R20

Unified Modeling Language

Lab Manual

**Department of CSE**

**Vasireddy Venkatadri Institute of Technology**

Nambur(V), Peda kakani(M), Guntur Dt.- 522508

****

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

|  |
| --- |
| **Vision of the Department** |
| To facilitate quality education by focusing on assimilation, generation and dissemination of knowledge in the area of Computer Science & Engineering to transform students into socially responsible engineers. |

|  |
| --- |
| **Mission of the Department** |
| * Equip our graduates with the knowledge by student centric teaching-learning process and expertise to contribute significantly to the software industry and to continue to grow professionally. * To train socially responsible, disciplined engineers who work with good leadership skills and can contribute for nation building. * To make our graduates aware of cutting edge technologies and make them industry-ready engineers. * To shape the department into a center of academic and research excellence. |

|  |  |
| --- | --- |
| **Program Educational Objectives** | |
| PEO-1 | To provide the graduates with ***solid foundation in Computer Science and Engineering*** along with the fundamentals of Mathematics and Sciences with a view to impart in them high quality technical skills like modeling, analyzing, designing, programming and implementation with global competence and helps the graduates for life-long learning. |
| PEO-2 | To prepare and motivate graduates with ***recent technological developments related to core subjects*** like Programming, Databases, Design of Compilers and Network Security aspects and future technologies so as to contribute effectively for Research & Development by participating in professional activities like publishing and seeking copy rights. |
| PEO-3 | To train graduates to choose a ***decent career option either in high degree of employability/Entrepreneur or, in higher education*** by empowering students with ethical administrative acumen, ability to handle critical situations and training to excel in competitive examinations. |
| PEO-4 | To train the graduates to have ***basic interpersonal skills and sense of social responsibility*** that paves them a way to become good team members and leaders. |

**VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY, NAMBUR**

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**Course : B. Tech**

**Branch : CSE**

**Regulation : R20**

**Code :**

**Lab Name : UML Lab**

**Internal Lab Marks : 15**

**External Lab Marks : 35**

**Total Marks : 50**

EVALUTION STRATERGY

**INTERNAL EXAMINATION**

**MAX MARKS: 15**

**SCHEME OF EVALUATION**

**CONTINUOUS EVALUATION : 5M**

* ATTENDANCE : 3M
* DISCIPLINE : 1M
* CLASS-ROOM/QA PERFORMANCE : 1M
* RECORD : 5M

**EXAM-TIME PERFORMANCE : 5M**

**EXTERNAL EXAMINATION**

**MAX MARKS: 35**

**SCHEME OF EVALUATION**

* D’M #1 FOR SYSTEM #1 IN UML WITH DESCRITPTION :10M
* D’M #2 FOR SYSTEM #2 IN UML WITH DESCRITPTION :10M
* DESCRIPTION & UML NOTATION

OF GIVEN DESIGN PATTERN :10M

* VIVA-VOCE :5M

MODEL POSTING OF MARKS ON THE SCRIPT: 10+10+10+5=35

**Course Objectives:**

Students who complete this course will be able to:

* To make the student learn the way of design various application using UML & DP in the

real world.

* Construct UML diagrams for static view and dynamic view of the system.
* Generate creational patterns by applicable patterns for given context.
* Create refined model for given Scenario using structural patterns.
* Construct behavioral patterns for given applications
* Create a Class Diagram using the Reverse Engineering feature in visual paradigm Software Development Platform
* Create a Class Diagrams using the Design Patterns feature in visual paradigm Software Development Platform
* Generate Java code from a Class Diagram in visual paradigm Development Platform
* Modify and run a Java application in visual paradigm Software Development Platform

