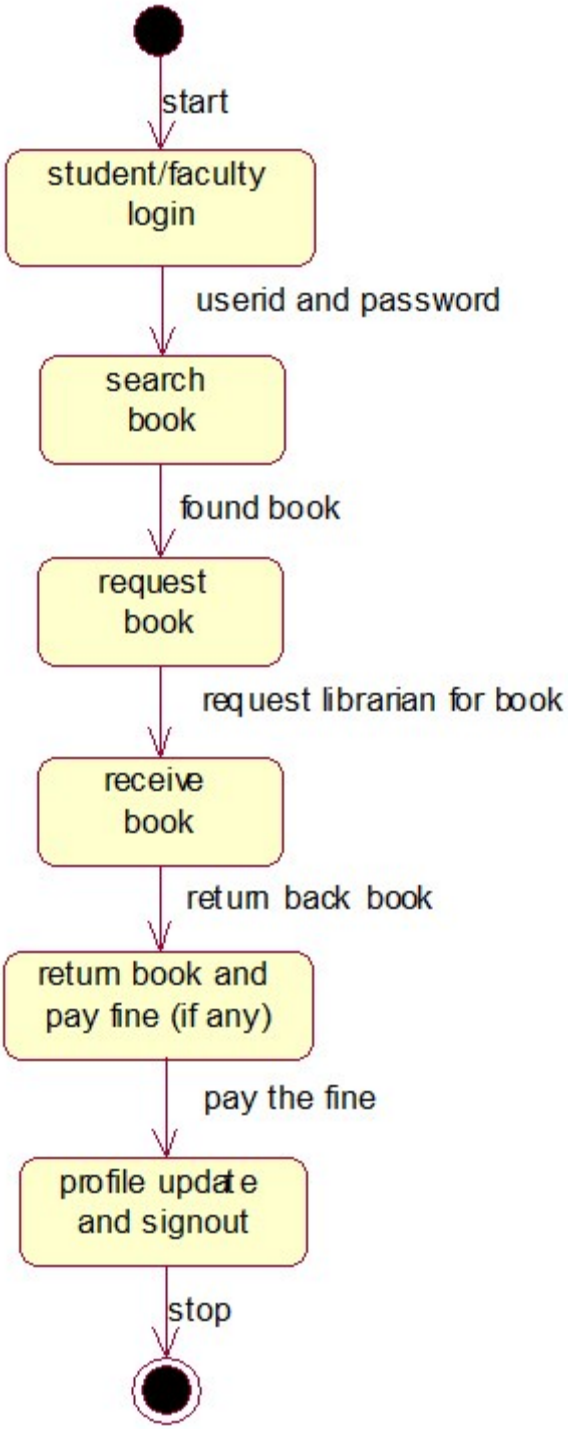
Sequence diagram



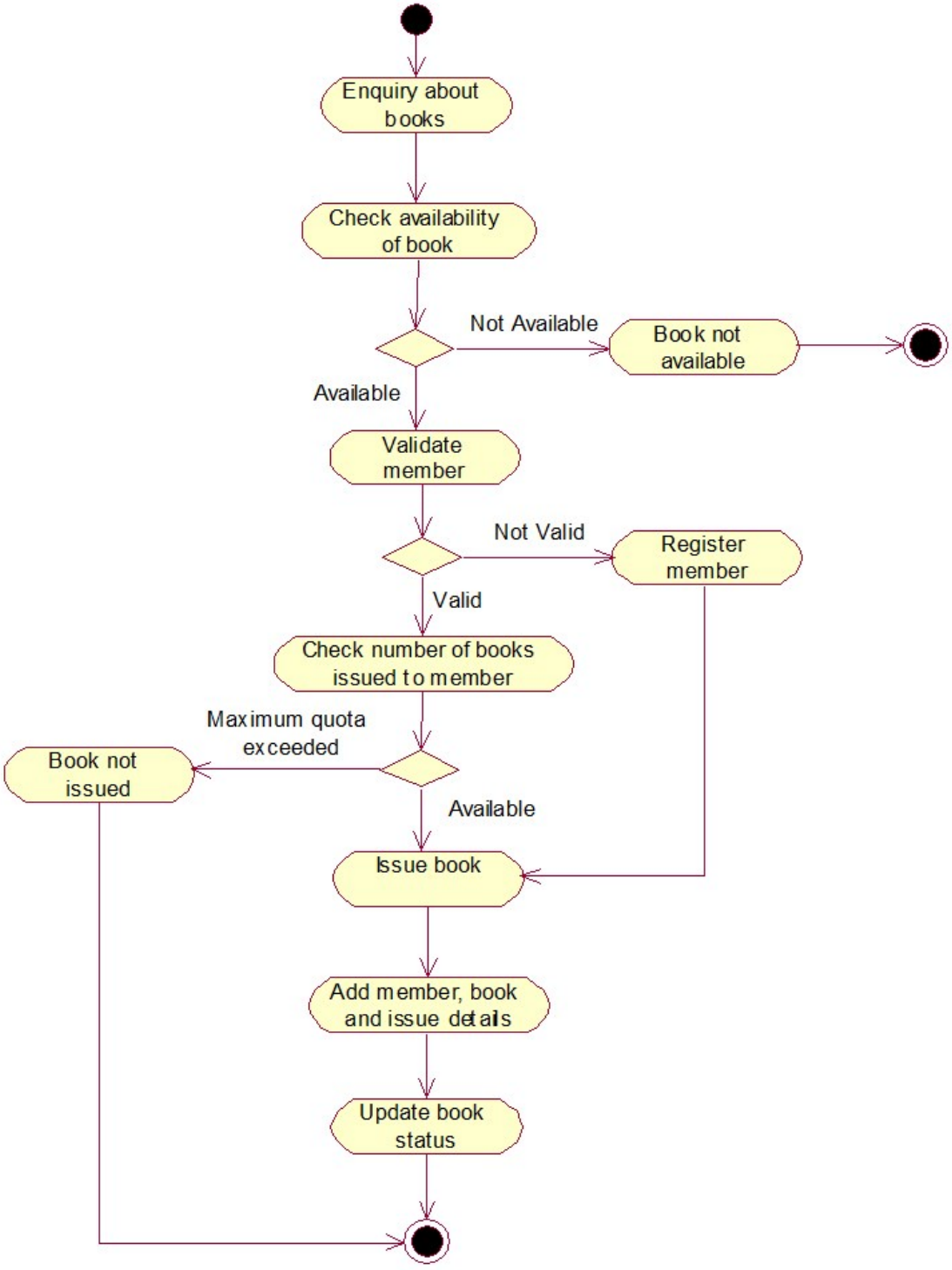
Collaboration diagram



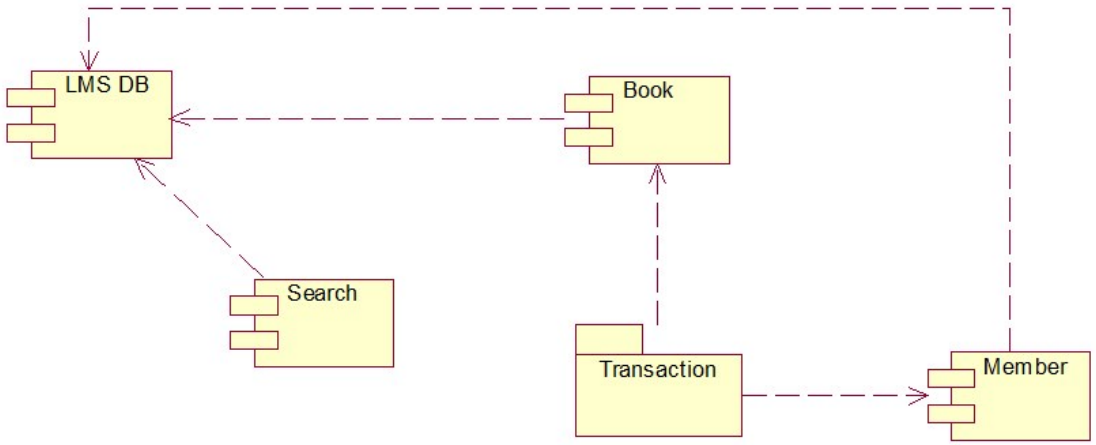
Statechart diagram



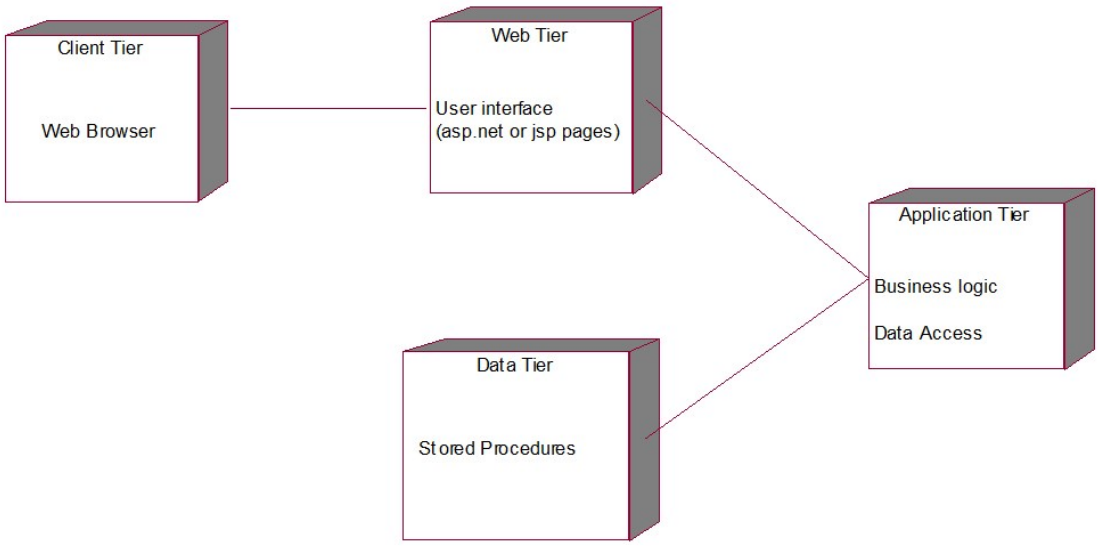
