

Department Of CSE/IT/MCA

LAB MANUAL OF
UNIFIED MODELING LANGUAGE
(MCA V SEM)

JYOTHISHMATHI INSTITUTE OF TECHNOLOGY & SCIENCE
NUSTULAPUR, KARIMNAGAR

INDEX

Program Name	Page No.
I) Introduction to UML	3
II) Case Study : Library System	11
1.Problem Statement	11
2. Vision Document	11
3. Glossary	12
4. Supplementary Specification Document	13
5. Use-Case Model	14
6. Design Model	24
7. Deployment Model	34
III) Case Study : Restaurant System	36
1. Problem Statement	36
2. Vision Document	36
3. Glossary	37
4. Supplementary Specification Document	38
5. Use-Case Model	39
6. Design Model	45
7. Deployment Model	53
IV) Viva – Voce Questions	55

I) Introduction to UML:

The Unified Modeling Language (UML) is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing object oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects. Using the UML helps project teams communicate, explore potential designs, and validate the architectural design of the software.

Goals of UML:

1. Provide users with a ready-to-use, expressive visual modeling language so they can develop and exchange meaningful models.
2. Provide extensibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development processes.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of the OO tools market.
6. Support higher-level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

Each UML diagram is designed to let developers and customers view a software system from a different perspective and in varying degrees of abstraction. UML diagrams commonly created in visual modeling tools include:

Use Case Diagram displays the relationship among actors and use cases.

Class Diagram models class structure and contents using design elements such as classes, packages and objects. It also displays relationships such as containment, inheritance, associations and others.

Interaction Diagrams

Sequence Diagram displays the time sequence of the objects participating in the interaction. This consists of the vertical dimension (time) and horizontal dimension (different objects).

Collaboration Diagram displays an interaction organized around the objects and their links to one another. Numbers are used to show the sequence of messages.

State Diagram displays the sequences of states that an object of an interaction goes through during its life in response to received stimuli, together with its responses and actions.

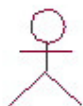
Activity Diagram displays a special state diagram where most of the states are action states and most of the transitions are triggered by completion of the actions in the source states. This diagram focuses on flows driven by internal processing.

Physical Diagrams

- **Component Diagram** displays the high level packaged structure of the code itself. Dependencies among components are shown, including source code components, binary code components, and executable components. Some components exist at compile time, at link time, at run times well as at more than one time.¹
- **Deployment Diagram** displays the configuration of run-time processing elements and the software components, processes, and objects that live on them. Software component instances represent run-time manifestations of code units.

Use Case Diagrams

A use case is a set of scenarios that describing an interaction between a user and a system. A use case diagram displays the relationship among actors and use cases. The two main components of a use case diagram are use cases and actors.



Actor



Use Case

An actor is represents a user or another system that will interact with the system you are modeling. A use case is an external view of the system that represents some action the user might perform in order to complete a task.

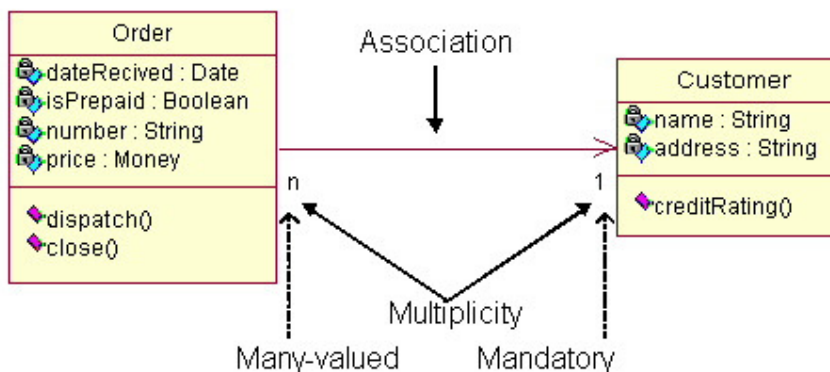
Class Diagrams

Class diagrams are widely used to describe the types of objects in a system and their relationships. Class diagrams model class structure and contents using design elements such as classes, packages and objects.² Class diagrams describe three different perspectives when designing a system, conceptual, specification, and implementation.¹ These perspectives become evident as the diagram is created and help solidify the design. This example is only meant as an introduction to the UML and class diagrams. If you would like to learn more see the [Resources](#) page for more detailed resources on UML.

Classes are composed of three things: a name, attributes, and operations. Below is an example of a class.

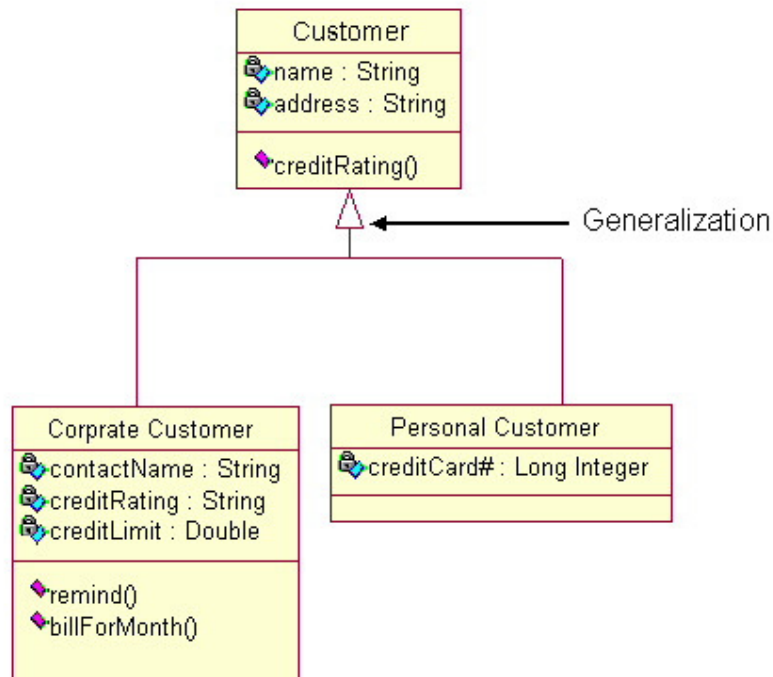


Class diagrams also display relationships such as containment, inheritance, associations and others.² Below is an example of an associative relationship:



The association relationship is the most common relationship in a class diagram. The association shows the relationship between instances of classes. For example, the class Order is associated with the class Customer. The multiplicity of the association denotes the number of objects that can participate in then relationship.¹ For example, an Order object can be associated to only one customer, but a customer can be associated to many orders.

Another common relationship in class diagrams is a generalization. A generalization is used when two classes are similar, but have some differences. Look at the generalization below:



Interaction Diagrams

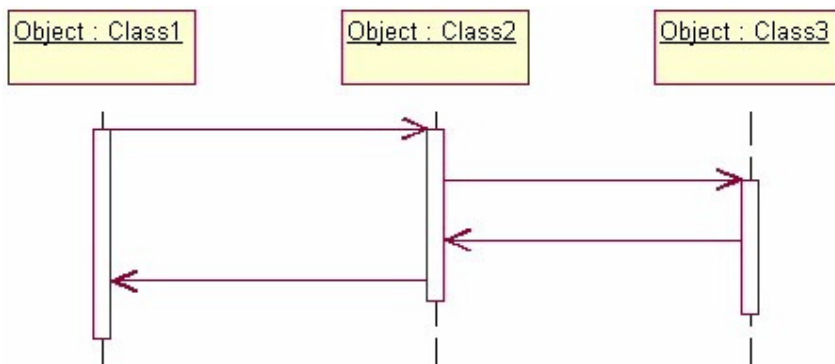
Interaction diagrams model the behavior of use cases by describing the way groups of objects interact to complete the task. The two kinds of interaction diagrams are **sequence** and **collaboration** diagrams. This example is only meant as an introduction to the UML and interaction diagrams. If you would like to learn more see the [Resources](#) page for a list of more detailed resources on UML.


Interaction diagrams are used when you want to model the behavior of several objects in a use case. They demonstrate how the objects collaborate for the behavior. Interaction diagrams do not give a in depth representation of the behavior. If you want to see what a specific object is doing for several use cases use a [state diagram](#). To see a particular behavior over many use cases or threads use an [activity diagrams](#).

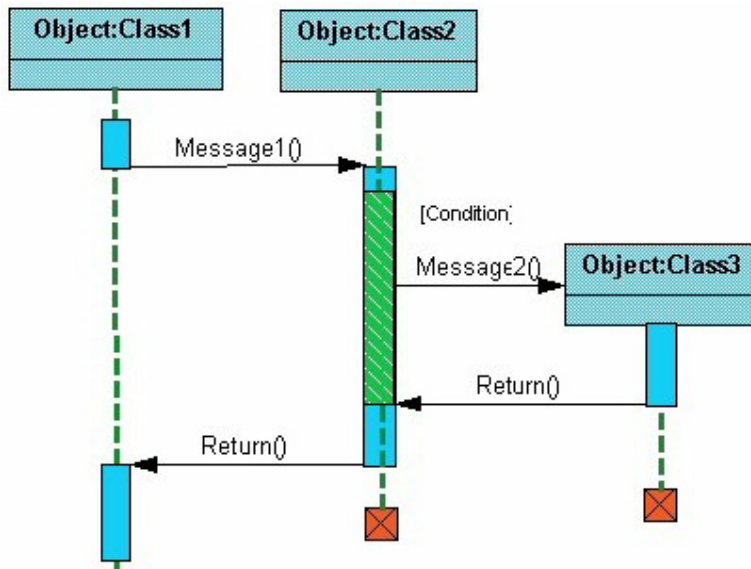
Sequence diagrams, collaboration diagrams, or both diagrams can be used to demonstrate the interaction of objects in a use case. Sequence diagrams generally show the sequence of events that occur. Collaboration diagrams demonstrate how objects are statically connected. Both diagrams are relatively simple to draw and contain similar elements.

Sequence diagrams:

Sequence diagrams demonstrate the behavior of objects in a use case by describing the objects and the messages they pass. the diagrams are read left to right and descending. The example below shows an object of class 1 start the behavior by sending a message to an object of class 2. Messages pass between the different objects until the object of class 1 receives the final message.

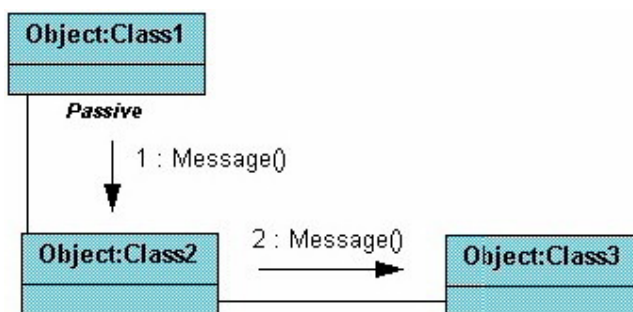


Below is a slightly more complex example. The light blue vertical rectangles the objects activation while the green vertical dashed lines represent the life of the object. The green vertical rectangles represent when a particular object has control. The  represents when the object is destroyed. This diagrams also shows conditions for messages to be sent to other object. The condition is listed between brackets next to the message. For example, a [condition] has to be met before the object of class 2 can send a message() to the object of class 3.



Collaboration diagrams:

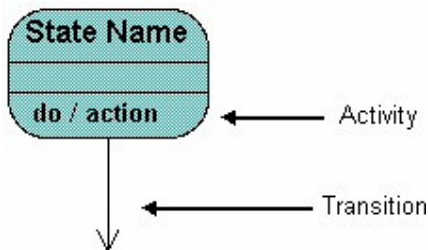
Collaboration diagrams are also relatively easy to draw. They show the relationship between objects and the order of messages passed between them. The objects are listed as icons and arrows indicate the messages being passed between them. The numbers next to the messages are called sequence numbers. As the name suggests, they show the sequence of the messages as they are passed between the objects. There are many acceptable sequence numbering schemes in UML. A simple 1, 2, 3... format can be used, as the example below shows, or for more detailed and complex diagrams a 1, 1.1 ,1.2, 1.2.1... scheme can be used.



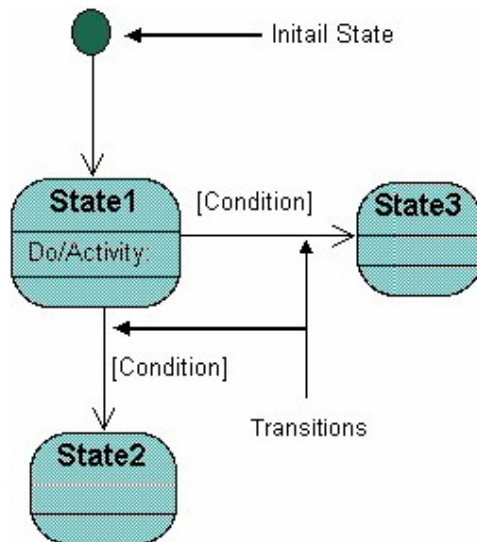
State Diagrams

State diagrams are used to describe the behavior of a system. State diagrams describe all of the possible states of an object as events occur. Each diagram usually represents objects of a single class and track the different states of its objects through the system.

State diagrams have very few elements. The basic elements are rounded boxes representing the state of the object and arrows indicating the transition to the next state. The activity section of the state symbol depicts what activities the object will be doing while it is in that state.



All state diagrams begin with an initial state of the object. This is the state of the object when it is created. After the initial state the object begins changing states. Conditions based on the activities can determine what the next state the object transitions to.

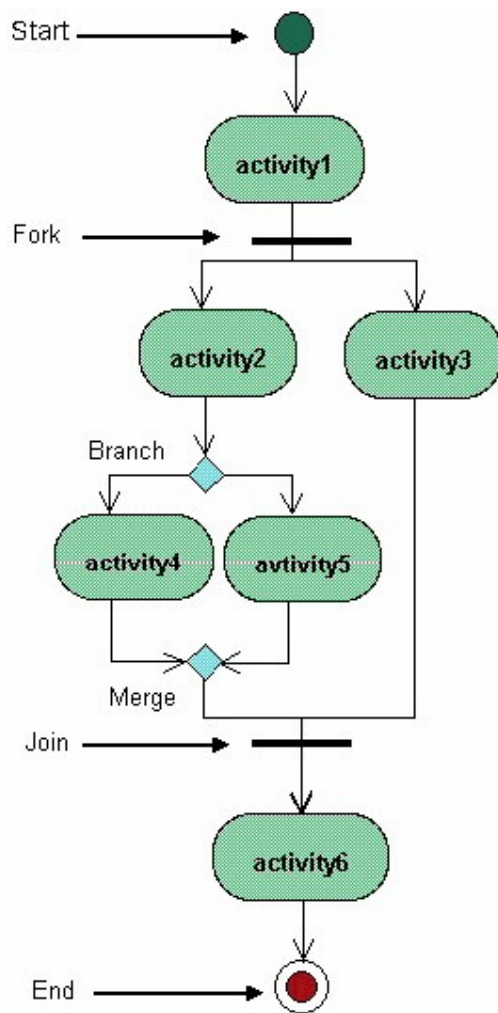


Activity Diagrams

Activity diagrams describe the workflow behavior of a system. Activity diagrams are similar to [state diagrams](#) because activities are the state of doing something. The diagrams describe the state of activities by showing the sequence of activities performed. Activity diagrams can show activities that are conditional or parallel

Activity diagrams show the flow of activities through the system. Diagrams are read from top to bottom and have branches and forks to describe conditions and parallel activities. A

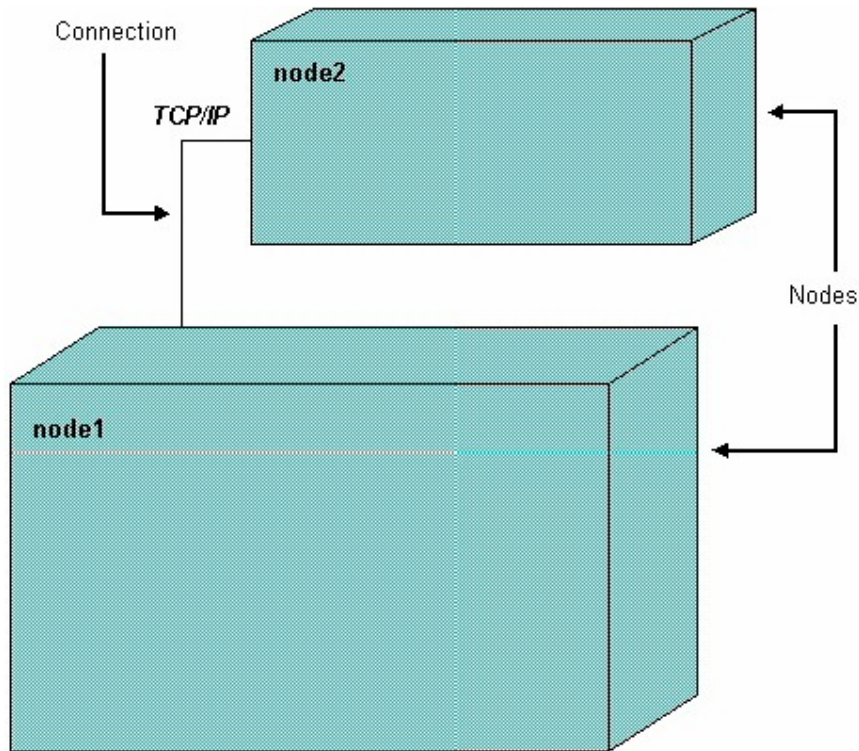
fork is used when multiple activities are occurring at the same time. The diagram below shows a fork after activity1. This indicates that both activity2 and activity3 are occurring at the same time. After activity2 there is a branch. The branch describes what activities will take place based on a set of conditions. All branches at some point are followed by a merge to indicate the end of the conditional behavior started by that branch. After the merge all of the parallel activities must be combined by a join before transitioning into the final activity state.