**Course Outcomes:**

|  |  |  |
| --- | --- | --- |
| UML Lab | | |
| C416 | Subject Code: | Year/Semester: III B.Tech./ II |
| C416.1 | Understanding the industry standard process including Rational unified process and Agile process. | |
| C416.2 | Design the software development process the fits the complexity of solving projects | |
| C416.3 | Applying design patterns to refine models. | |
| C416.4 | Will be able to differentiate how the object-oriented approach differs from the traditional approach to systems analysis and design. | |
| C416.5 | Recognize the difference between various object relationships: inheritance, association, whole-part, and dependency relationships. | |
| C416.6 | Construct various UML models (including use case diagrams, class diagrams, interaction diagrams, state chart diagrams, activity diagrams, and implementation diagrams) using the appropriate notation for different case studies. | |

**CO/ PO, PSO Mapping:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CO/ PO | PO1 | PO2 | PO3 | PO4 | PO5 | PO6 | PO7 | PO8 | PO9 | PO10 | PO11 | PO12 | PSO1 | PSO2 |
| C416.1 | 1 | 2 | 1 | 1 | 1 | 3 | 1 | - | 2 | 1 | 2 | - | 1 | 1 |
| C416.2 | - | 2 | 3 | 1 | 2 | 2 | 1 | - | 1 | 2 | 3 | - | - | - |
| C416.3 | - | - | 1 | - | 3 | 2 | 1 | - | 2 | 1 | 2 | - | 1 | 2 |
| C416.4 | 1 | 1 | 2 | 2 | 1 | 3 | 3 | - | - | 2 | 3 | - | 2 | 1 |
| C416.5 | - | - | 1 | - | - | - | - | - | 2 | - | - | - | 1 | - |
| C416.6 | - | 1 | 2 | - | - | 2 | - | - | 1 | 1 | - | - | 1 | 1 |

**LIST OF CONTENTS**

**Experiment 1**

Familiarization with Rational Rose or Umbrella

**Experiment 2**

1. Identify and analyze events
2. Identify Use cases
3. Develop event table

**Experiment 3**

1. Identify & analyze domain classes
2. Represent use cases and a domain class diagram using Rational Rose
3. Develop CRUD matrix to represent relationships between use cases and problem domain classes

**Experiment 4**

1. Develop Use case diagrams
2. Develop elaborate Use case descriptions & scenarios
3. Develop prototypes (without functionality)

**Experiment 5**

1. Develop system sequence diagrams Develop high-level sequence diagrams for each use case
2. Identify MVC classes / objects for each use case
3. Develop Detailed Sequence Diagrams / Communication diagrams for each use case showing interactions among all the three-layer objects

**Experiment 6**

a) Develop detailed design class model (use GRASP patterns for responsibility assignment)

1. Develop three-layer package diagrams for each case study

**Experiment 7**

1. Develop Use case Packages
2. Develop component diagrams
3. Identify relationships between use cases and represent them
4. Refine domain class model by showing all the associations among classes

**Experiment 8**

1. Develop sample diagrams for other UML diagrams - state chart diagrams, activity Diagrams and deployment diagrams.

**INTRODUCTION TO UML**

**Definition of UML**

The Unified Modeling Language (UML) is used to specify, visualize, modify, construct and document the artifacts of an object-oriented software intensive system under development. UML offers a standard way to visualize a system's architectural blueprints, including elements such as:

* actors
* business processes
* (logical) components
* activities
* programming language statements
* database schemas, and
* reusable software components

The UML represents the culmination of best practical object oriented modeling. The UML is the product of several years of hardworking which we focused on bringing about a unification of the methods used around the world, the adoption of good ideas from many quarters of the industry and above all, effort to make things simple.

**Goals of UML**

The primary goals in the design of the UML were:

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

**SCOPE OF UML:**

The Unified Modeling Language (UML) is a graphical language for visualizing, specifying, Constructing and documenting the aircrafts of a software intensive system. The UML specifies a modeling language that incorporates the object oriented communities consensus on core modeling concepts. It allows deviations to be expressed in terms of extension mechanisms.

The UML provides the following:

* Semantics and notation to address a wide variety of contemporary modeling issues in a direct and economical fashion.
* Semantics to address certain expected future modeling issues, specifically related to component technology, distributed computing, frame works and excitability.
* Extensibility mechanisms so individual projects can extend the Meta model for their application at low cost.
* Extensibility mechanisms so that future modeling approaches could be grown on the top of the UML.
* Semantics to facilitate the model interchange among variety of tools.
* Semantic to specify the interface to repositories for the sharing and storage of model aircrafts.