Activity diagram



Component diagram

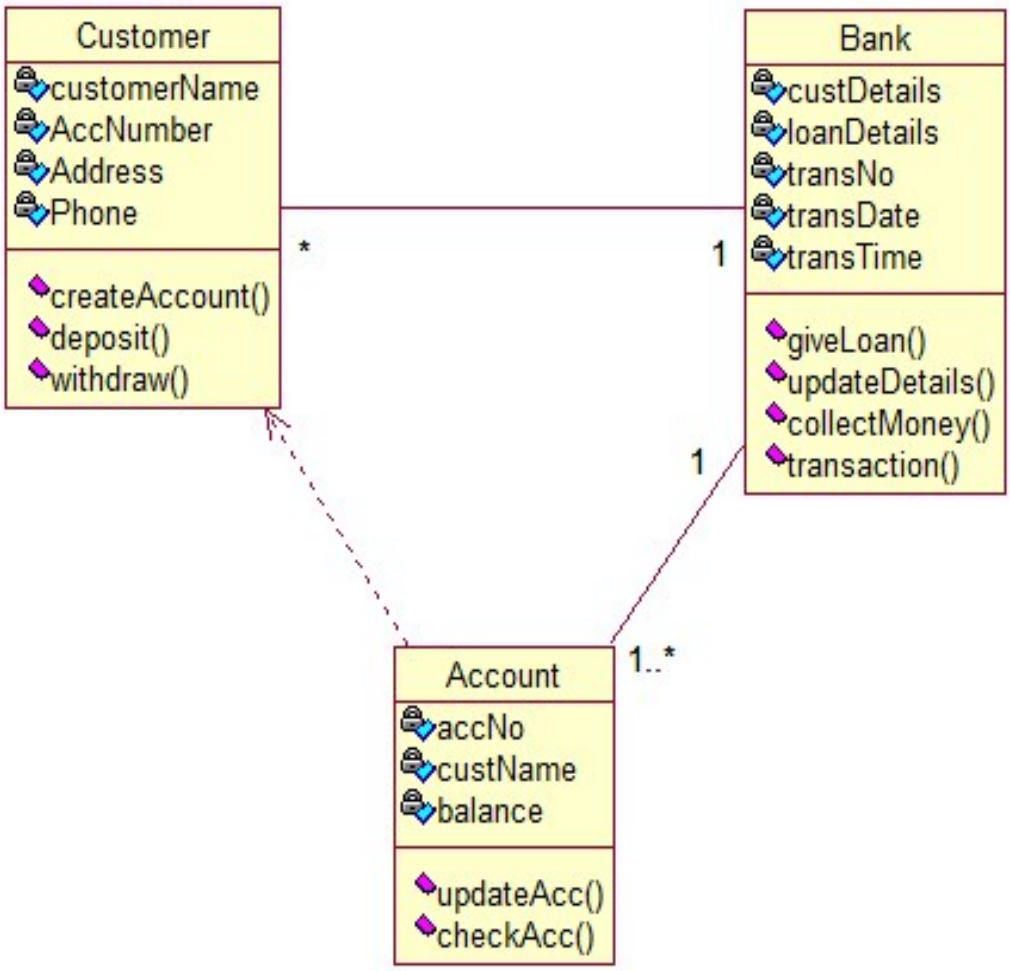


Deployment diagram

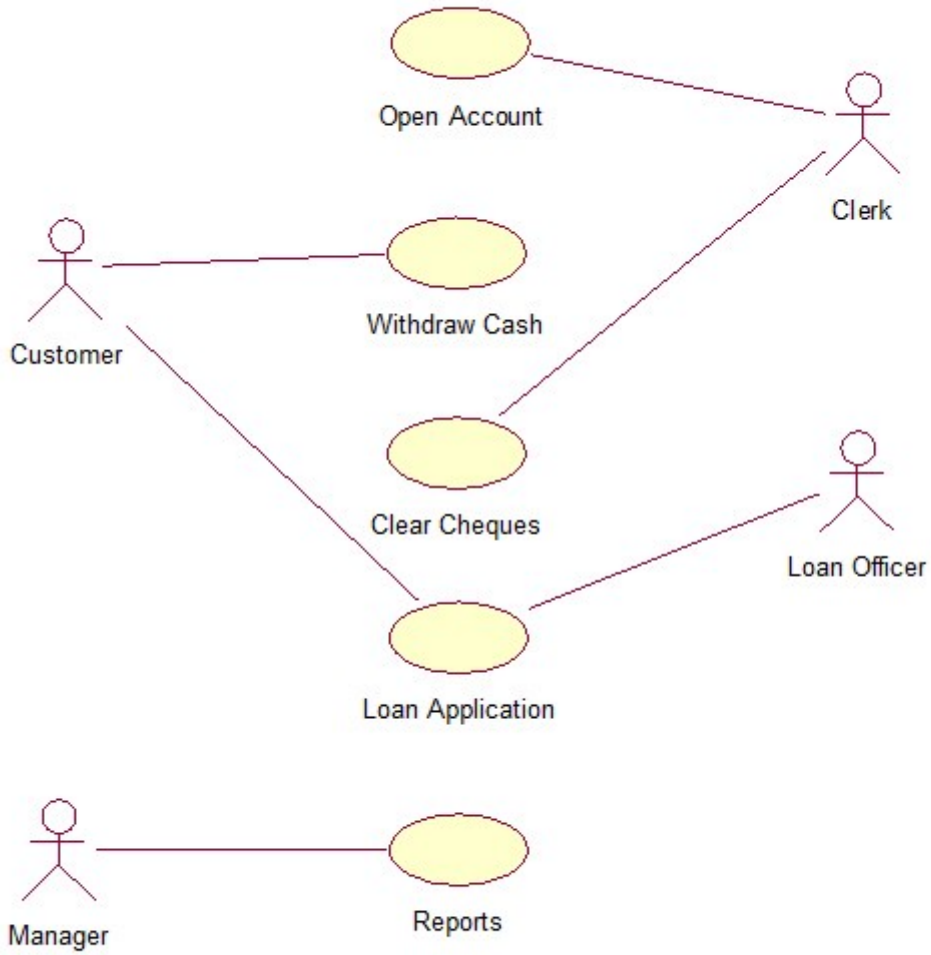


UML Diagrams for Online Banking System

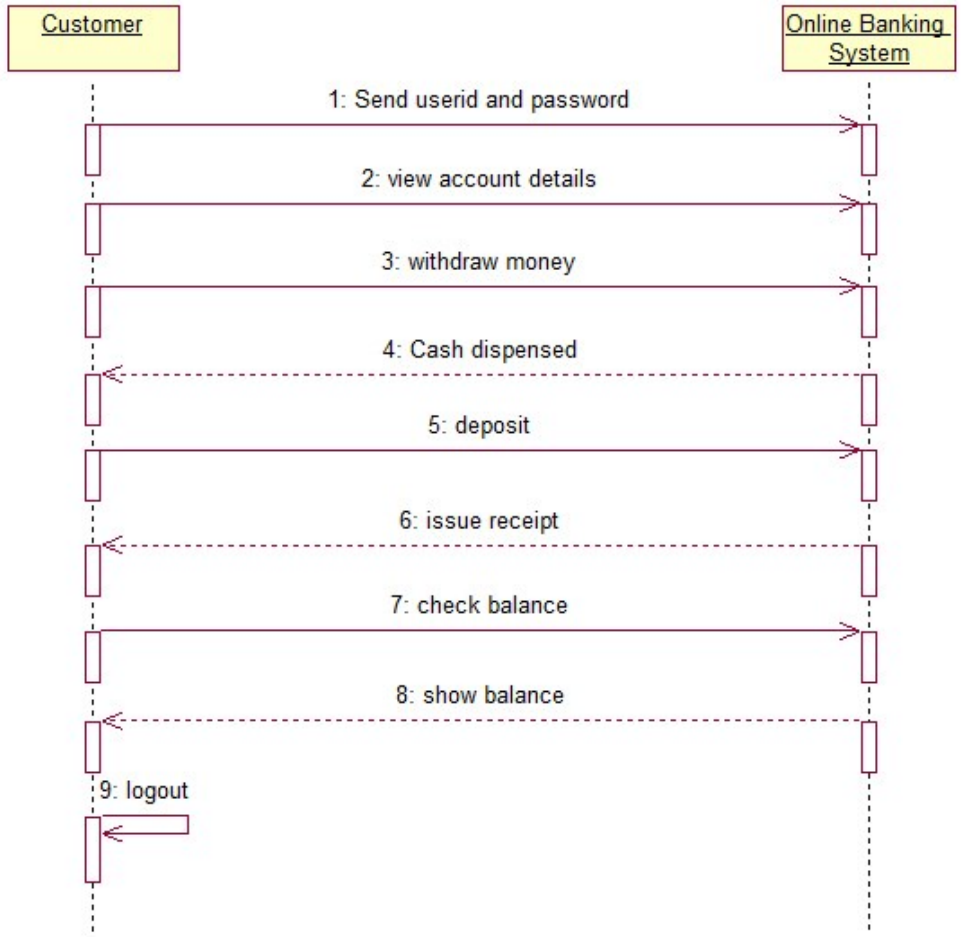