Physical Diagrams

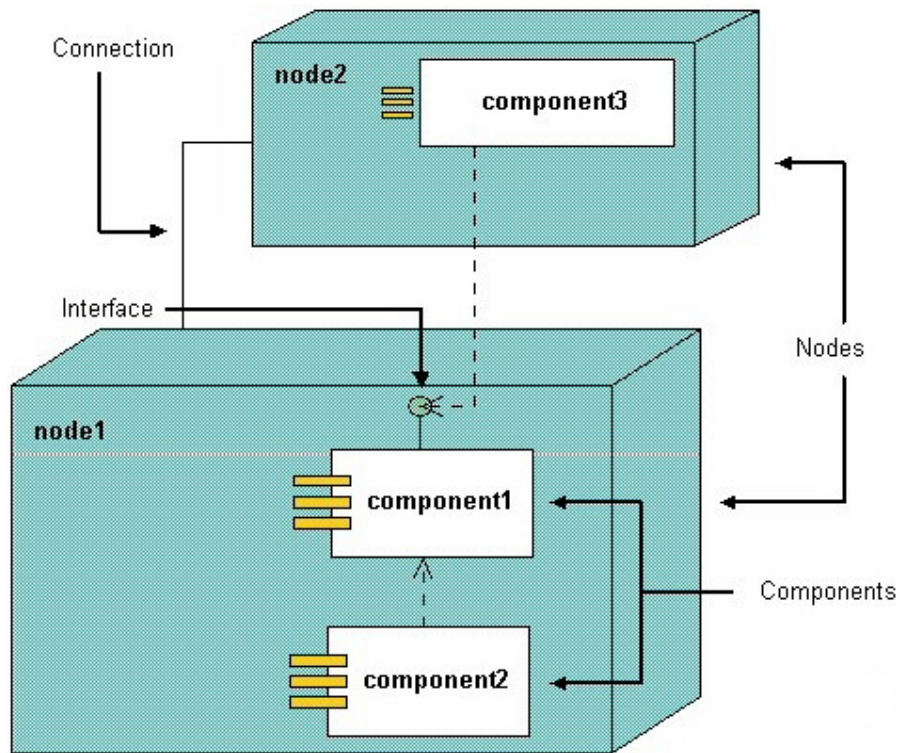
There are two types of physical diagrams: **deployment diagrams** and **component diagrams**. Deployment diagrams show the physical relationship between hardware and software in a system. Component diagrams show the software components of a system and how they are related to each other. These relationships are called dependencies.

The deployment diagram contains nodes and connections. A node usually represents a piece of hardware in the system. A connection depicts the communication path used by the hardware to communicate and usually indicates a method such as TCP/IP.



The component diagram contains components and dependencies. Components represent the physical packaging of a module of code. The dependencies between the components show how changes made to one component may affect the other components in the system. Dependencies in a component diagram are represented by a dashed line between two or more components. Component diagrams can also show the interfaces used by the components to communicate to each other.

The combined deployment and component diagram below gives a high level physical description of the completed system. The diagram shows two nodes which represent two machines communicating through TCP/IP. Component2 is dependant on component1, so changes to component 2 could affect component1. The diagram also depicts component3 interfacing with component1. This diagram gives the reader a quick overall view of the entire system.



II) Case Study: Library System

1. Problem Statement

The Library System is a web-based application used to automate a library. It allows the librarian to maintain the information about books, magazines and CDs. It also allows the librarian to maintain the information about its users. It provides the facilities such as search for items, browse, checkout items, return items, make reservation, remove reservation etc. to its users.

To borrow the items from the library, the users must register in the system. The search option allows the users to search for any item in the library. If the user finds that the required item is available in the library, he/she can checkout the item from the library. If the item is currently not available in the library, the user can make reservation for the item. When the item becomes available the respective user who made the reservation for that item first is notified.

The reservation is canceled when the user checks out the item from the library or through an explicit cancellation procedure.

The system allows the librarian to easily create, update, and delete information about titles, borrowers, items and reservations in the system. The librarian is an employee of the library who interacts with the borrowers whose work is supported by the system.

The Library System can run on popular web-browser platforms like Windows Explorer, Netscape Navigator etc. It can be easily extended with new functionality.

2. Vision Document

A vision document describes the higher level requirements of the system specifying the scope of the system.

The vision document for the Library System might be

- It is a support system
- The library lends books, magazines and CDs to borrowers who are registered in the system
- The Library System handles the purchases of new titles for the library
- Popular titles are brought in multiple copies. Old books, magazines and CDs are removed when they are out of date or in poor condition
- The librarian is an employee of the library who interacts with the borrowers whose work is supported by the system
- A borrower can reserve a book, magazine or CD that is not currently available in the library so that when it is returned or purchased by the library, the borrower is notified
- The reservation is canceled when the borrower checks out the book, magazine or CD or through an explicit cancellation procedure
- The librarian can easily create, update, and delete information about titles, borrowers, items and reservations in the system
- The system can run on popular web-browser platforms like Windows Explorer, Netscape navigator etc.
- The system is easy to extend with new functionality

3. Glossary

Key terms are denoted in italics within the use-case specifications.

Item - A tangible copy of a *Title*.

Title - The descriptive identifying information for a book or magazine. Includes attributes like name and description.

Reservation - Whenever a borrower wishes to checkout an *Item* that is not available due to previous checkout by a different borrower a request can be made (a reservation) that locks the borrower in as the next person able to checkout the *Item*.

Actors

Borrower - Interactive actor who uses the library to search for *Titles*, make reservations, checkout, and return *Items*.

Librarian - Interactive actor responsible for maintenance of the inventory, acting on behalf of the borrowers, and general support of the library (non-automated as well).

Master Librarian - Interactive actor, themselves a Librarian, who is also responsible for maintaining the set of librarians for the system.

Registered User - Any interactive user for whom the system maintains a system account. This includes borrowers, librarians, and master librarians. Capabilities include basic login, browsing and searching for *Titles*.

4. Supplementary Specification Document

Objective

The purpose of this document is to define the requirements of the Library system. This document lists the requirements that are not readily captured in the use-cases of the use-case model. The supplementary specification and use-case model together capture a complete set of requirements of the system.

Scope

This supplementary specification defines the non-functional requirements of the system such as reliability, performance, supportability, and security as well as functional requirements that are common across a number of use-cases.

Reference

None

Common Functionalities

- Multiple users must be able to perform their work concurrently

- If the reserved item has been purchased or available, the borrower must be notified

Usability

The desktop user interface shall be Windows NT or Windows 2000 compliant

Reliability

The system shall be 24 hours a day, 7 days a week and not more than 10% down time

Performance

- The system shall support up to 2000 simultaneous users against the central database of any given data
- The system must be able to complete 80% of all transactions within 5 minutes

Supportability

None

Security

- The system must prevent borrowers from changing borrowers information, items information, titles information, and librarians information
- Only Librarian can modify borrowers information, items information, and titles information
- Only Master Librarian can modify librarians information

5. Use – Case Model

Actors

Actor is something external to the system and interacts with the system. Actor may be a human being, device or some other software system.

For Library system, actors might be;

- Librarian
- Borrower

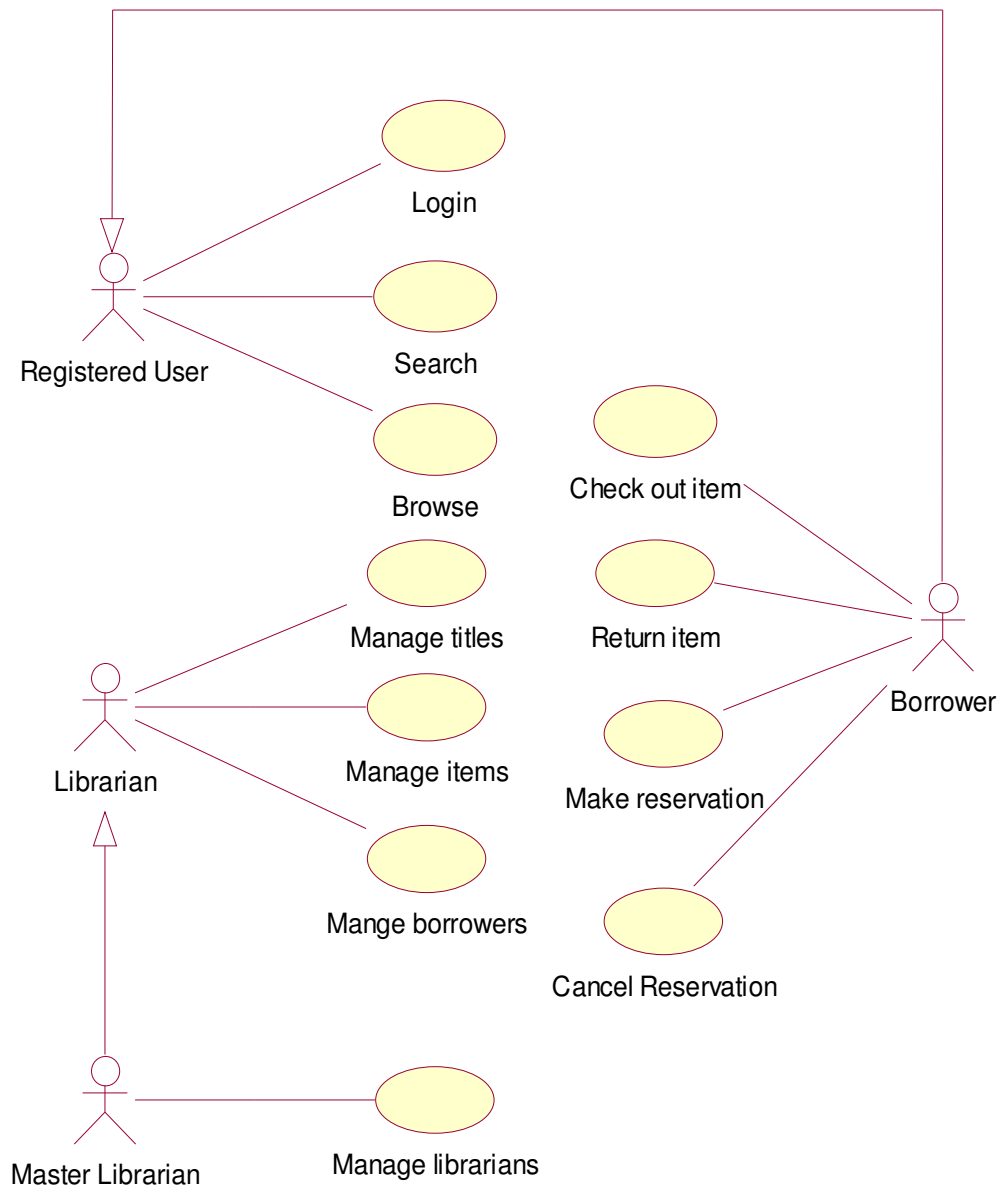
Use – Case

A use-case represents sequence of actions performed by the system that yields an observable result of value for a particular actor. Use-case represents a functional requirement of a system.

For Library system, we can find the following use-cases;

- Login
- Search
- Browse
- Check out item
- Return item
- Make reservation
- Cancel reservation
- Manage titles
- Manage items
- Manage borrowers
- Manage librarians

Use - Case Diagram



5.4 Use – Case Descriptions

5.4.1 Use-Case Specification: Login

5.4.1.1 Description

A registered user can log in and, upon verification, can initiate subsequent actions.

5.4.1.2 Flow of Events

5.4.1.2.1 Basic Flow

1. Initiated when a registered user chooses to Login.
2. The system prompts for username and password.
3. The registered user enters a username and password and submits them.
4. The system authenticates the username and password combination.
5. The system authorizes the registered user according to the role(s) to which the registered user has been assigned.
6. The system displays the main page and awaits subsequent action.

5.4.1.2.2 Alternative Flows

- **Invalid Username/Password**

1. The system displays the Authentication Failed message.

- **Account Locked**

1. The system displays the <appropriate message>.

- **Authentication Service Unavailable**

1. The system displays a Service Unavailable message and does not permit any further attempts to login.

5.4.1.3 Special Requirements

1. Up to three consecutive failed tries to login with invalid username/password combination until locking an account.
2. Minimum password length is 8 characters, and must include a combination of characters including at least one non-alphabetic character.

5.4.1.4 Preconditions

User has an account with the system

5.4.1.5 Post-conditions

5.4.1.5.1 Primary Success Post-condition

The user is considered authenticated and is viewing the main page from which additional actions can be initiated.

5.4.1.5.2 Login Failure

If the Login fails as described in any of the alternatives above, an appropriate message is displayed and the user is not considered authenticated.

5.4.1.6 Notes

1. So far we are not doing much with roles.
2. The “appropriate message” above is vague; we need to come up with how we report this to the user.
3. We need to talk to security people about how reasonable it is to lock the user account after some number of failed attempts. If we keep that rule, we’ll need an Unlock Account use case.

5.4.2 Use-Case Specification: Browse

5.4.2.1 Description

A registered user can browse the contents of the library as a precursor to other actions.

5.4.3 Flow of Events

5.4.3.1 Basic Flow

1. Initiated when a registered user chooses to browse *Titles*.

2. The system responds by displaying all of the *Titles* in the system, along with topical descriptions.
3. The registered user selects a *Title* for further information.
4. The system displays *Title* detail along with the *Items* and the available action on each *Item*.

5.4.3.2 Alternative Flows

- **No records**

1. The system displays message indicating no *Titles* are in the system.

5.4.4 *Special Requirements*

1. The *Titles* will be sorted alphabetically by the name.

5.4.5 *Preconditions*

The user has been authenticated.

5.4.6 *Post-conditions*

5.4.6.1 Primary Success Post-condition

The registered user is viewing a *Title* along with the related *Items*.

5.4.3 Use-Case Specification: Search

5.4.3.1 *Description*

A registered user can search the contents of the library as a precursor to other actions.

5.4.3.2 *Flow of Events*

5.4.3.2.1 Basic Flow

1. Initiated when a registered user chooses to perform a search of *Titles*.
2. The system responds by providing the registered user a means to enter search criteria.
3. The registered user enters search criteria and initiates the query.
4. The system determines results and displays the matching *Titles*, along with topical descriptions.
5. The registered user selects a *Title* for further information.
6. The system displays *Title* detail along with the *Items* and the available action on each *Item*.

5.4.3.2.2 Alternative Flows

- **No matches**

1. The system displays message indicating no *Titles* in the system match this criteria.

5.4.3.3 Special Requirements

1. The search only searches based on the name of the *Item*, not description or any other field.
2. The system shall use the percent sign as a wildcard (in keeping with standard SQL idioms).
3. The results will be sorted alphabetically by the name.

5.4.3.4 Preconditions

The user has been authenticated.

5.4.3.5 Post-conditions

5.4.3.5.1 Primary Success Post-condition

The registered user is viewing a *Title* along with the related *Items*.