Class Diagram



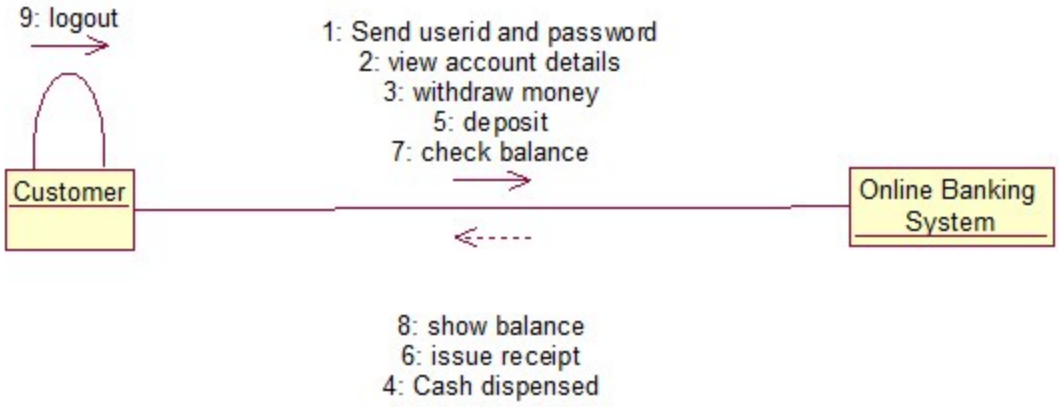
Use Case Diagram



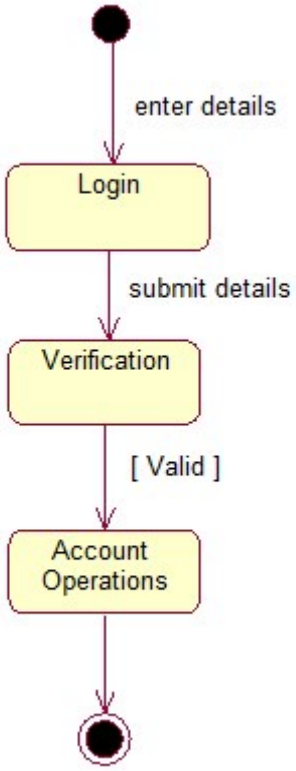
Sequence Diagram



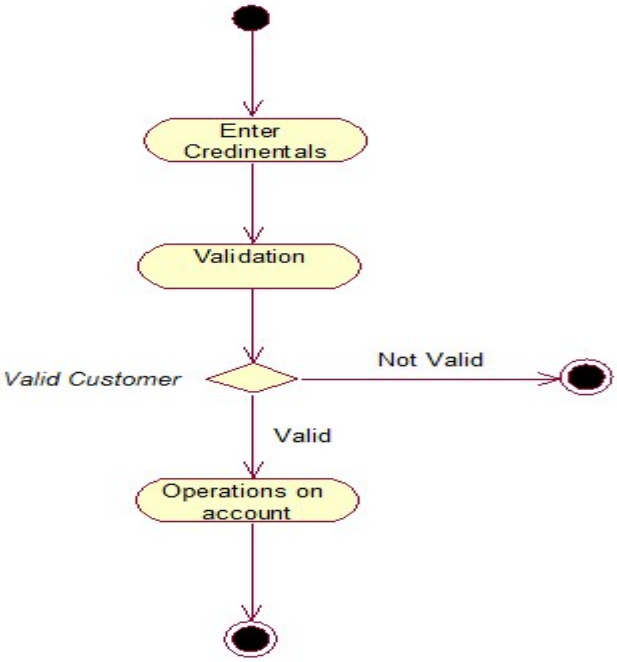
Collaboration Diagram



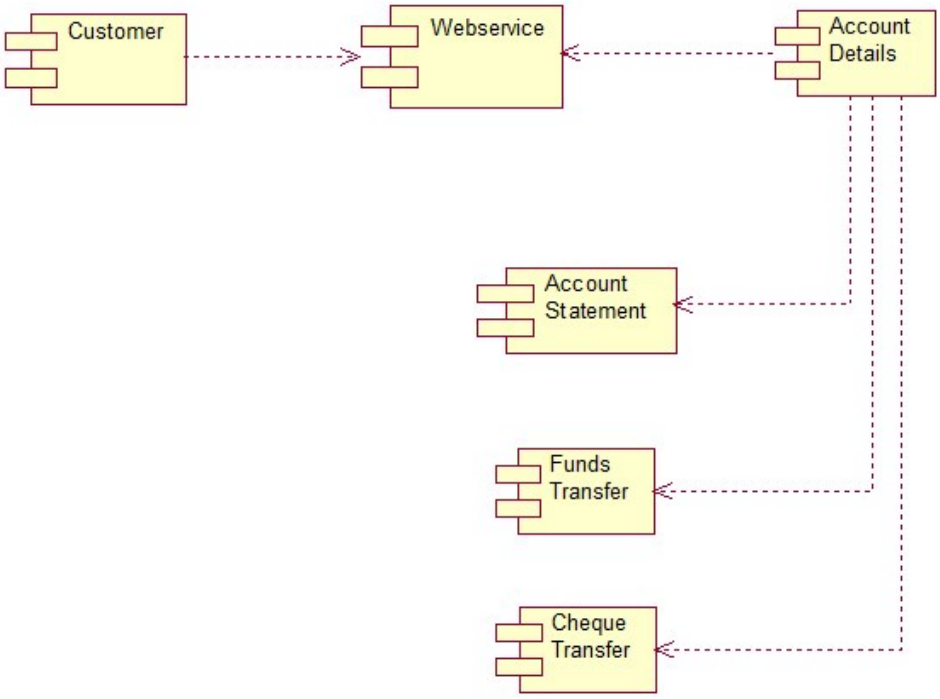
Statechart Diagram



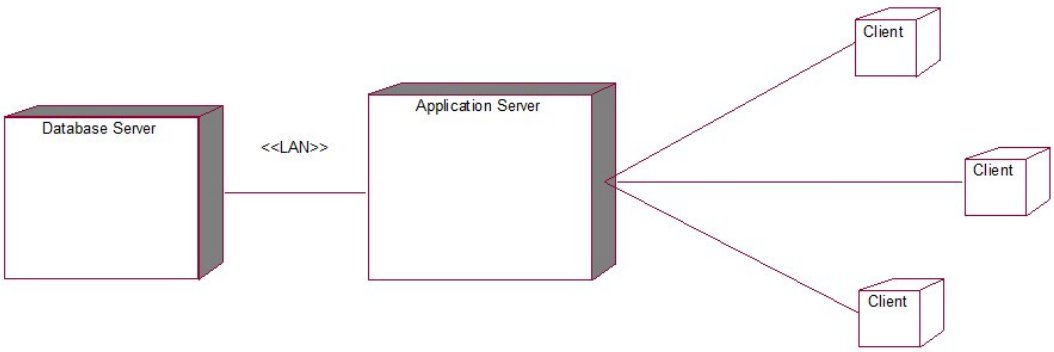
Activity Diagram



Component Diagram

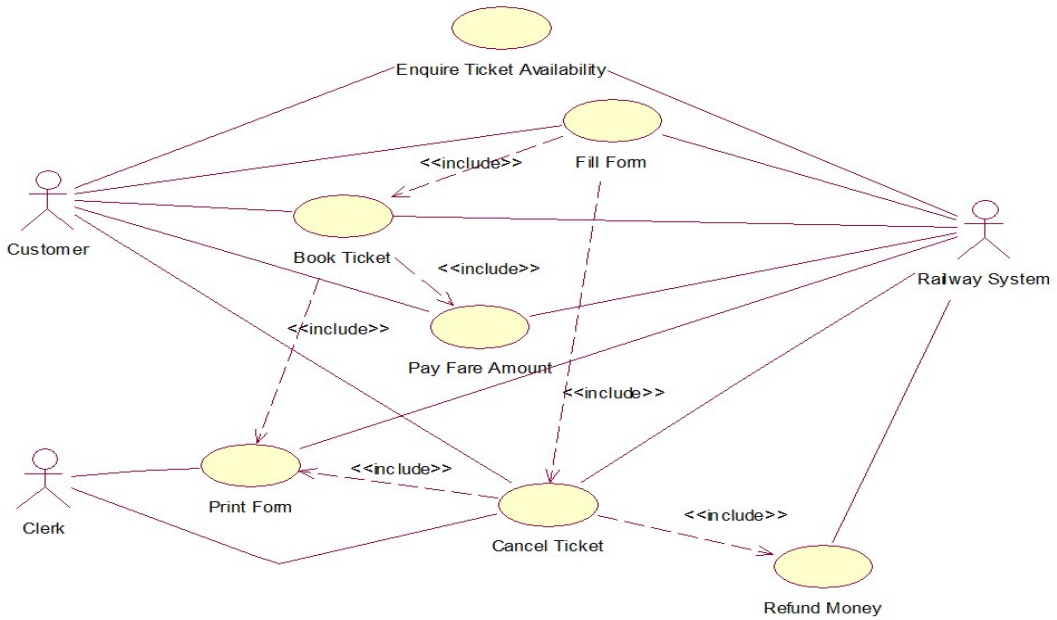


Deployment Diagram

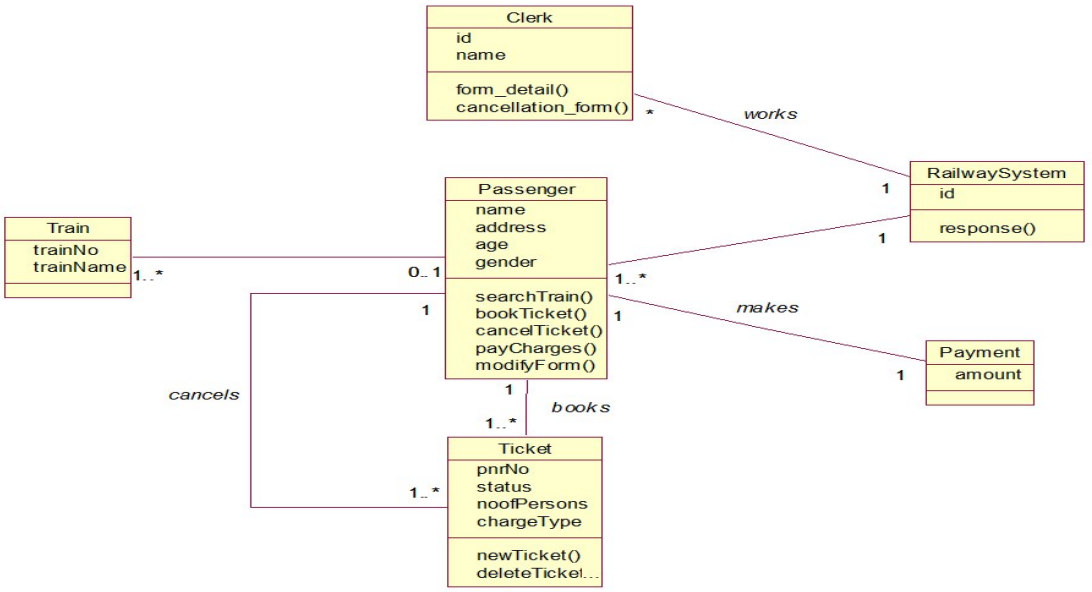