5.4.3.6 Notes

1. We might want to combine this with the Search use case. The combined use case could be called Select *Title* and one of the original use cases could be the basic flow and the other would be the alternative.

5.4.4 Use-Case Specification: Make Reservation

5.4.4.1 Description

This use-case starts when the user wants to make a reservation for an item

5.4.4.2 Flow of Events

5.4.4.2.1 Basic flow

1. The system prompts the borrower to enter the item information for which he wants reservation
2. The borrower submits the item information
3. The system marks the item as reserved and associates the borrower with the reservation

5.4.4.2.2 Alternative Flow

None

5.4.4.3 Special requirements

None

5.4.4.4 Pre-conditions

The borrower is viewing a particular title with an item that is not currently available

5.4.4.5 Post-conditions

The item is marked as reserved and the reservation is saved in the database

5.4.4.6 Notes

1. So far there is no nice way to figure out what a borrower has reserved.

5.4.5 Use-Case Specification: Remove Reservation

5.4.5.1 Description

The borrower can remove an existing reservation for an item.

5.4.5.2 Flow of events

5.4.5.2.1 Basic Flow

1. The system prompts the borrower for the item information for which the reservation is removed
2. The borrower enters the item information and submits
3. System marks the item as no longer reserved

5.4.5.2.2 Alternative Flows

None

5.4.5.3 Special requirements

None

5.4.5.4 Pre-conditions

The borrower is viewing a particular *Title* with an *Item* that is reserved by the borrower.

5.4.5.5 Post-conditions

The previously reserved *Item* is no longer reserved.

5.4.6 Use-Case Specification: Check out Item

5.4.6.1 Description

This use-case starts when the borrower wishes to check out an item from the library

5.4.6.2 Flow of Events

5.4.6.2.1 Basic Flow

1. The borrower performs a search for the desired titles
2. The system prompts the borrower to enter search criteria
3. The borrower specifies the search criteria and submits
4. The system locates matching titles and displays them to the borrower
5. The borrower selects titles to check out
6. The system displays the details of titles as well as whether or not there is an available item to be checked out
7. The borrower confirms the check out
8. the system checks out the item
9. Steps 1-8 can be repeated as often as needed by the borrower
10. The borrower completes the check out
11. The system notifies the Librarian that the borrower has concluded the check out item session and displays instructions for the borrower to collect the items

5.4.6.2.2 Alternative Flows

None

5.4.6.3 Special requirements

None

5.4.6.4 Pre-conditions

The borrower is viewing a particular *Title* with an *Item* that is currently available.

5.4.6.5 Post-conditions

The *Item* is demarked as checked out to the borrower.

5.4.7 Use-Case Specification: Return Item

5.4.7.1 Description

This use-case starts when the borrower wishes to return an item

5.4.7.2 Flow of Events

5.4.7.2.1 Basic Flow

1. The system prompts the borrower to enter the item information he wants to return
2. The borrower enters the item information and submits
3. The system marks the item as available

5.4.7.2.2 Alternative Flows

None

5.4.7.3 Special requirements

None

5.4.7.4 Pre-conditions

The borrower is viewing a particular *Title* with an *Item* that is checked out by the borrower.

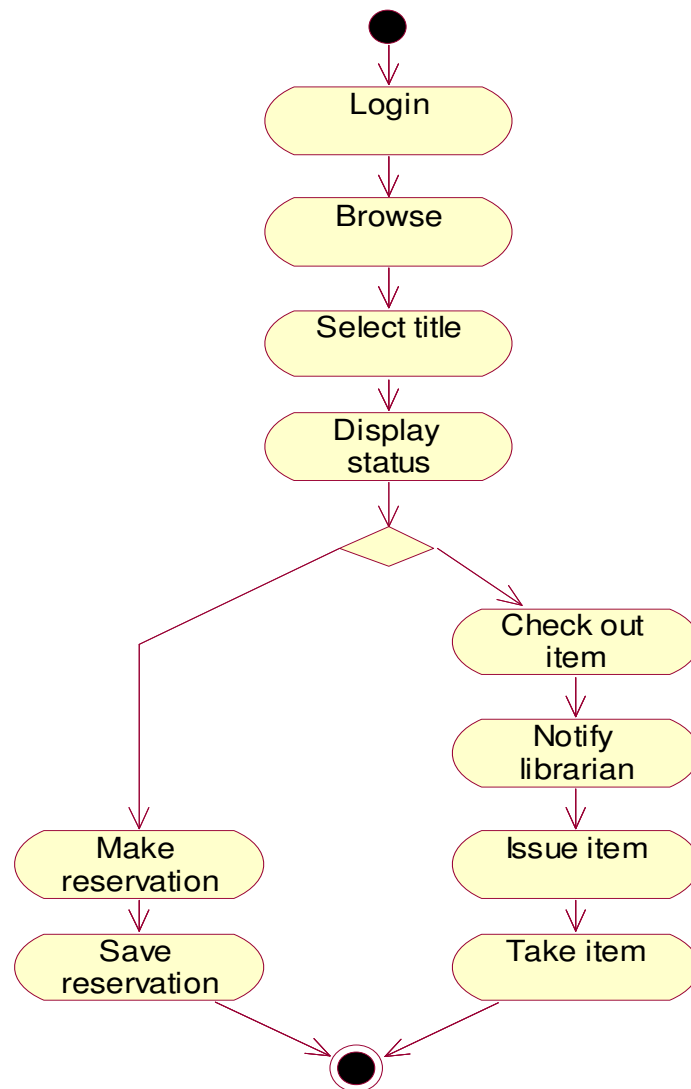
5.4.7.5 Post-conditions

The *Item* is demarked as available.

5.4.7.6 Notes

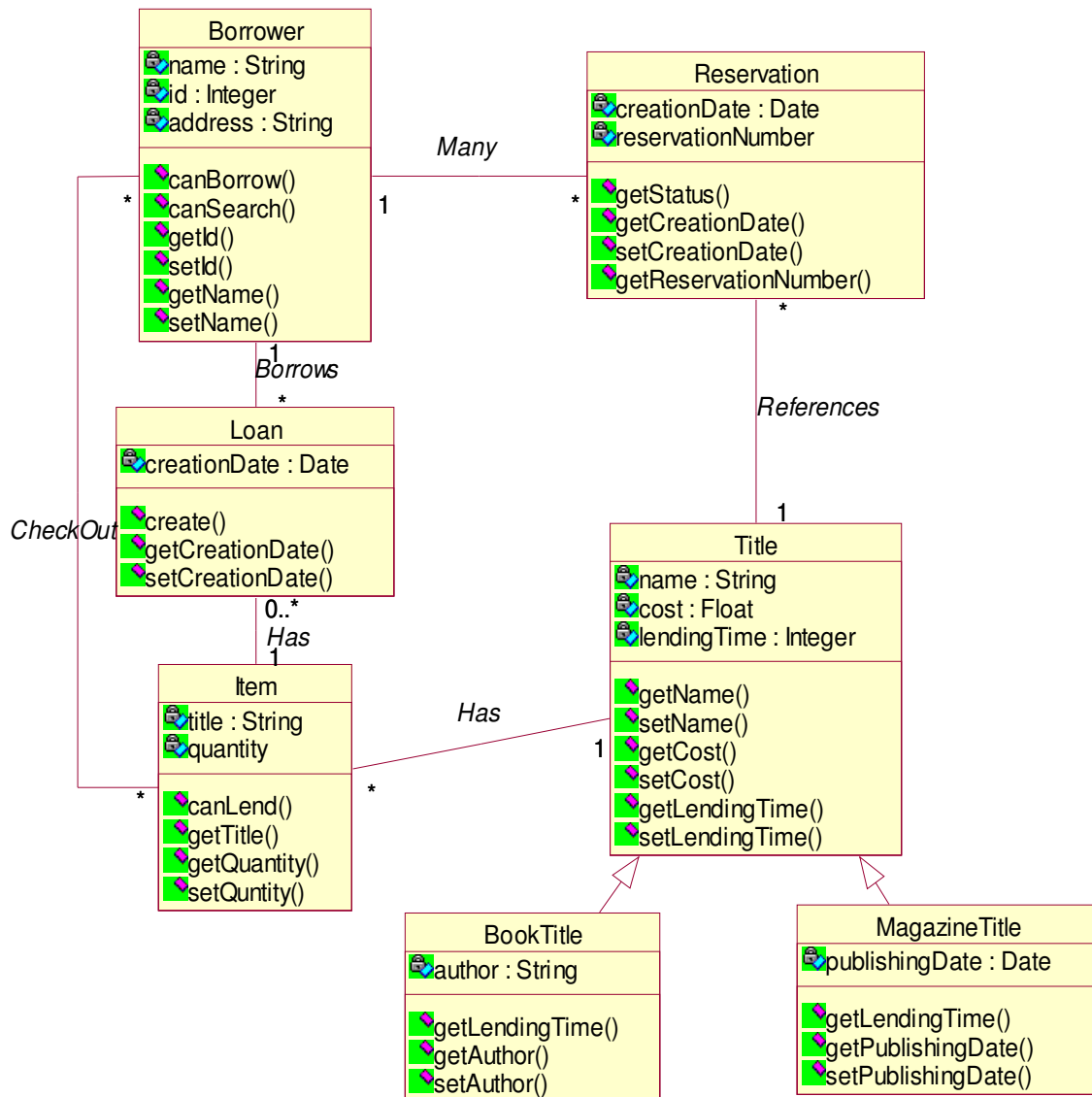
A reasonable future enhancement would be to notify anyone with a reservation on the *Item*.

5.5 Activity Diagram



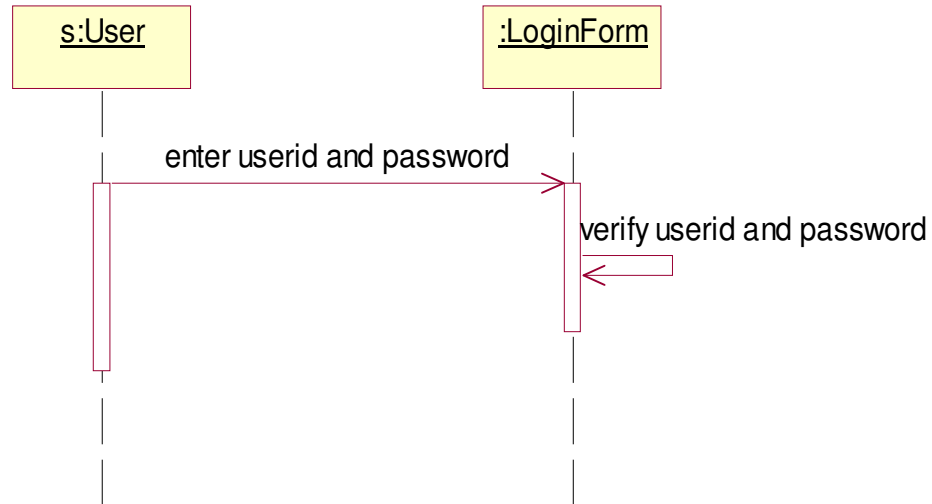
6. Design Model

6.1 Class Diagram

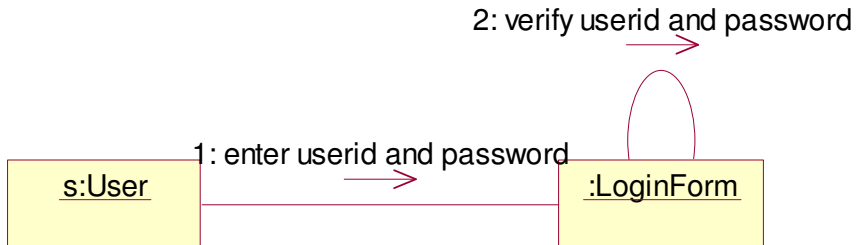


6.2 Sequence Diagram and Collaboration diagrams

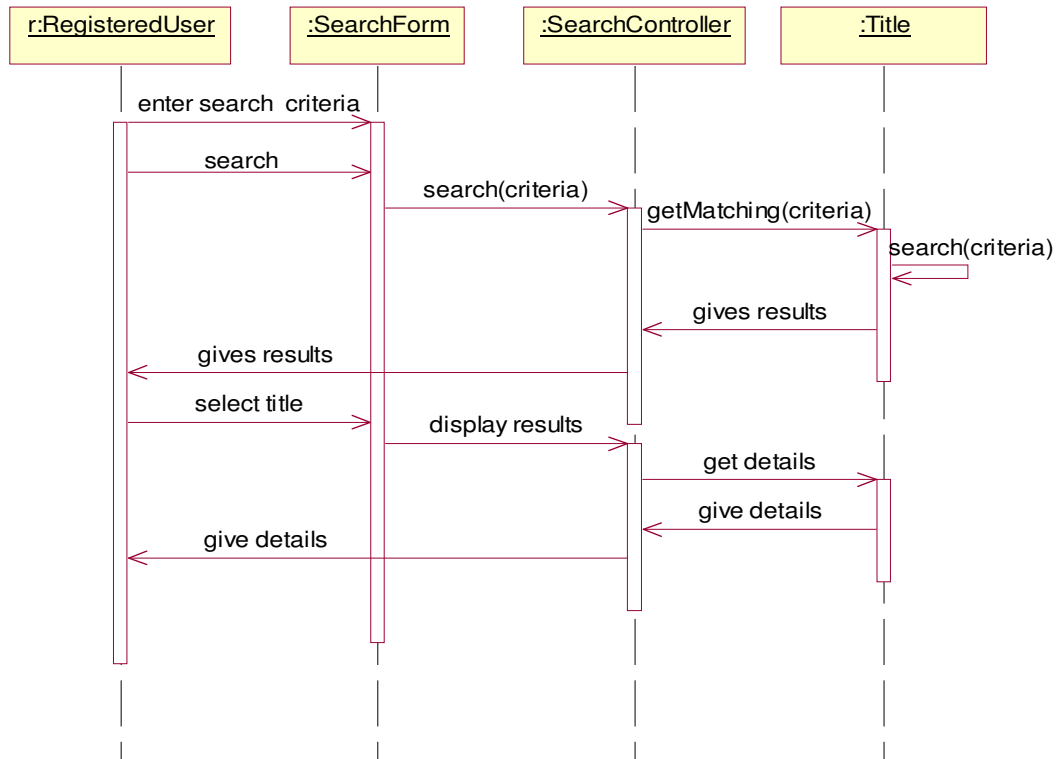
6.2.1 Sequence Diagram for Login Use-case



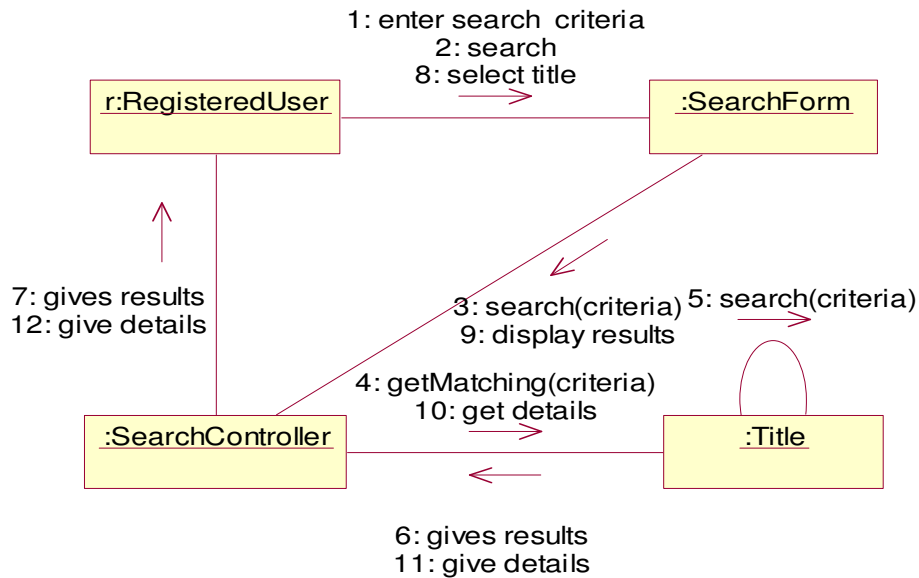
6.2.2 Collaboration Diagram for Login Use-case