UML diagrams for railway reservation system

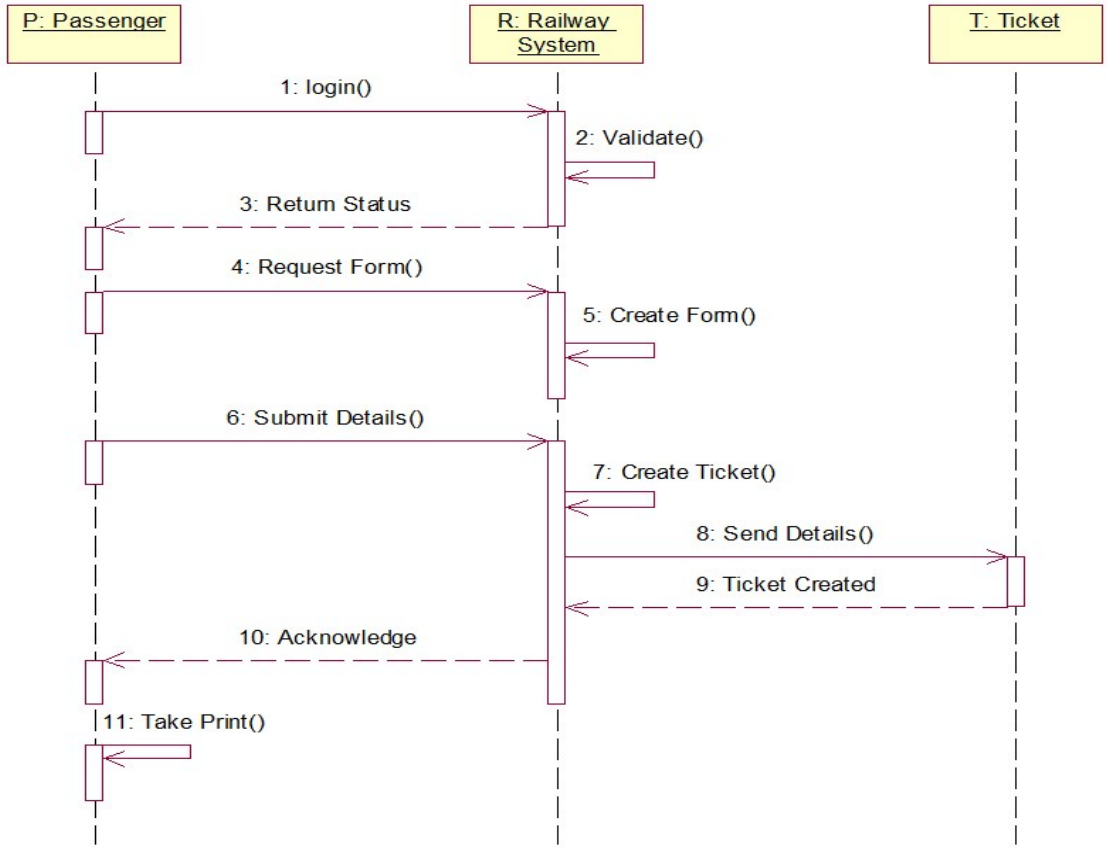
Use case diagram



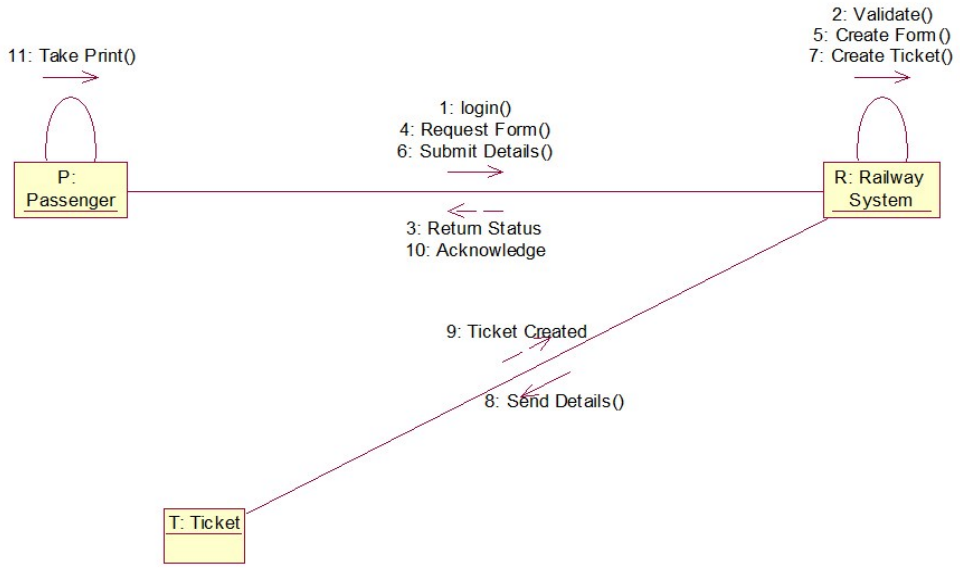
Class diagram



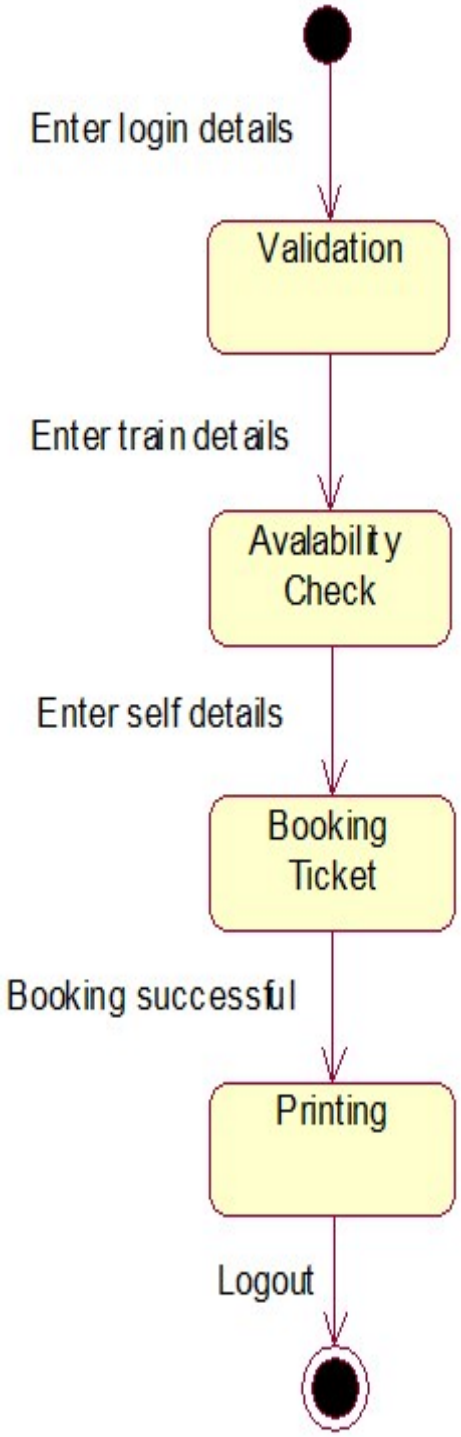
Sequence diagram



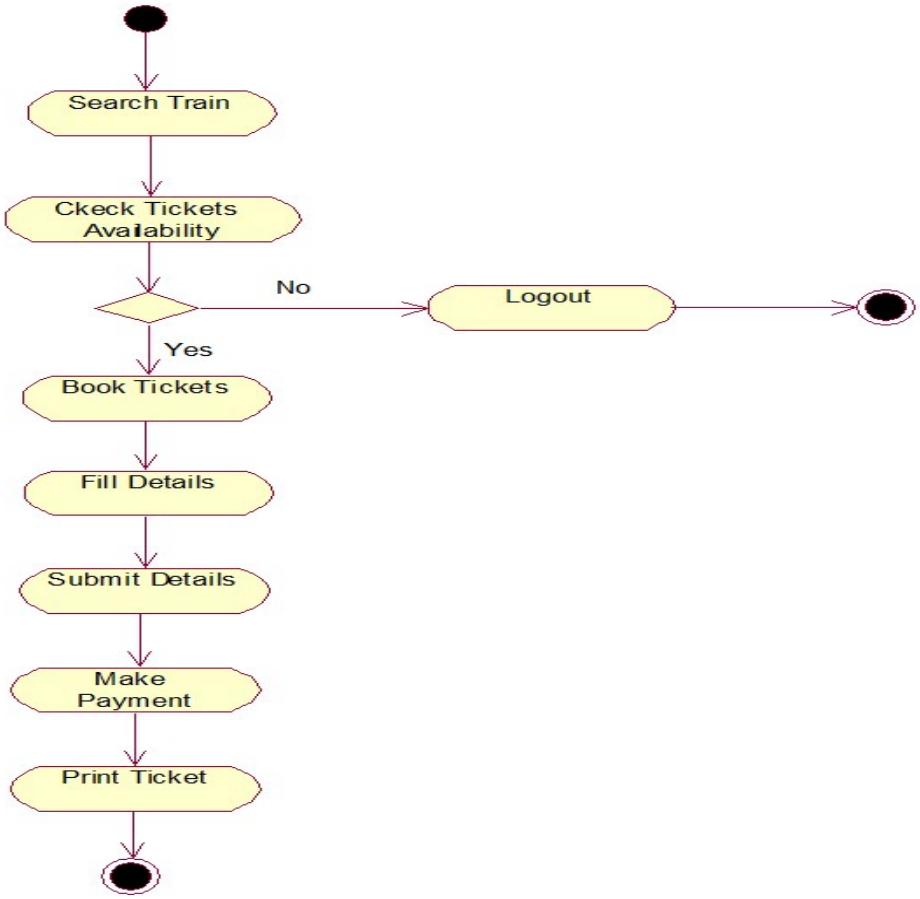
Collaboration diagram



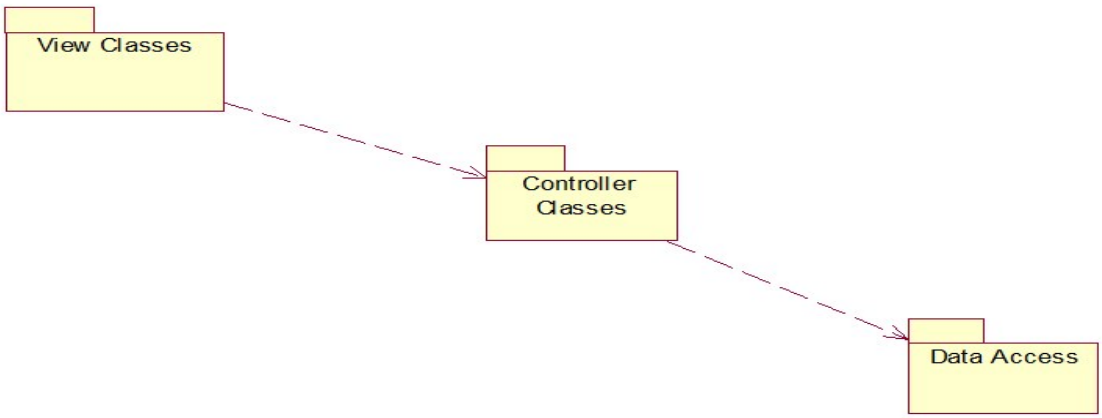
Statechart diagram



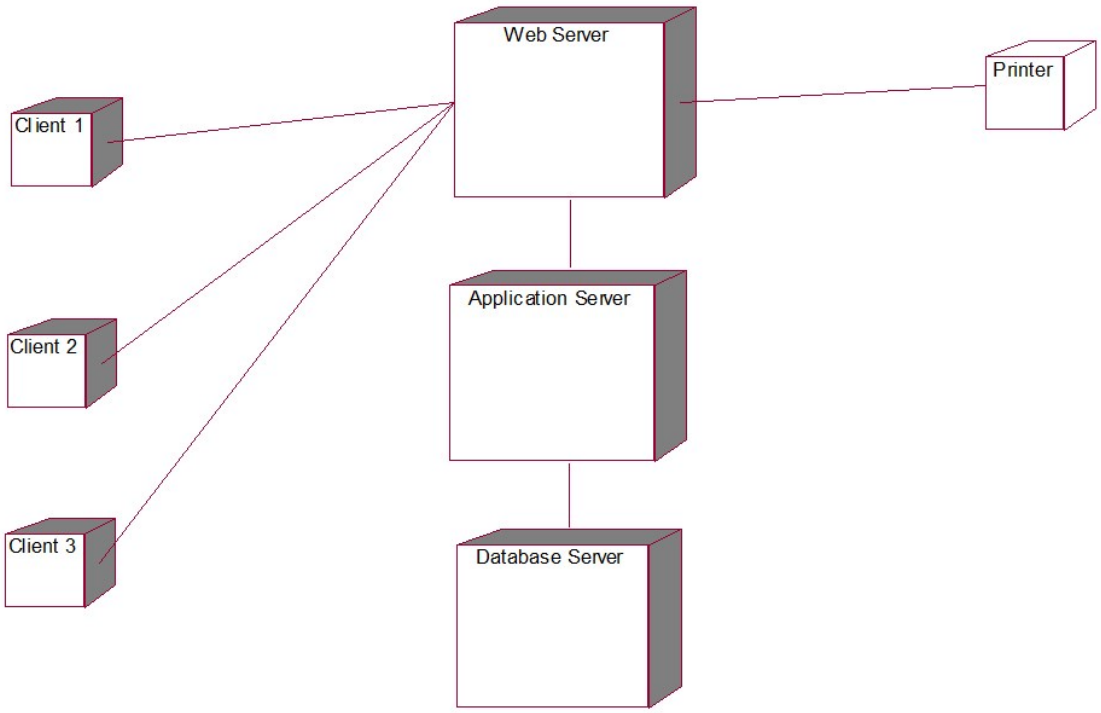
Activity diagram



Component diagrams

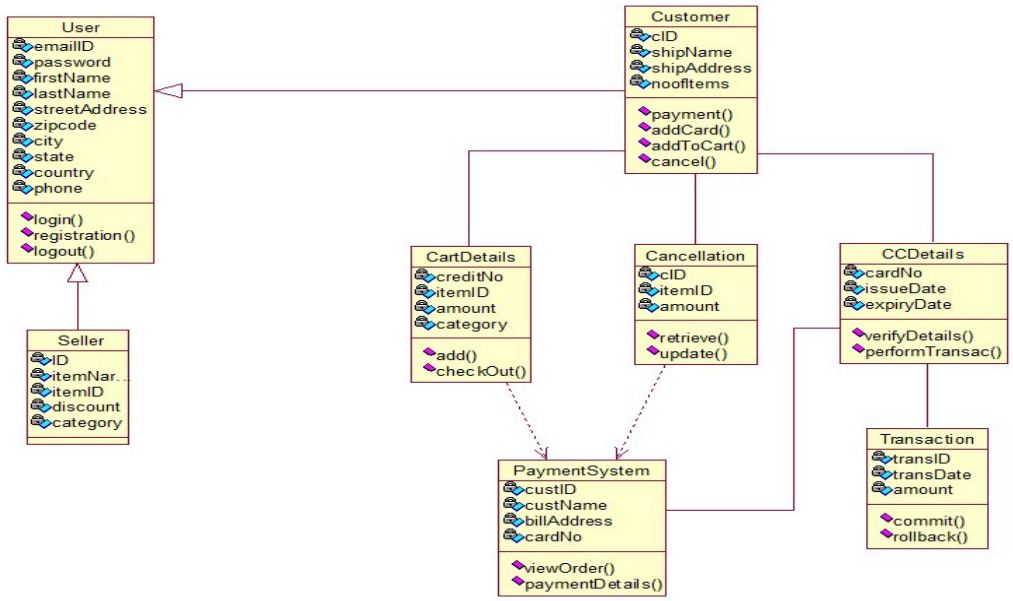