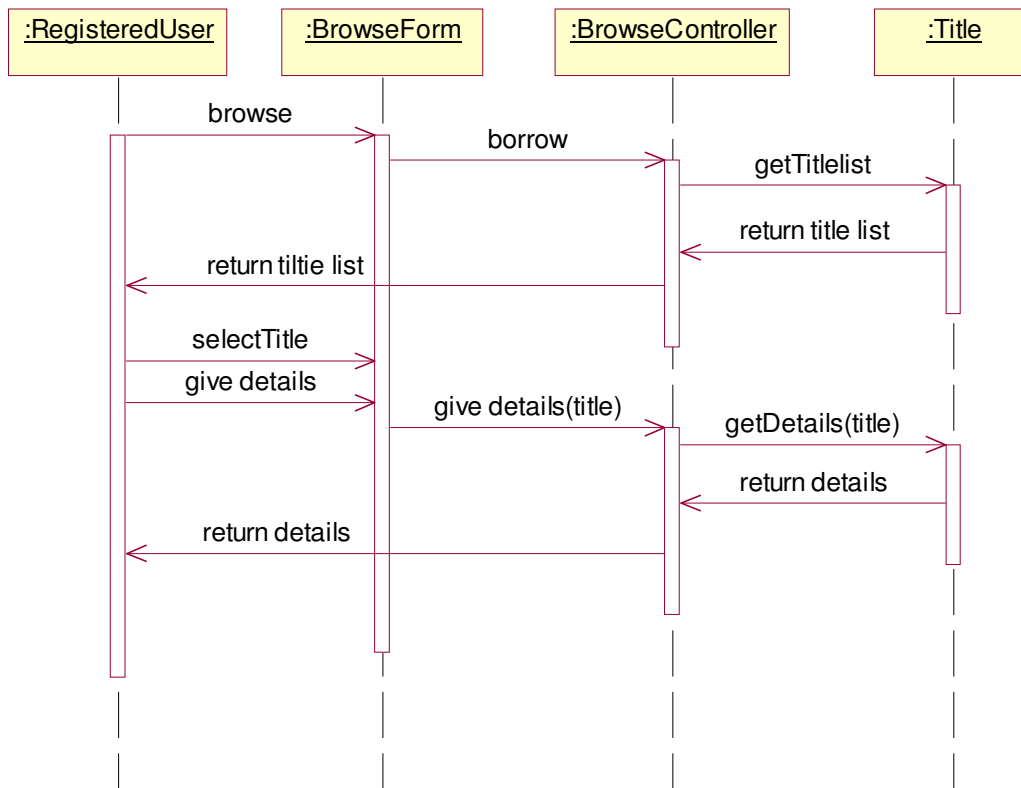
6.2.3 Sequence Diagram for Search Use-case



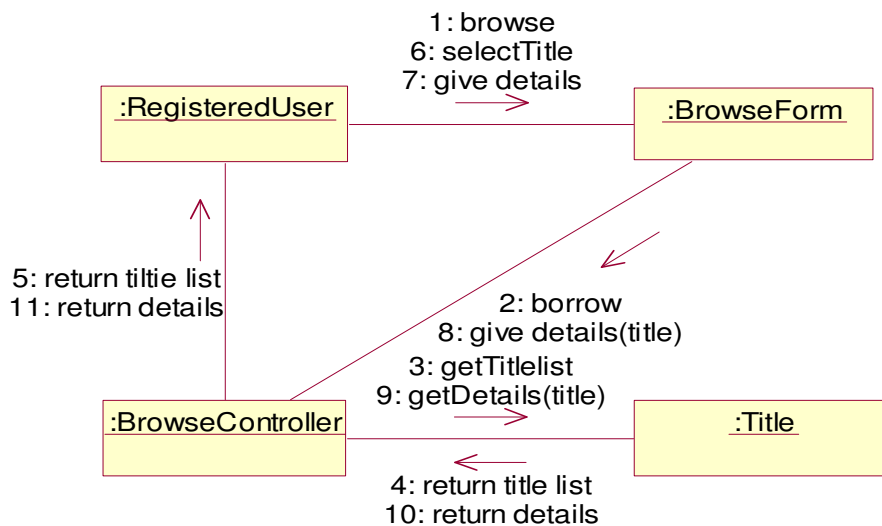
6.2.4 Collaboration Diagram for Search Use-case



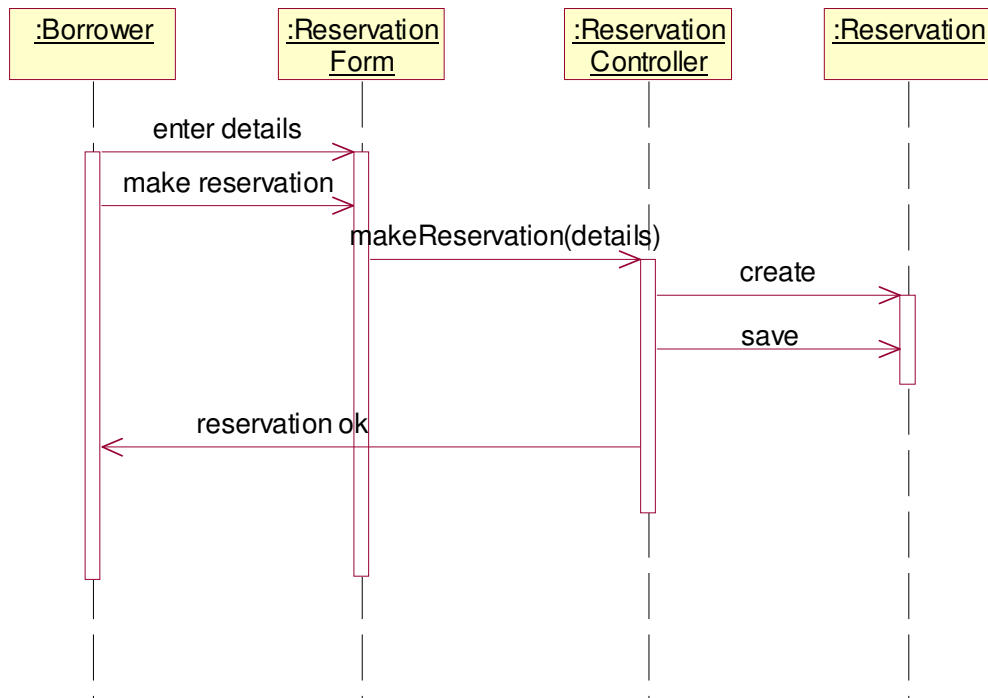
6.2.5 Sequence Diagram for Browse Use-case



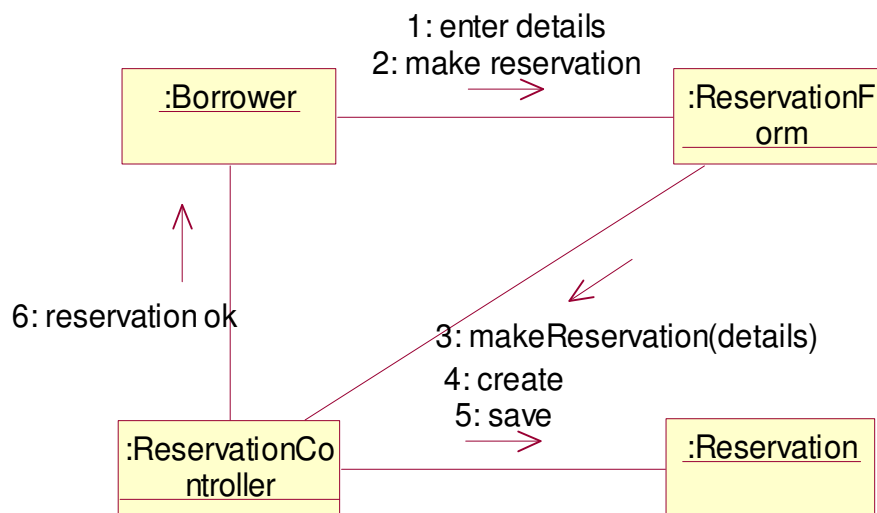
6.2.6 Collaboration Diagram for Browse Use-case



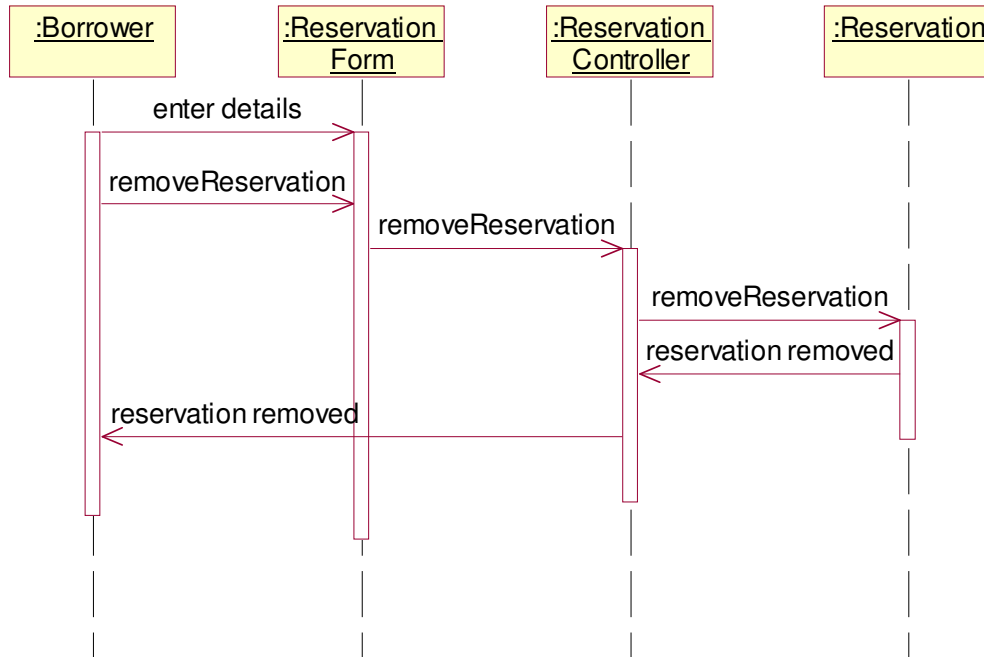
6.2.7 Sequence Diagram for Make Reservation Use-case



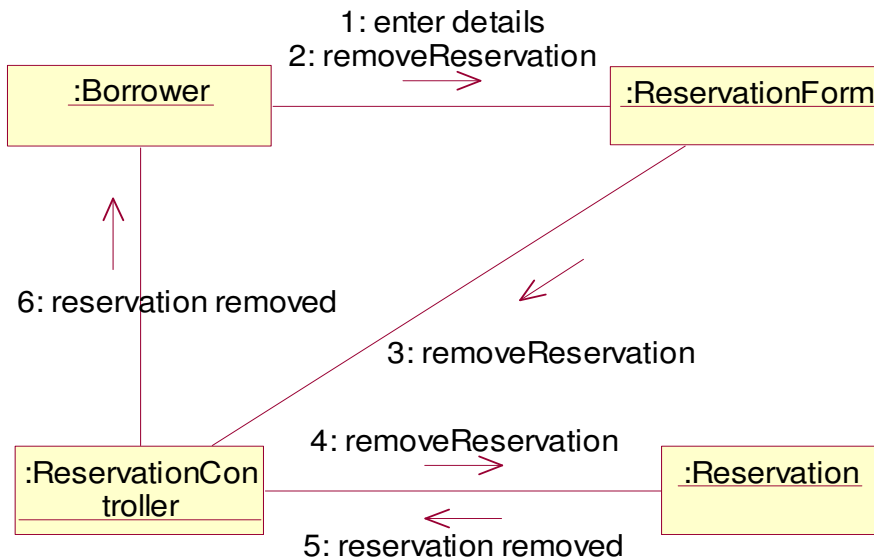
6.2.8 Collaboration Diagram for Make Reservation use-case



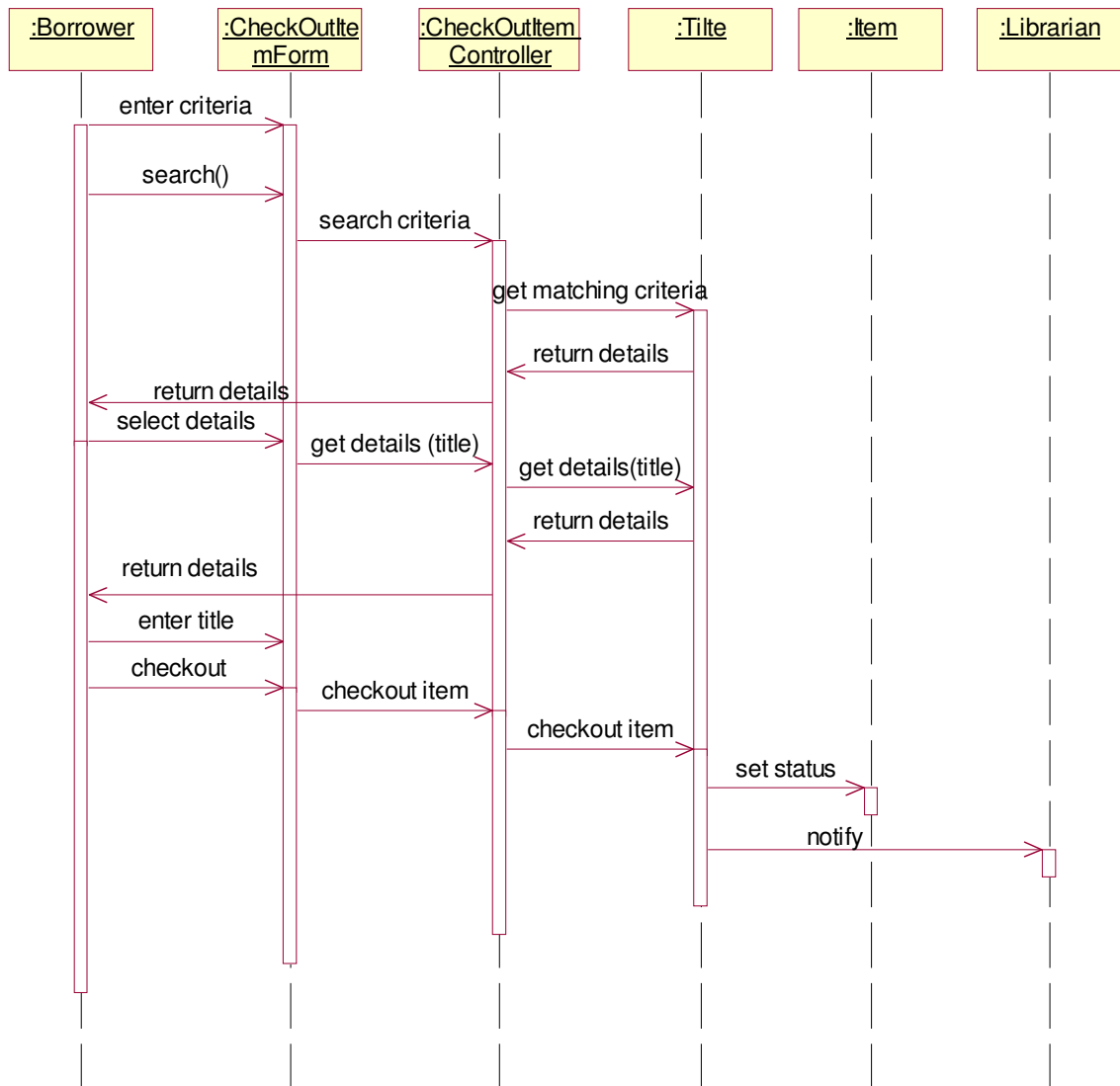
6.2.9 Sequence Diagram for Remove Reservation Use-case



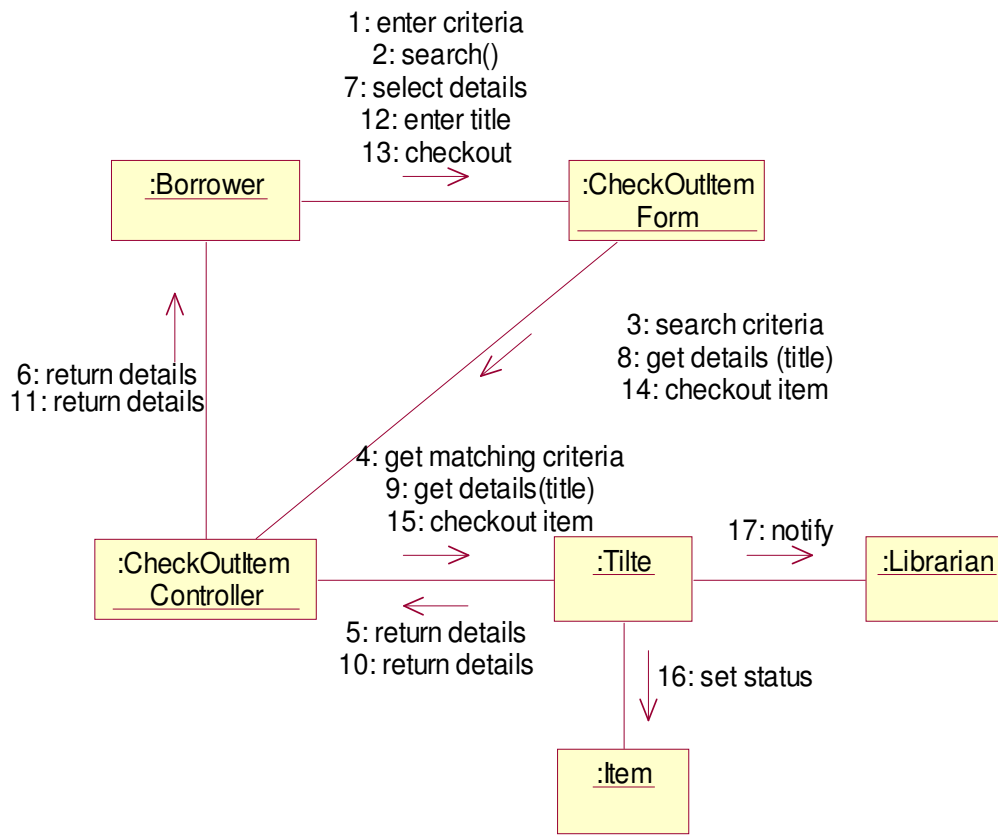
6.2.10 Collaboration Diagram for Remove Reservation Use-case



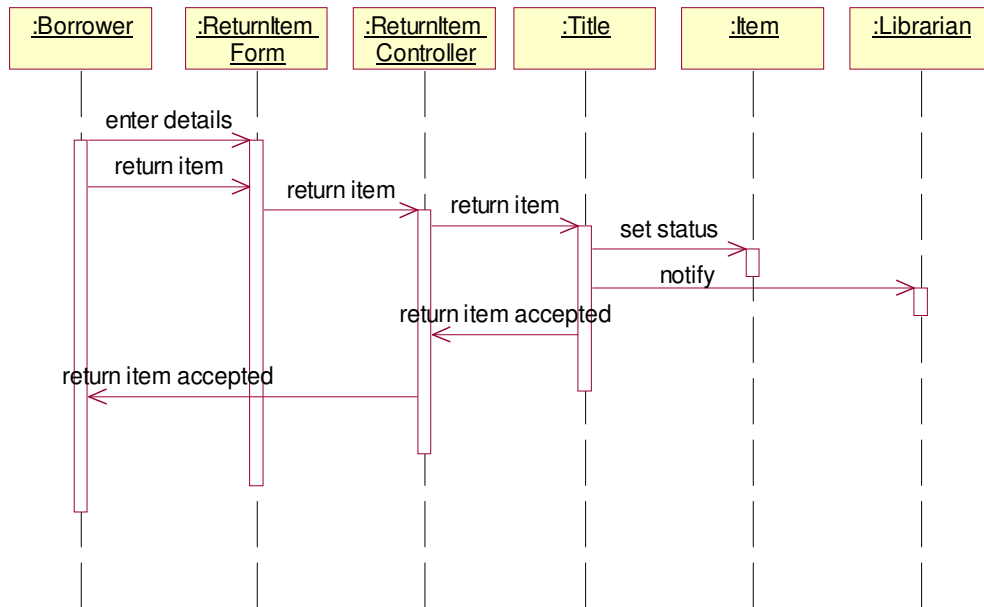
6.2.11 Sequence Diagram for Check Out Item Use-case



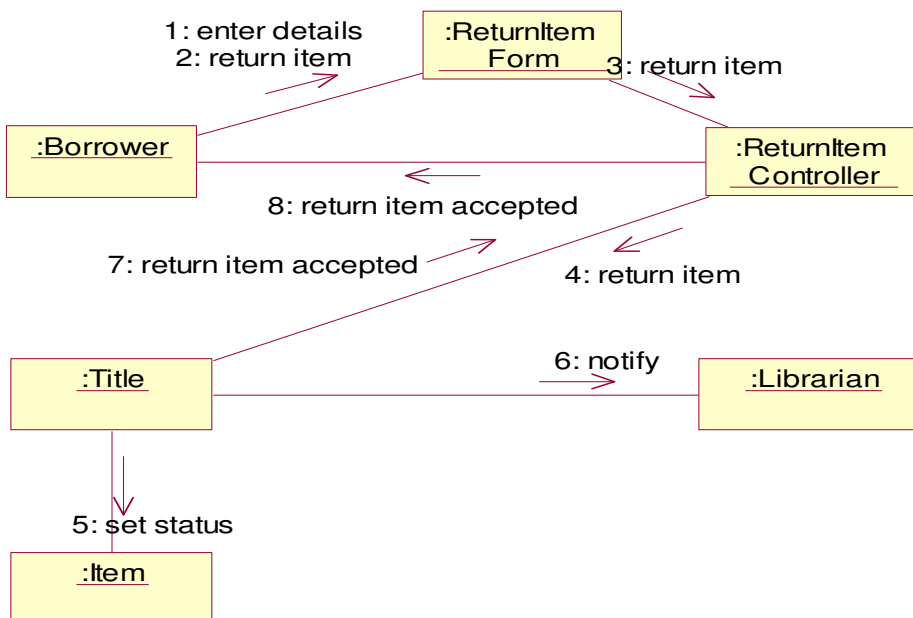
6.2.12 Collaboration Diagram for CheckOut Item Use-case



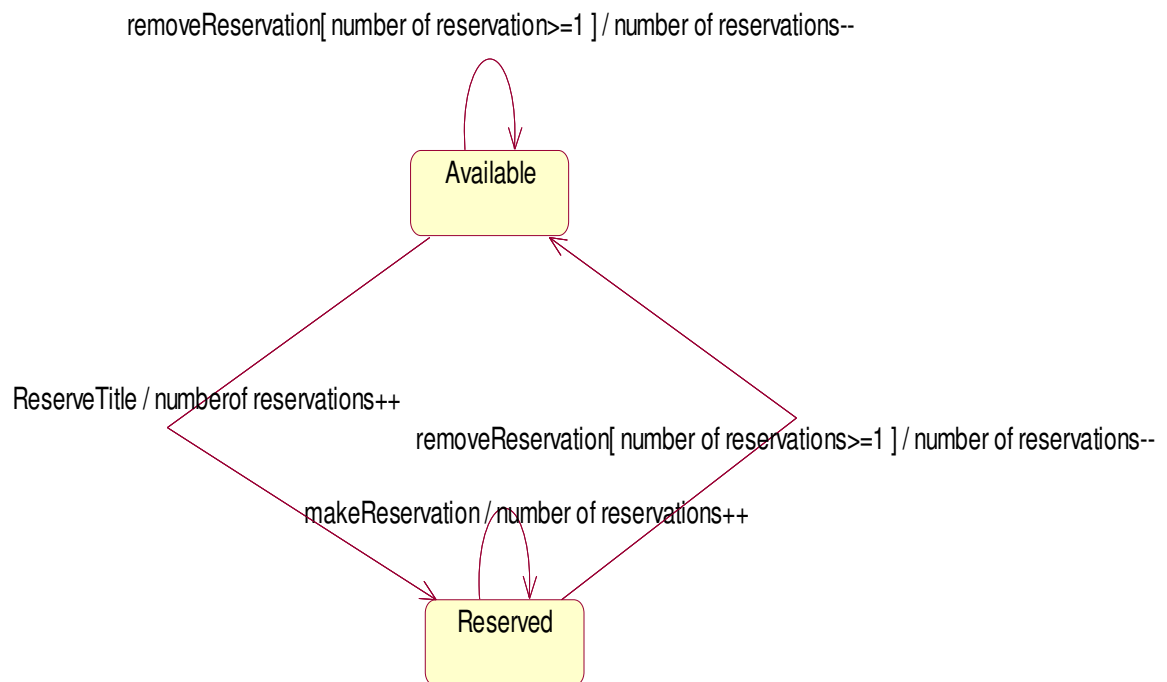
6.2.13 Sequence Diagram for Return Item Use-case



6.2.14 Collaboration Diagram for Return Item Use-case

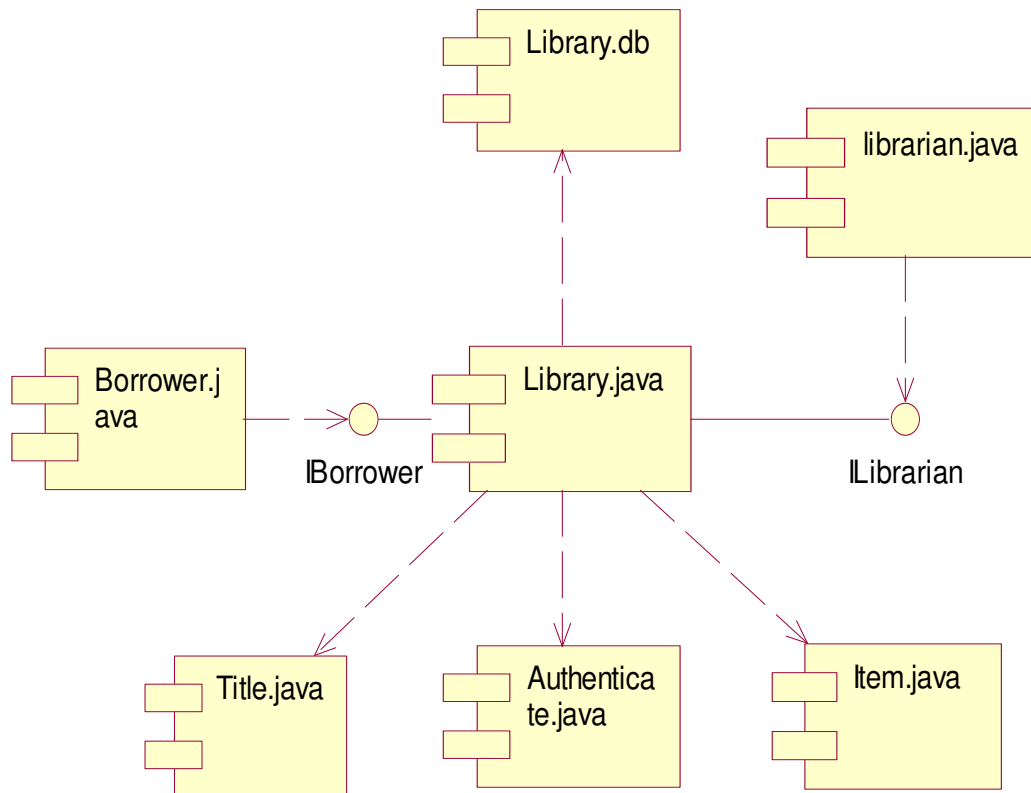


6.3 State chart Diagram for Title Class

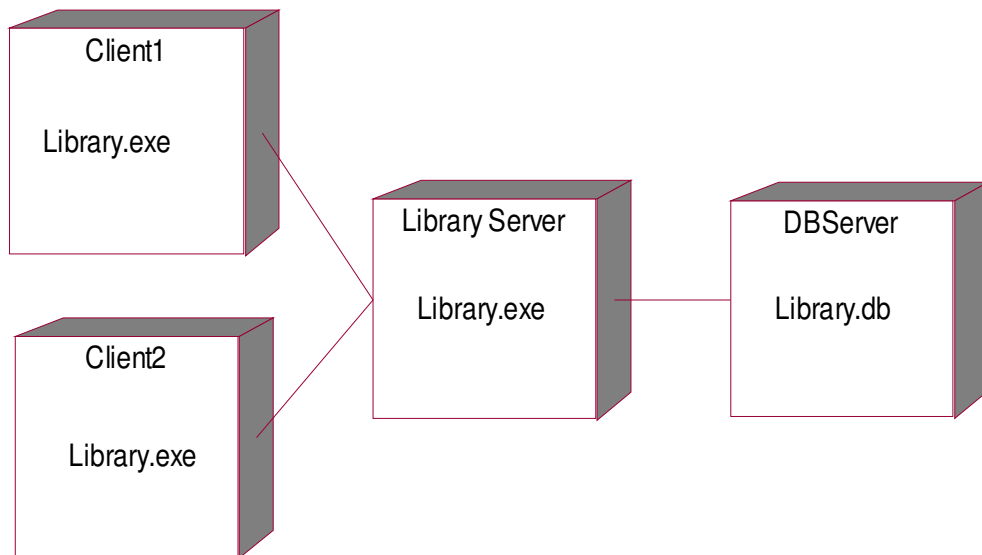


7. Deployment Model

7.1 Component Diagram



7.2 Deployment Diagram



III) Case Study: Restaurant System

1. Problem statement

The system is intended to support the day-to-day operations of a restaurant by improving the processes of making reservations and allocating tables to customers.

The Restaurant system provides the facilities like

- Record Booking
- Cancel Booking
- Record Arrival
- Table Transfer

The new system can offer diners eat at the restaurant without making an advance booking, if a free table is available. This is known as Walk-in.

The new system should display the same information as the existing booking sheet and in same format, to make it easy for restaurant staff to transfer, to the new system. When new bookings are recorded or changes made to existing bookings, the display should be immediately updated, so that restaurant staff is working with the latest information available.

2. Vision Document

The vision document describes the higher level requirements of the system, specifying the scope of the system.

The vision document of restaurant system might be

- It is a support system for restaurant
- The restaurant makes bookings, cancel bookings, record arrivals and table transfers of the customers.
- The receptionist is the employee of the restaurant who interacts with the customer whose work is supported by the system.

- The customer rings up to make a booking there is a suitable table free at the required day and time and the required day and time and the receptionist enters customer's name, phone no. and records booking.
- When the customer arrives, his arrival is updated in the system and waiter attends to them.
- The customer can also cancel booking what he made or transfer the booking to another day or time.
- The receptionist can easily record , update and cancel the information about the bookings and customers
- The customers eat in restaurants even with out any reservations or bookings called Walk-in.

3. Glossary

This document is used to define terminology specific to the problem domain, explaining terms which may be unfamiliar to the reader. This document can be used as informal data dictionary capturing data definitions, key terms.

Booking:

An assignment of a table to a party of dinners for a meal

Covers:

The number of diners that a booking is made for

Reservation:

An advanced booking for a table at a particular time

Places:

The number of diners that can be seated at a particular table

Walk-In:

A booking that is not made in advance.

Actors:**Customer:**

The person making a reservation

Diner:

A person eating at the restaurant

4. Supplementary Specification Document

4.1 Objective

The purpose of this document is to define the requirements of the restaurant system. The supplementary specification lists the requirements those are not readily captured in the use cases of the use case model. The supplementary specification and the use case model together capture a complete set of requirements of the system.

4.2 Scope

This specification defines the non-functional requirements of the system such as reliability, usability, performance, supportability and security. As well as functional requirements that are common across a no. of use cases.

4.3 References

None

4.4 Common Functionalities

- Multiple customers can be able to visit a restaurant.
- Customers who have recorded booking must be notified.

4.5 Usability

The desktop user interface shall be Windows NT and Windows 2000 complaint.

4.6 Reliability

The system shall be available 24 hrs a day, 7 day a week to no more than 10% downtime.

4.7 Performance

The system shall support up to 1000 simultaneous users against the central data base of any given time.

4.8 Supportability

None

4.9 Security

- The system must prevent customers from changing information like record booking, date and timings of reservations.
- Only receptionist can modify customer's information of record booking, updates and cancel booking.
- Only proprietor can modify the receptionist information.

5. Use - Case Model

5.1 Actors

Actor is something external and interacts with the system. Actor may be a human being or some other software system.