Deployment diagram

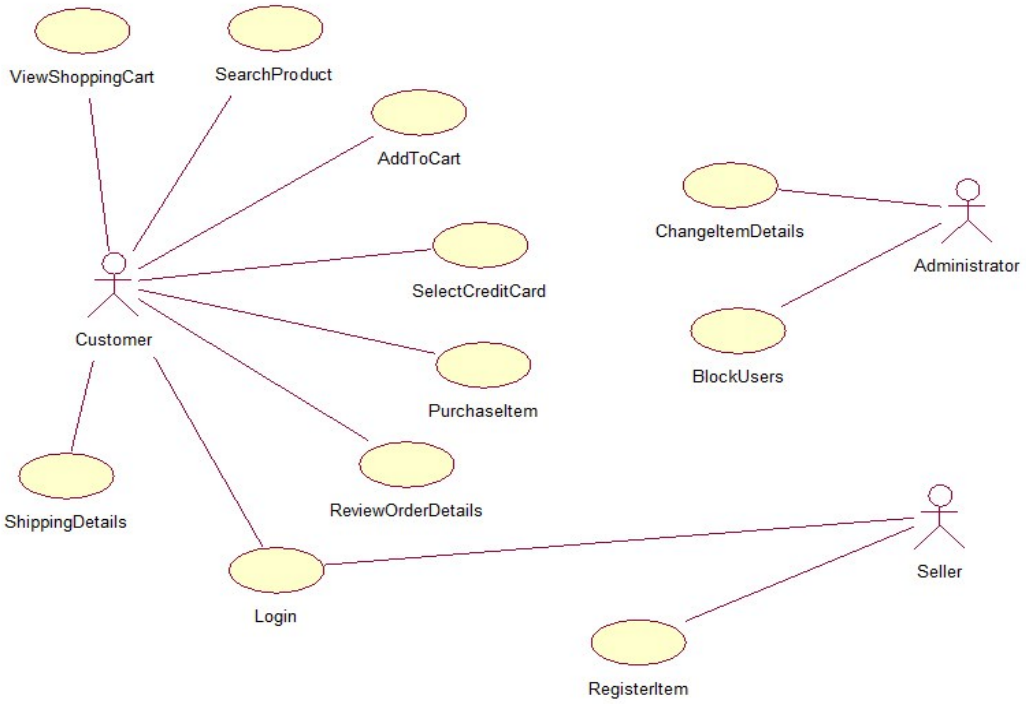


UML Diagrams for Online Book Shop

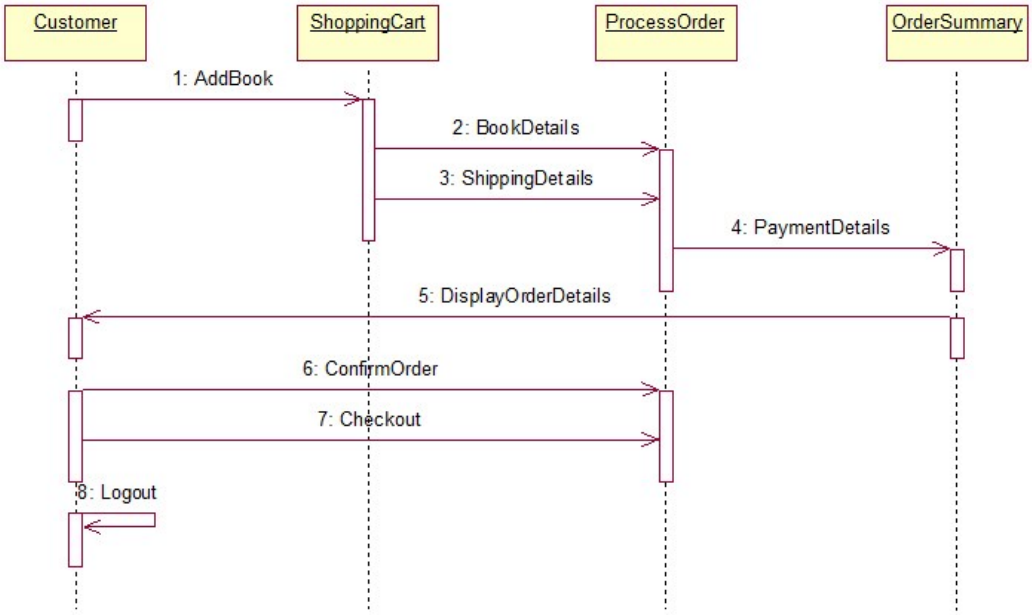
Class Diagram



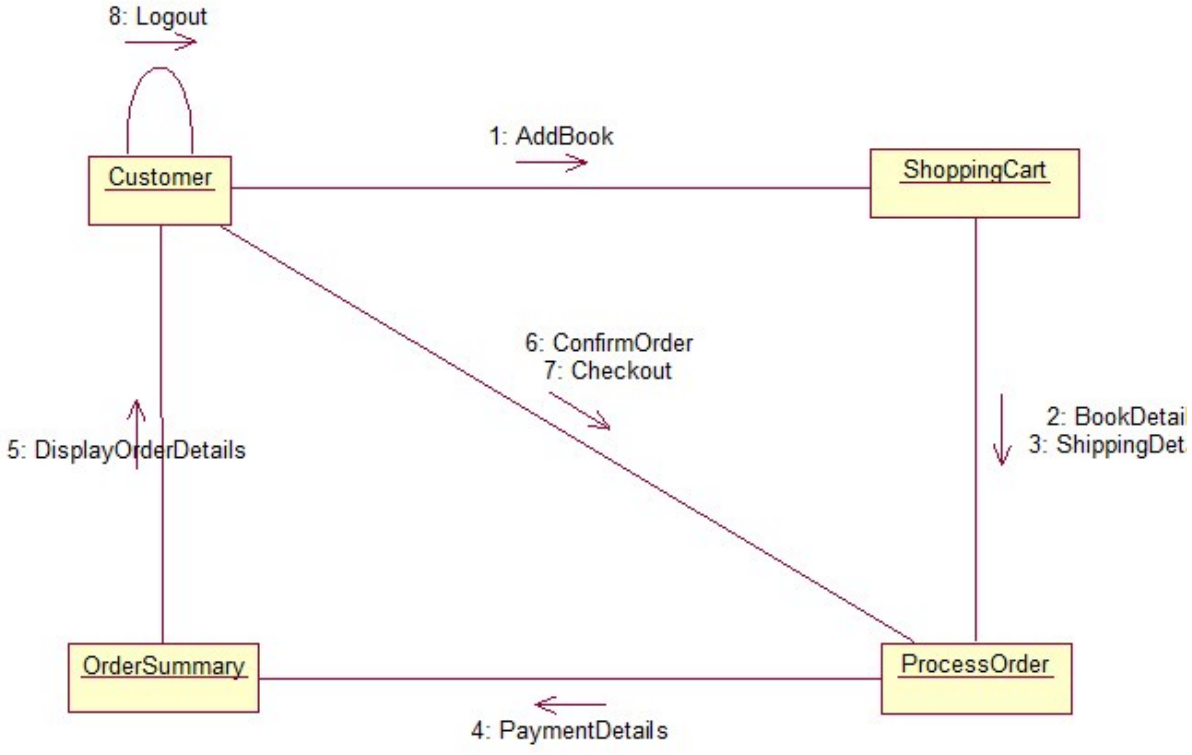
Use Case Diagram



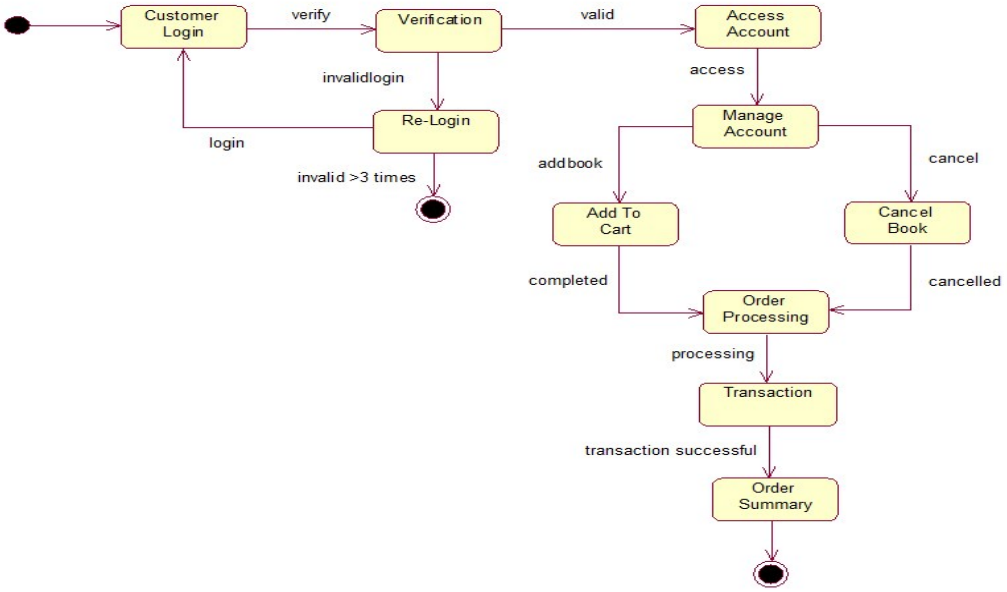
Sequence Diagram



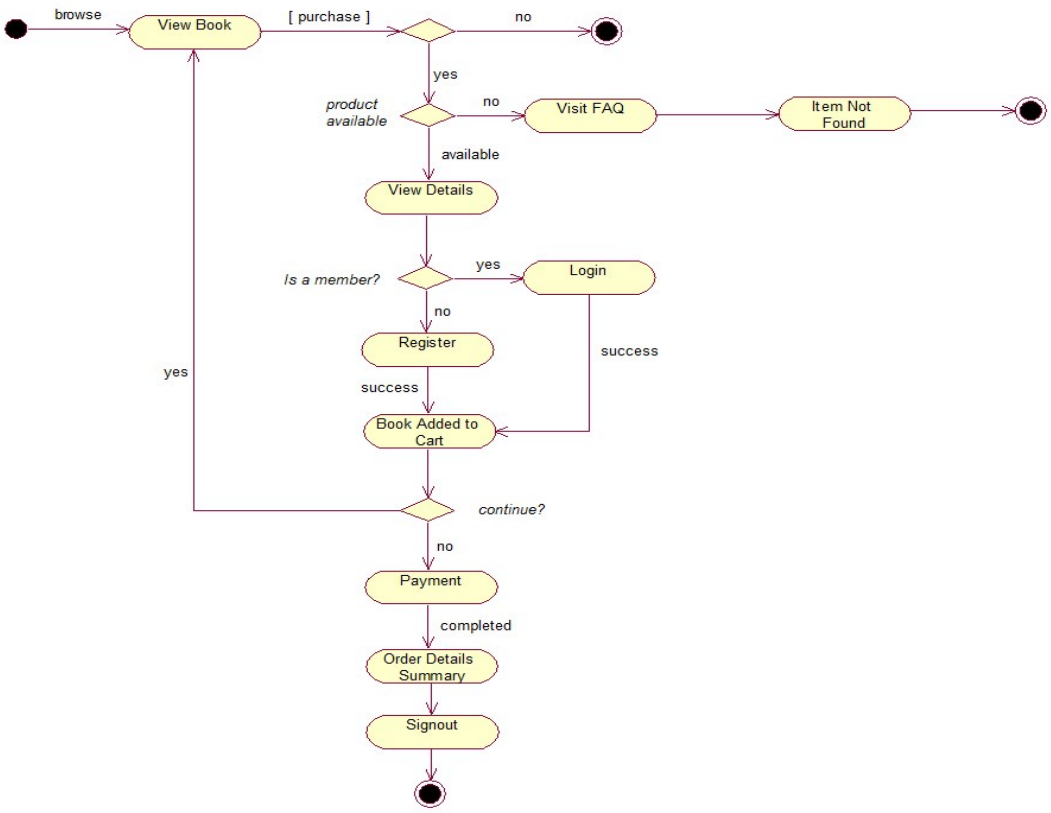
Collaboration Diagram



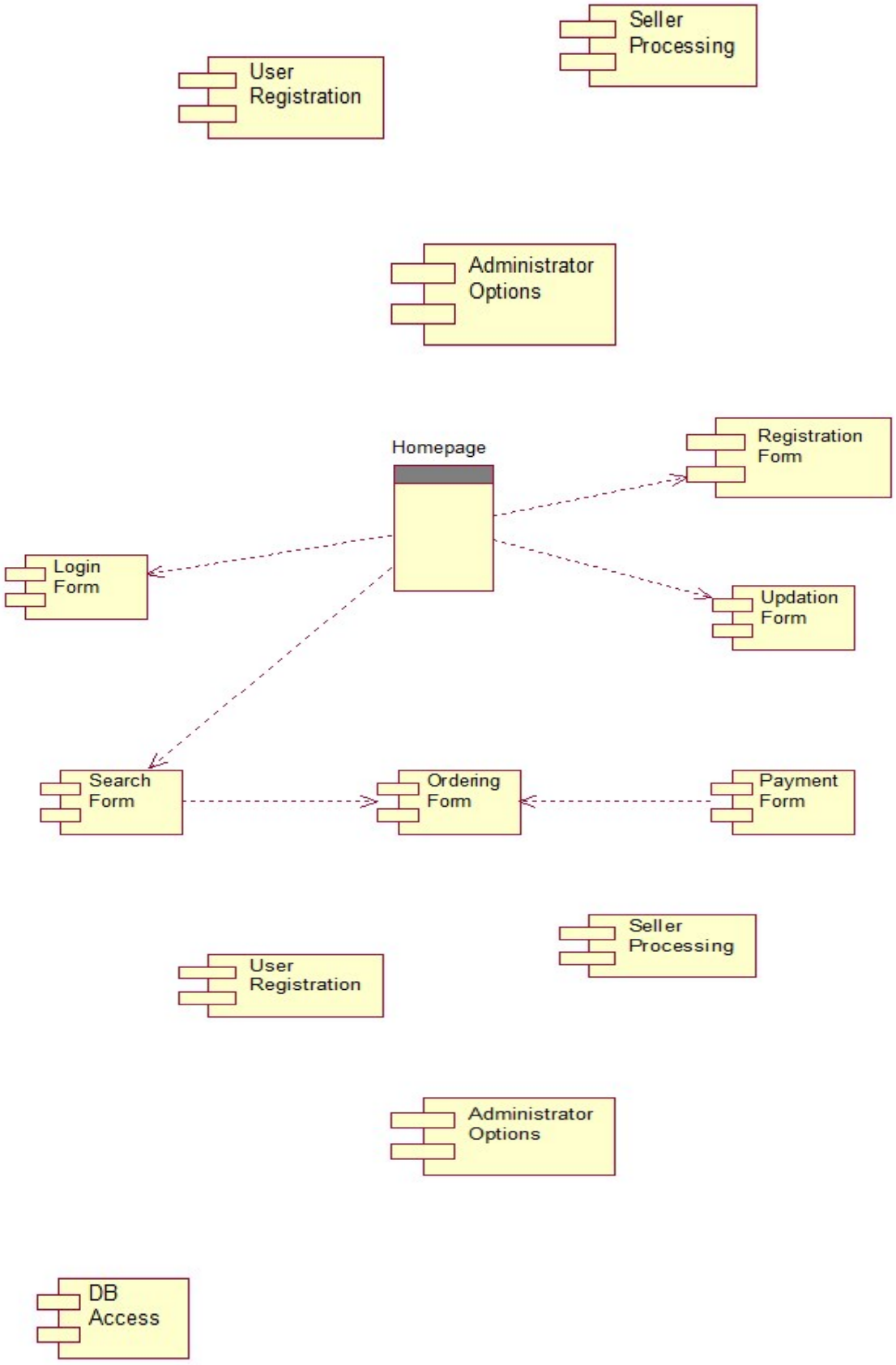
Statechart Diagram



Activity Diagram



Component Diagrams



Deployment Diagram