For restaurant system

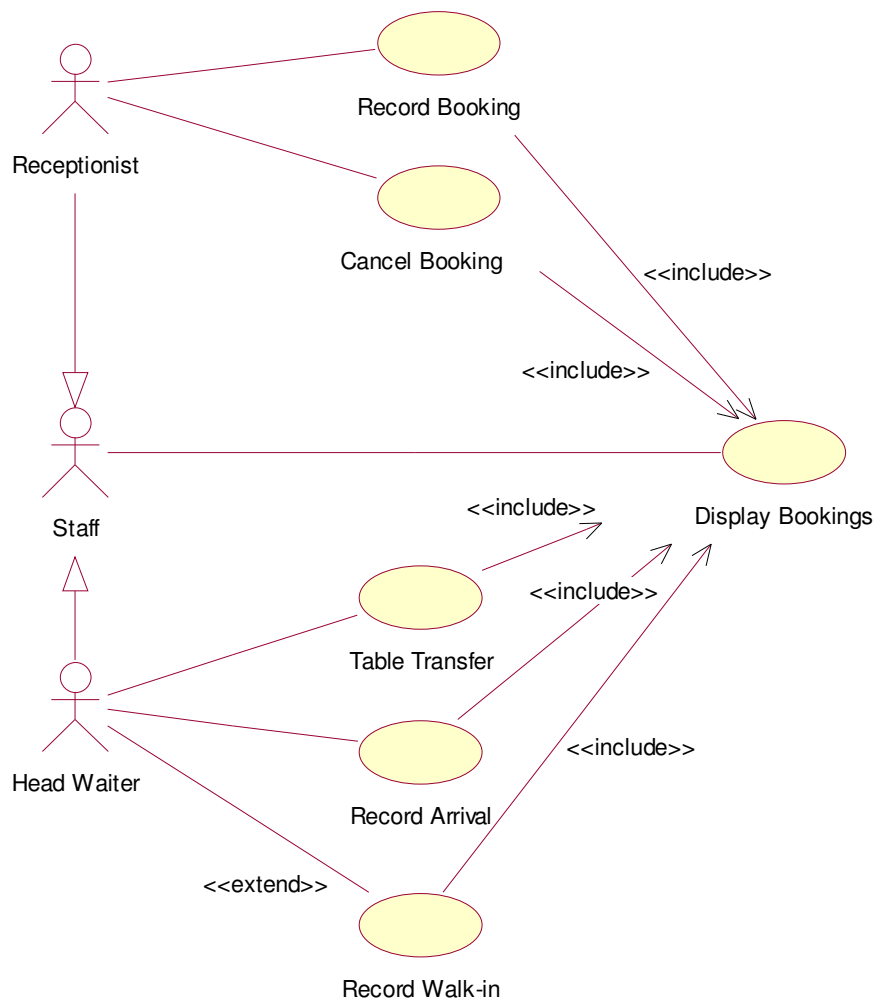
- Receptionist

5.2 Use cases

Use cases represent the functional requirements:

- Record Booking
- Cancel Booking
- Table Transfer
- Record Arrival
- Record Walk-in
- Display Bookings

5.3 Use Case Diagram



5.4 Use-case Specifications

5.4.1 Use-Case Specification: Record Booking

5.4.1.1 Description

This use case starts when the receptionist want to record the bookings the customer have made.

5.4.1.2 Flow of Events

5.4.1.2.1 Basic Flow

1. The bookings are recorded into the restaurant database.
2. The receptionist enters the date of the requested reservation.
3. The system displays the bookings for that date.
4. There is a suitable table available; the receptionist enters the customers name and phone number the time of booking, the number of covers and the table.
5. The system records and displays new booking.

5.4.1.2.2 Alternate Flow

1. The receptionist enters the date of the requested reservation.
2. The system displays the bookings for that date.
3. No suitable table is available and the use-case terminates.

5.4.1.3 Precondition

The customer wants to reserve the table in restaurant on particular date.

5.4.1.4 Post condition

The customer successfully reserves the table on the required date.

5.4.2 Use-Case Specification: Record Arrival

5.4.2.1 Description

This use case starts when customer arrived at restaurant.

5.4.2.2 Flow of Events

5.4.2.1.1 Basic Flow

1. The headwaiter enters the current date.
2. The system displays the booking for the date.
3. The headwaiter confirms arrival for the selected booking.
4. The system records this and updates the display, marking the customer as having arrived.

5.4.2.1.2 Alternate Flow

1. The headwaiter enters the current date.
2. The system displays the bookings for that date.
3. There are no bookings recorded on the system for the customer, so the head waiter creates a walk-in booking, by entering the time of booking, number of covers and the table number.
4. The system records and displays the new bookings.

5.4.2.3 Precondition

The customer must reserve the table.

5.4.2.4 Post condition

The arrival of the customer is updated.

5.4.3 Use-Case Specification: Display Booking

5.4.3.1 Description

This use case starts when the customer wants to see the bookings on that date.

5.4.3.2 Flow of Events

5.4.3.2.1 Basic Flow

1. The user enters a date.
2. The system displays the bookings for that date.

5.4.3.2.2 Alternate Flow

1. System displays no bookings on that date.

5.4.3.3 Pre-condition

User should be the member of restaurant and must enter the date.

5.4.3.4 Post condition

System displays the bookings on the entered date.

5.4.4 Use-Case Specification: Cancel Booking

5.4.4.1 Description

This use case starts when the customer wants to cancel the booking he has made.

5.4.4.2 Flow of Events

5.4.4.2.1 Basic Flow

1. The receptionist selects required booking.
2. The receptionist cancels the booking.
3. The system asks the receptionist to confirm the cancellation.

4. The receptionist answers 'yes', so the system records the cancellation and updates display.

5.4.4.2.2 Alternate Flow

1. The system displays no reservations of customers.

5.4.4.3 Precondition

The customer must have already booked the table.

5.4.4.4 Post condition

The booking of the customer gets canceled.

5.4.5 Use-case Specification: Table Transfer

5.4.5.1 Description

This use case specification starts when head waiter wants to transfer the booked table.

5.4.5.2 Flow Of Events

5.4.5.2.1 Basic Flow

The head waiter selects required booking.

1. The head waiter changes the table allocation of the booking.
2. The system records the alteration and updates display.

5.4.5.2.2 Alternative Flows

1. When it is not possible move a booking to a table that is already occupied

5.4.5.3 Precondition

The table is already booked by another person.

5.4.5.4 Post condition

The table gets transferred.

5.4.6 Use-case Specification: Record Walk In

5.4.6.1 Description

This use case starts when some one arrives to eat in the restaurant with out reservation.

5.4.6.2 Flow Of events

5.4.6.2.1 Basic Flow

1. The headwaiter performs display bookings use case.
2. The head waiter enters the time, the number of covers and table allocated to customer.
3. The system records and displays new booking.

5.4.6.3 Alternative Flow

No table is available as free.

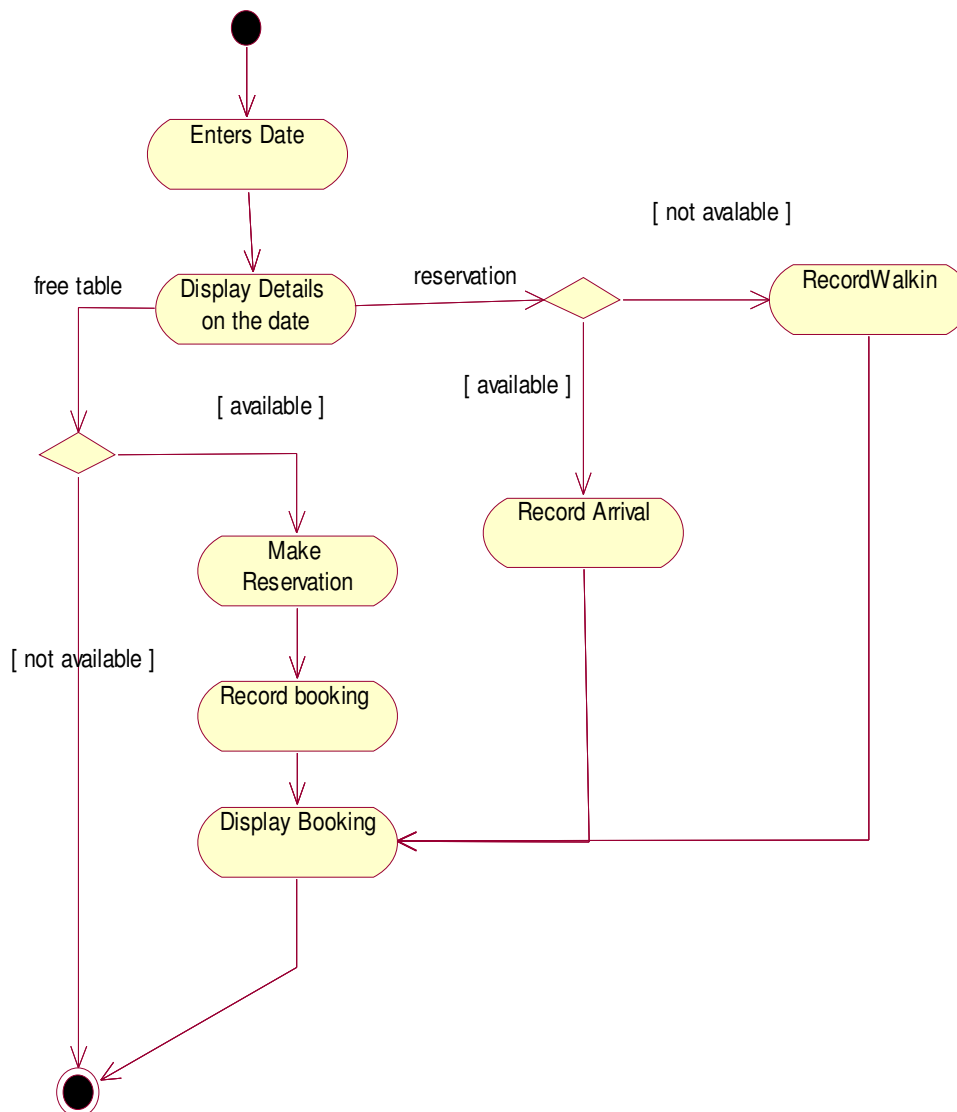
5.4.6.4 Pre-conditions

The person should not reserve the table in advance.

5.4.6.5 Post conditions

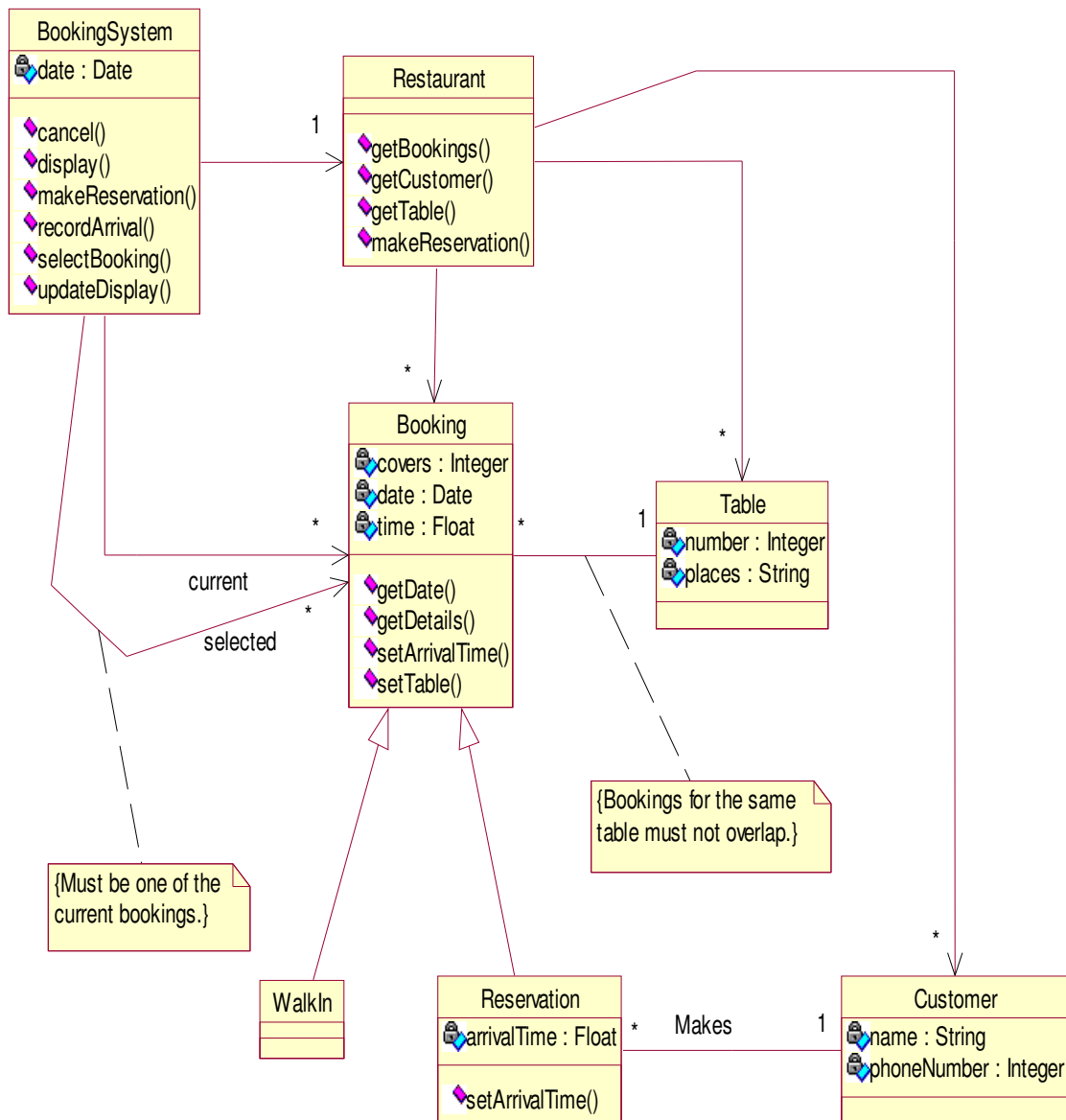
The system records walk in.

5.5 Activity Diagram



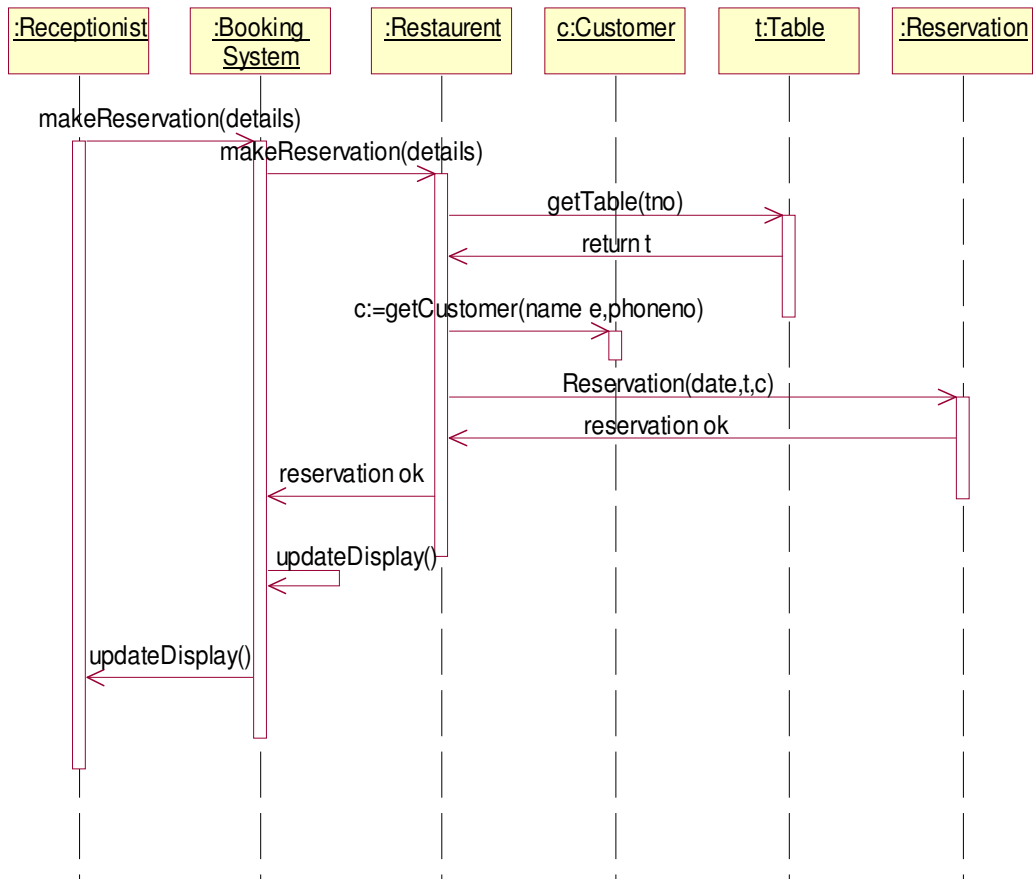
6. Design Model

6.1 Class Diagram

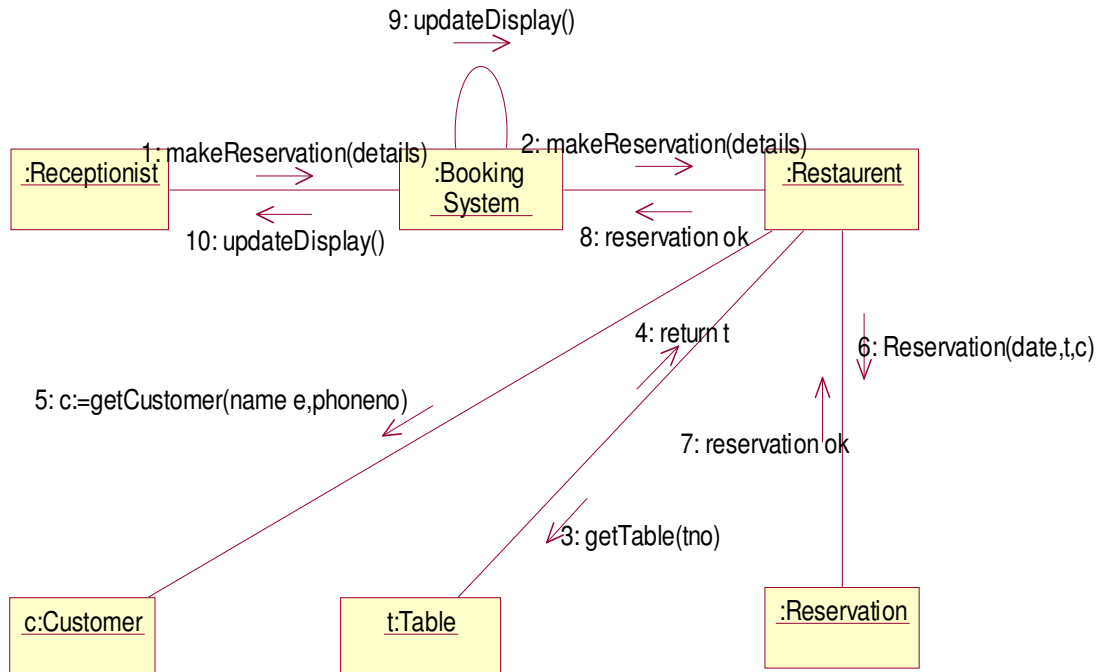


6.2 Sequence and Collaboration Diagrams

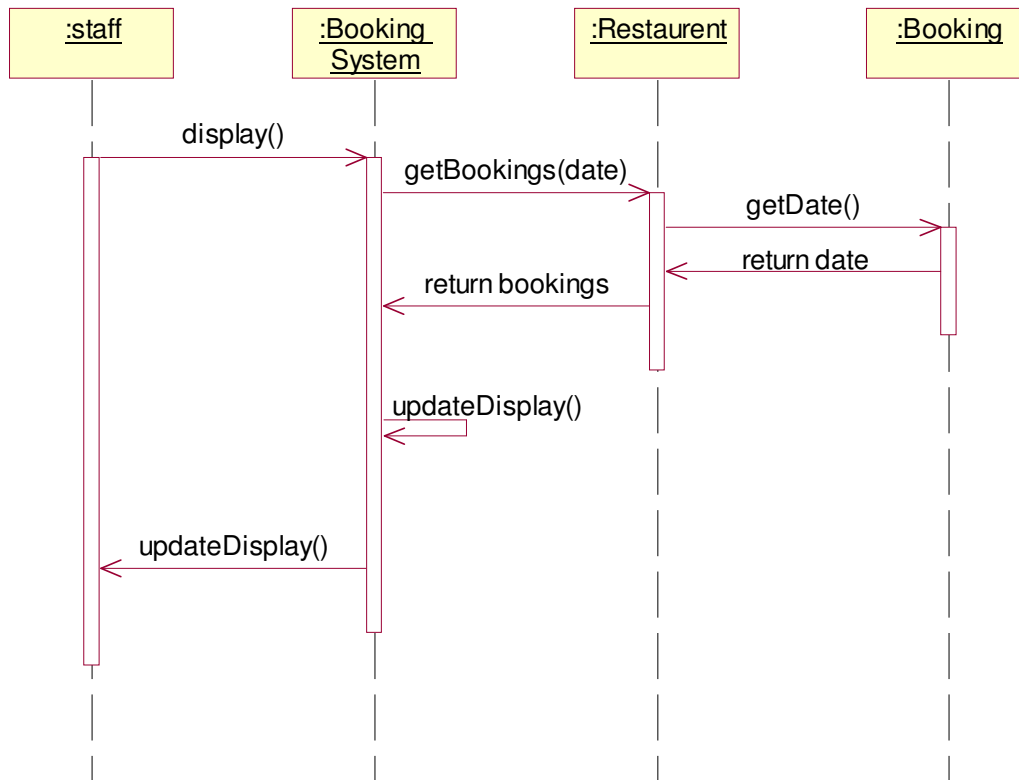
6.2.1 Sequence Diagram for *Record Booking* Use-case



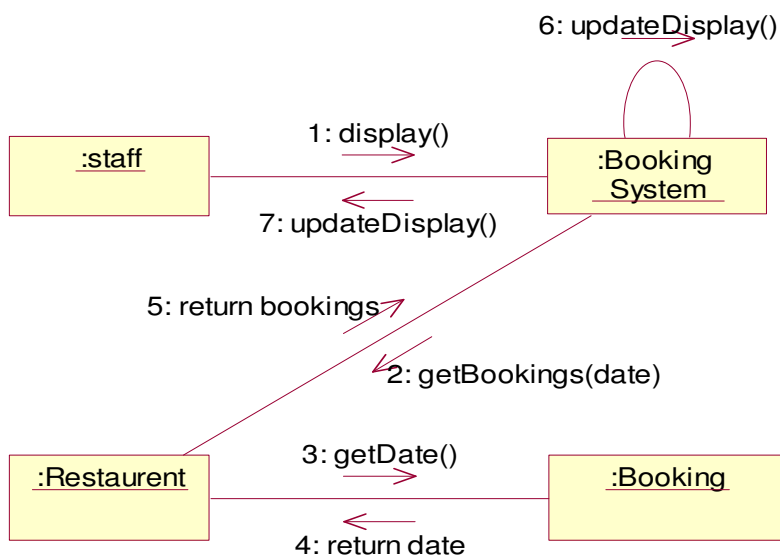
6.2.2 Collaboration Diagram for *Record Booking* Use-case



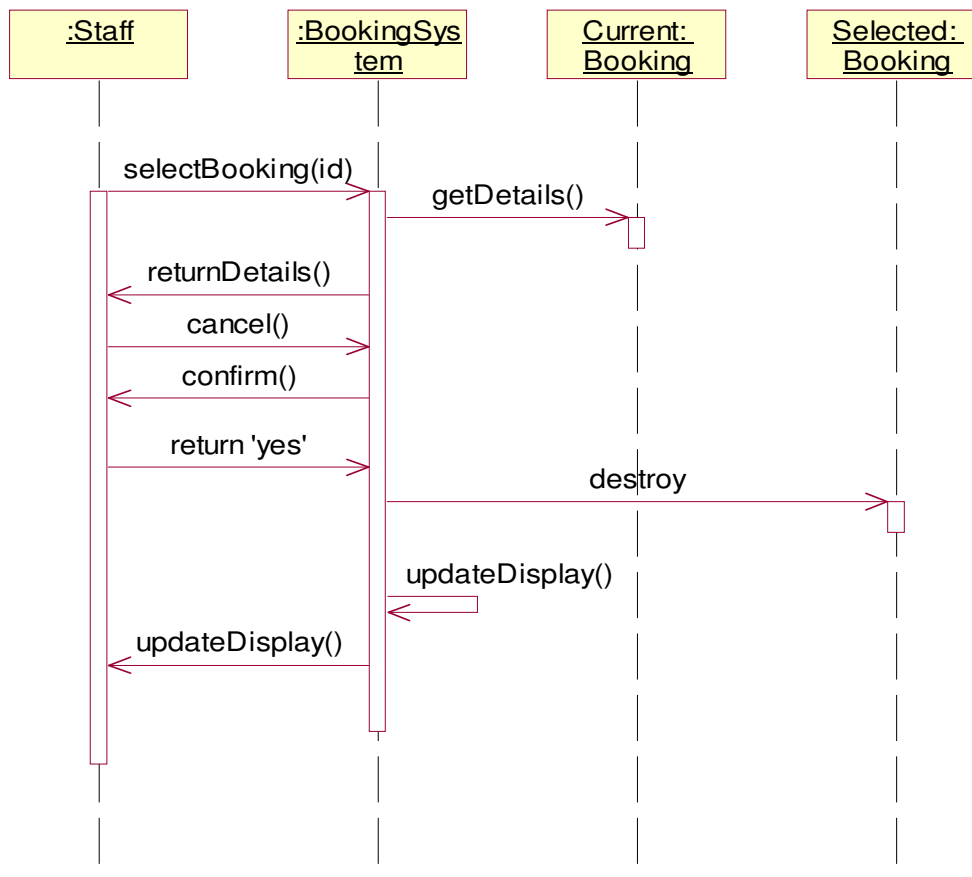
6.2.3 Sequence Diagram for *Display Booking* Use-case



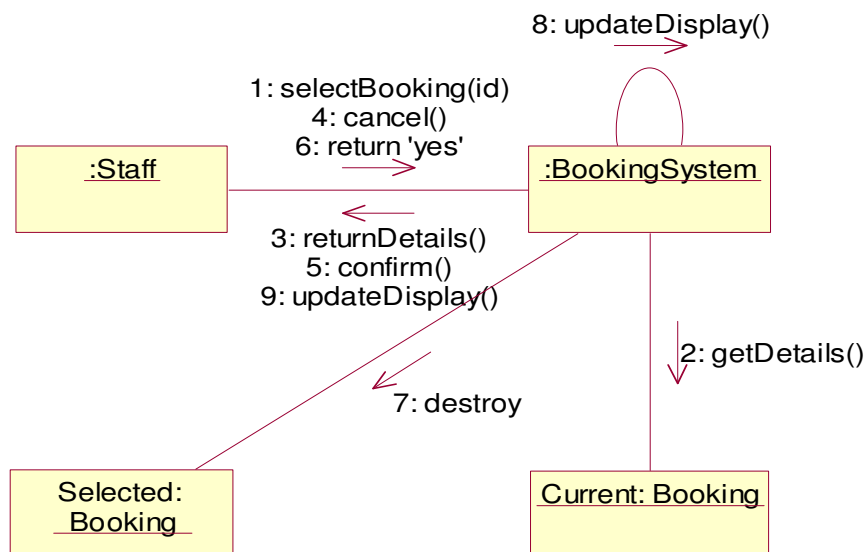
6.2.4 Collaboration Diagram for *Display Booking* Use-case



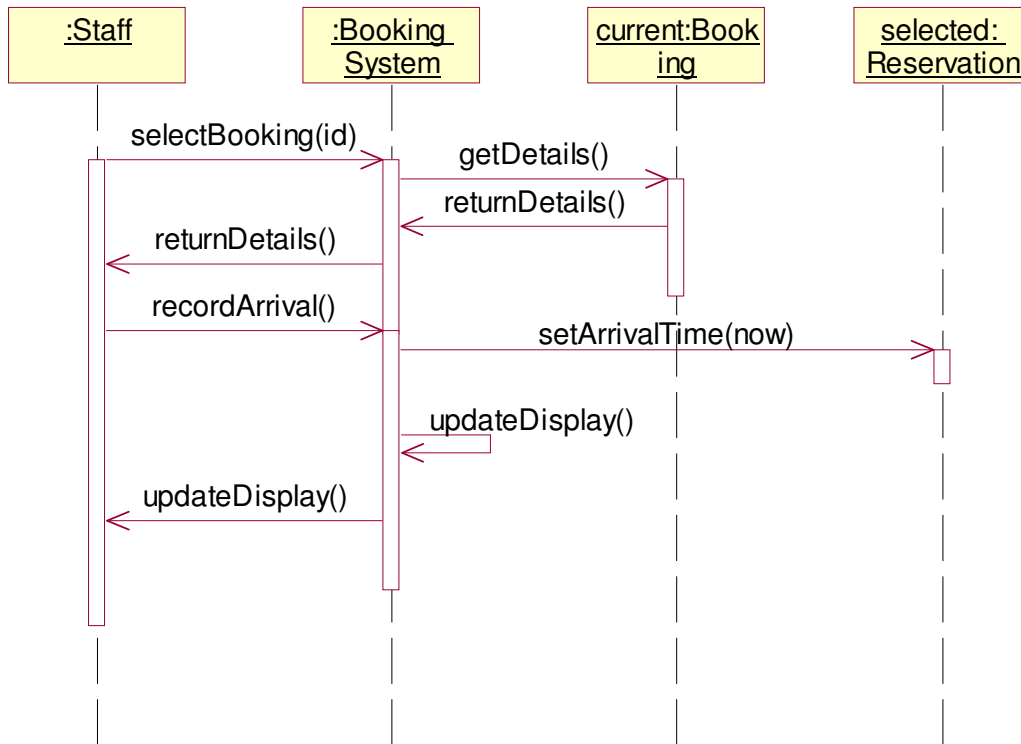
6.2.5 Sequence Diagram for *Cancel booking* Use-case



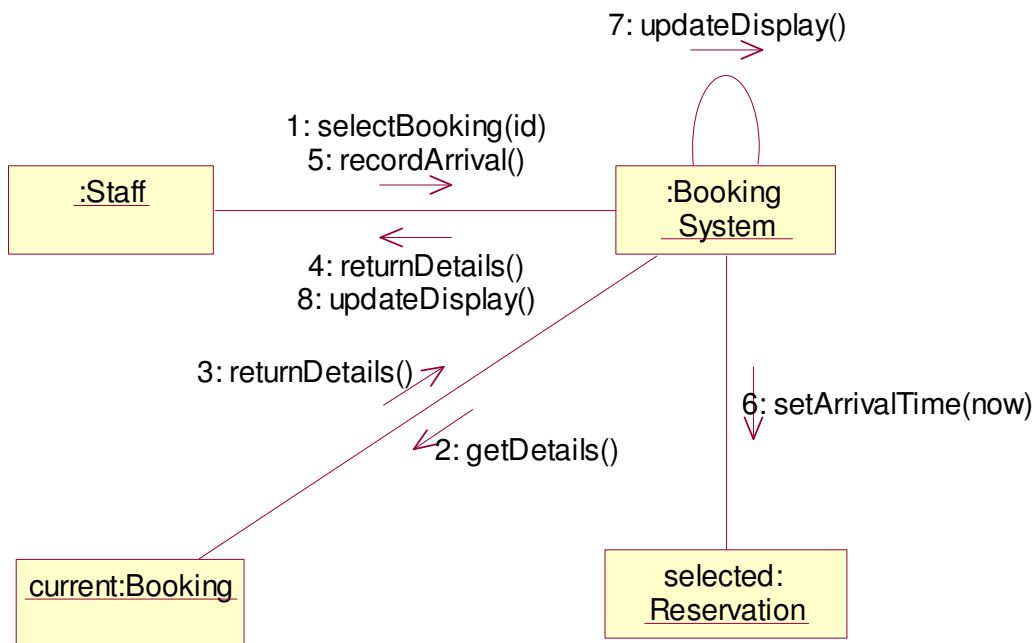
6.2.6 Collaboration Diagram for *Cancel booking* Use-case



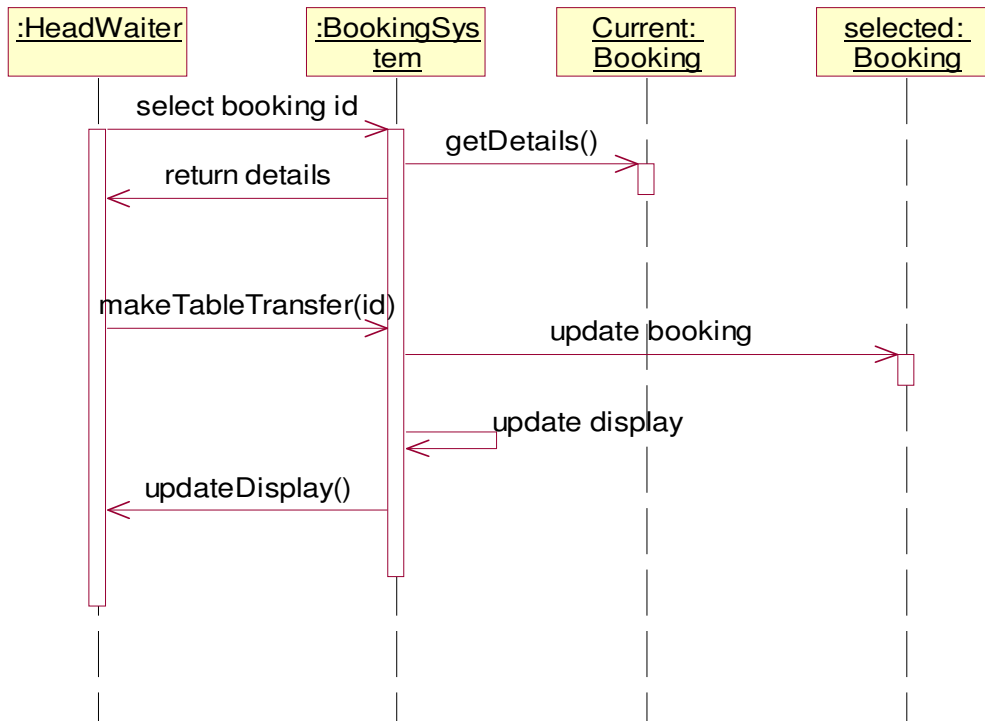
6.2.7 Sequence Diagram for *Record Arrival* Use-case



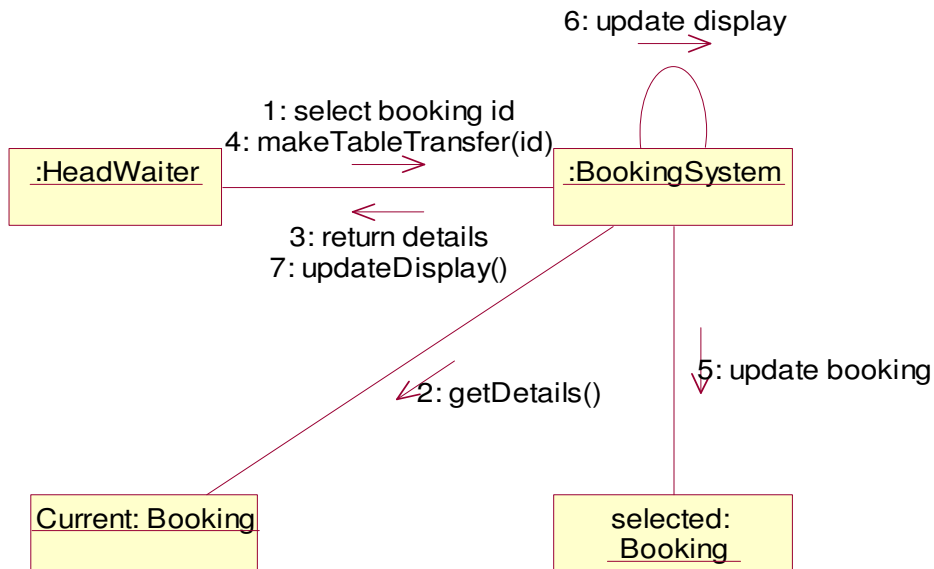
6.2.8 Collaboration Diagram for *Record Arrival* Use-case



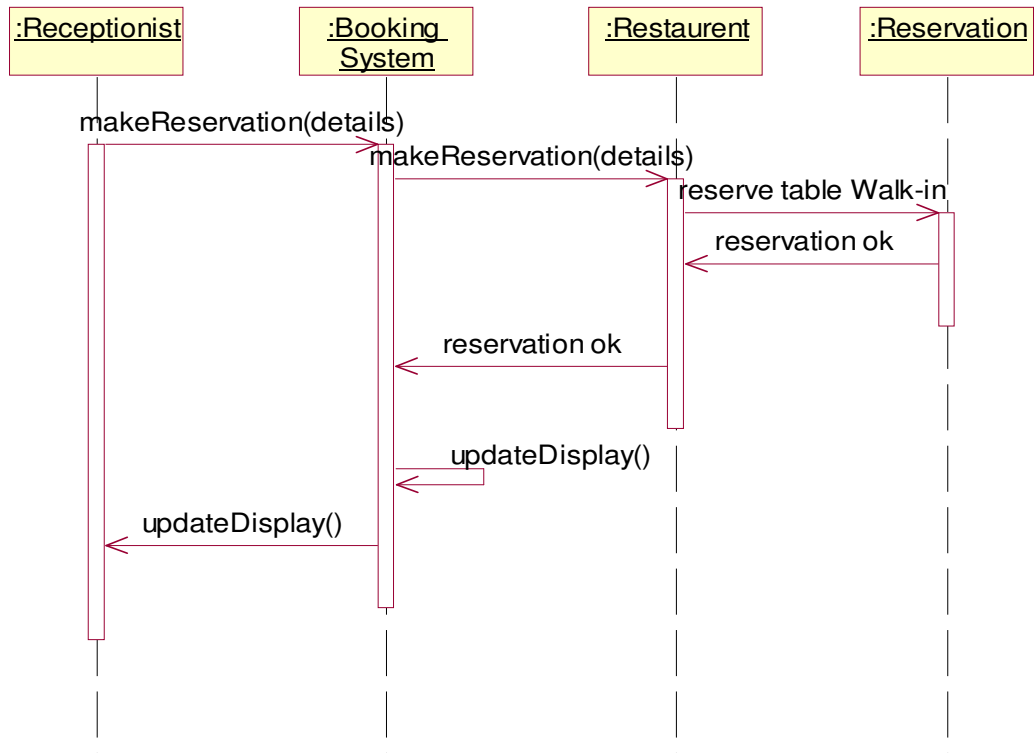
6.2.9 Sequence Diagram for *Table Transfer* Use-case



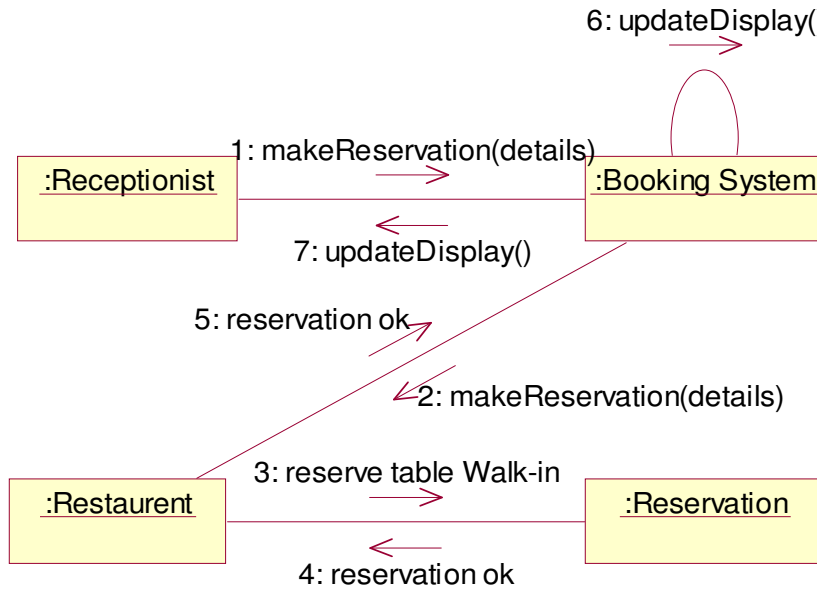
6.2.10 Collaboration Diagram for *Table Transfer* Use-case



6.2.11 Sequence Diagram for *Record Walk-in* Use-case

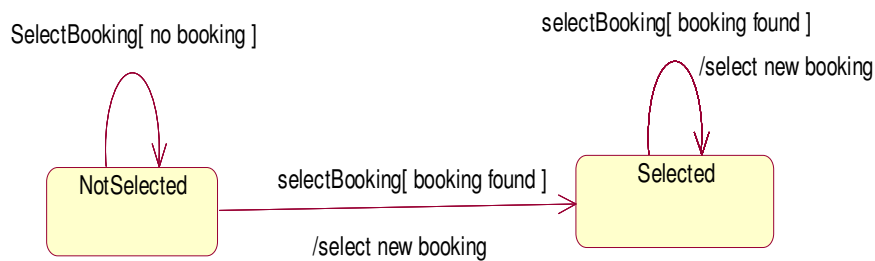


6.2.12 Collaboration Diagram for *Record Walk-in* Use-case

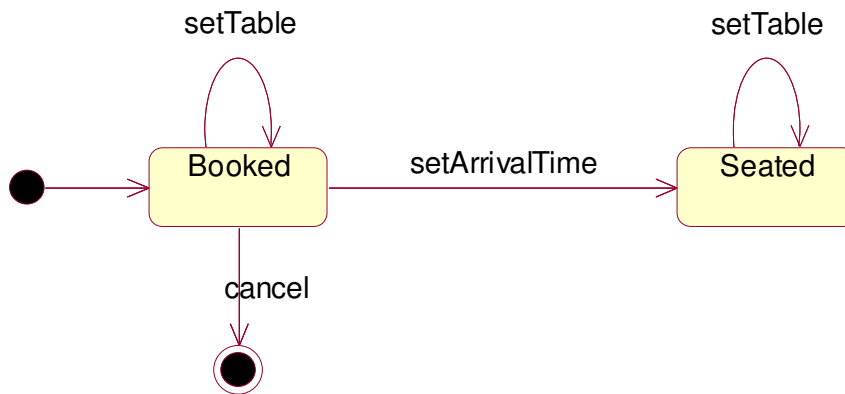


6.3 State chart Diagrams

6.3.1 State chart diagram for *Booking* Class

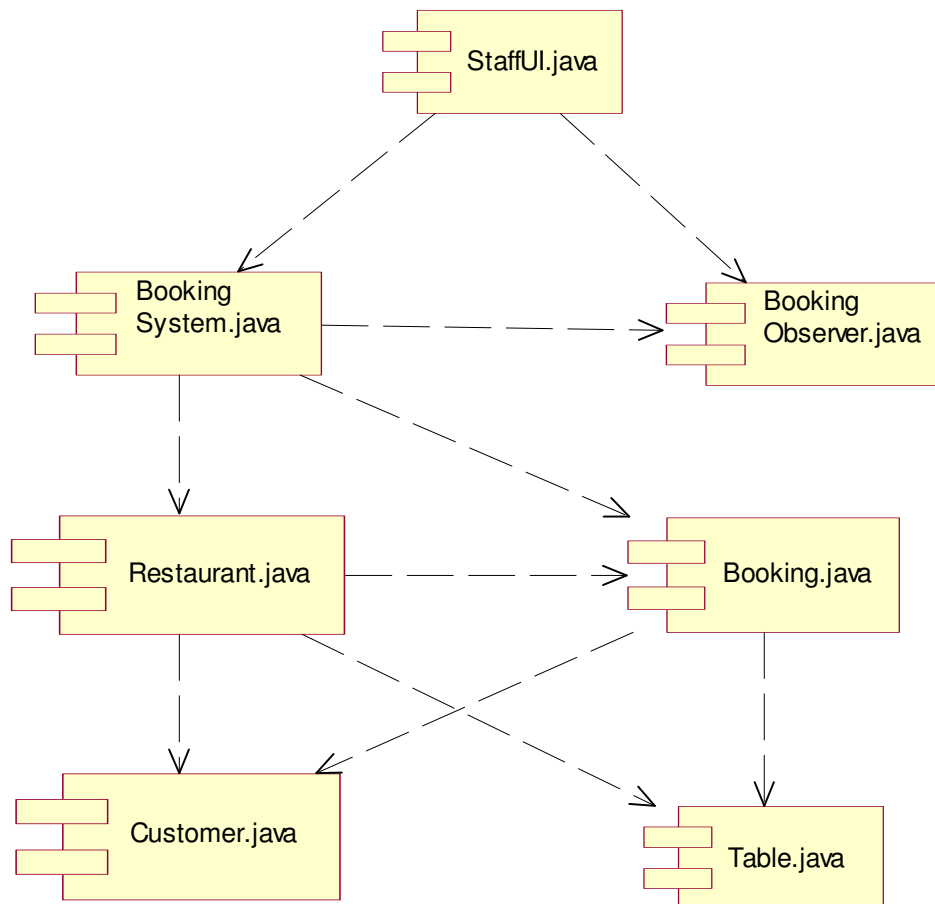


6.3.1 State chart diagram for *Reservation* Class

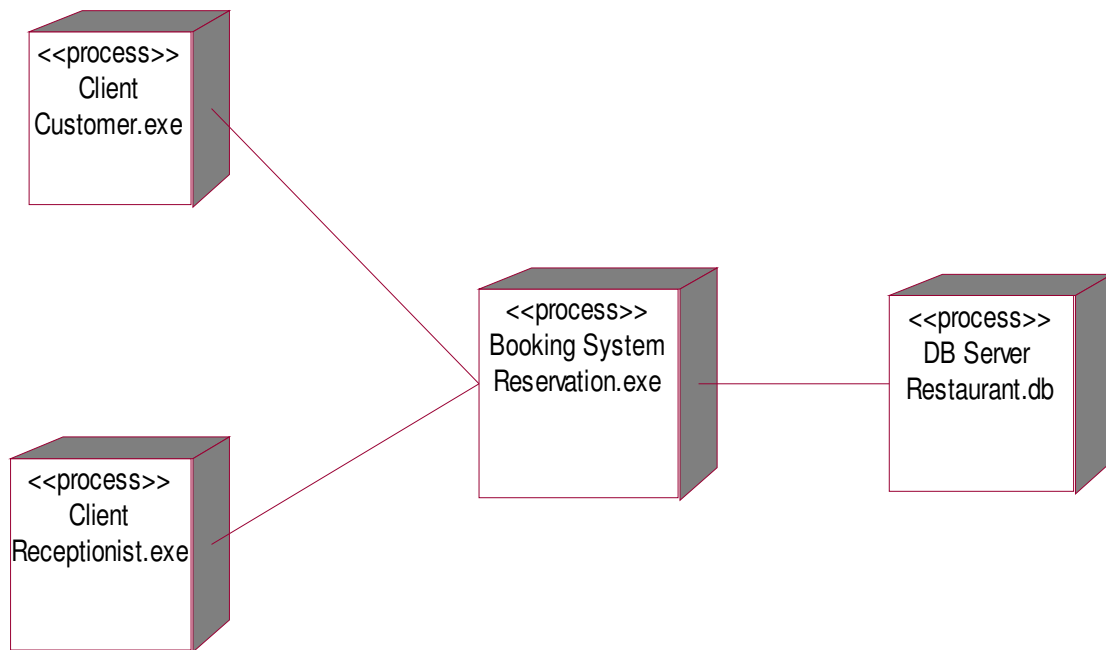


7. Deployment Model

7.1 Component Diagram



7.2 Deployment Diagram



Viva – Voce Questions

1. How many structural and behavioral diagrams are there in UML?
Ans: 4- Structural and 5-behavioral diagrams
2. What is used to create new building blocks in UML?
Ans: stereotypes
3. A use case is rendered as
Ans: As an ellipse
4. The names of use cases are generally given as
Ans: Verb phases
5. Actors are connected to use cases only by
Ans: Association relationship
6. The behavior of a use case is specified by
Ans: Flow of events
7. Extension scenarios are also called as
Ans: Alternative Flows
8. What diagram is used to model the requirements of a system
Ans: Use Case Diagram
9. In UML signals are modeled as
Ans: Stereo Typed classes
10. What diagram is used to model the vocabulary of a system
Ans: Class diagram
11. Internal event occurs when
Ans: A method is invoked via a message
12. Graphically, a component is rendered as
Ans: A Rectangle with tabs
13. What diagram is used to model a physical database
Ans: Component Diagram
14. What is used to specify the properties of UML building blocks?
Ans: Tagged values
15. What diagram is used to model logical database schema
Ans: Class diagram