There are several new components included in UML:

* Extensibility mechanisms
* Threads and process
* Distribution and concurrency
* Patterns / collaboration
* Activity diagrams
* Refinement
* Interface and components
* A constraint language

**UML modeling**

You can draw all kinds of UML 2.x diagrams in VP-UML, which include:

* Class diagram
* Use case diagram
* Sequence diagram
* Communication diagram
* State machine diagram
* Activity diagram
* Component diagram
* Deployment diagram
* Package diagram
* Object diagram
* Composite structure diagram
* Timing diagram
* Interaction overview diagram

Requirement modeling

Capture requirements with SysML Requirement Diagram, Use Case Modeling, Textual Analysis, CRC Cards, and create screen mock-up with User Interface designer.

Database modeling

You can draw the following kinds of diagrams to aid in database modeling:

* Entity Relationship Diagram
* ORM Diagram (visualize the mapping between object model and data model)

You can model not only database table, but also stored procedure, triggers, sequence and database view in an ERD.

Besides drawing a diagram from scratch, you can reverse engineer a diagram from an existing database.

Apart from diagramming, you can also synchronize between class diagram and entity relationship diagram to maintain the consistency between them.

SQL generation an execution feature is available for producing and executing SQL statement from model instantly.

Business process modeling

You can draw the following kinds of diagrams to aid in business process modeling:

* Business process diagram
* Data flow diagram
* Event-drive process chain diagram
* Process map diagram
* Organization Chart

You can also export Business process diagram to BPEL.

Object-Relational mapping

Object-Relational Mapping enables you to access relational database in an object relational approach when coding. VP-UML generates object-relational mapping layer which incorporates features such as transaction support, pluggable cache layer, connection pool and customizable SQL statement.

Team collaboration

For users that work as a team, team collaboration support lets you perform modeling collaboratively and concurrently with any one of the following tools or technologies:

* VP Teamwork Server (Need to buy Visual Paradigm Teamwork Server additionally)
* CVS
* Subversion
* Perforce

Interoperability: The interoperability support allows you to exchange model data with other tools.

## Things

**UML Building Blocks**

**Structural things** define the static part of the model. They represent the physical and conceptual elements. Following are the brief descriptions of the structural things.

* 1. **Class −** Class represents a set of objects having similar responsibilities.



* 1. **Interface −** Interface defines a set of operations, which specify the responsibility of a class.



* 1. **Collaboration −**Collaboration defines an interaction between elements.



* 1. **Use case −**Use case represents a set of actions performed by a system for a specific goal.



* 1. **Component −**Component describes the physical part of a system.



* 1. **Node −** A node can be defined as a physical element that exists at run time.



**Behavioral Things** consists of the dynamic parts of UML models

1. **Interaction −** Interaction is defined as a behavior that consists of a group of messages exchanged among elements to accomplish a specific task.



1. **State machine −** State machine is useful when the state of an object in its life cycle is important. It defines the sequence of states an object goes through in response to events. Events are external factors responsible for state change



**Grouping things** can be defined as a mechanism to group elements of a UML model together.

1. **Package −** Package is the only one grouping thing available for gathering structural and behavioral things.



**Annotational things** can be defined as a mechanism to capture remarks, descriptions, and comments of UML model elements.

* 1. **Note** - It is the only one Annotational thing available. A note is used to render comments, constraints, etc. of an UML element.



1. **Relationship**

**Relationship** is another most important building block of UML. It shows how the elements are associated with each other and this association describes the functionality of an application.

* 1. **Dependency** is a relationship between two things in which change in one element also affects the other.



* 1. **Association** is basically a set of links that connects the elements of a UML model. It also describes how many objects are taking part in that relationship.



* 1. **Generalization** can be defined as a relationship which connects a specialized element with a generalized element. It basically describes the inheritance relationship in the world of objects.



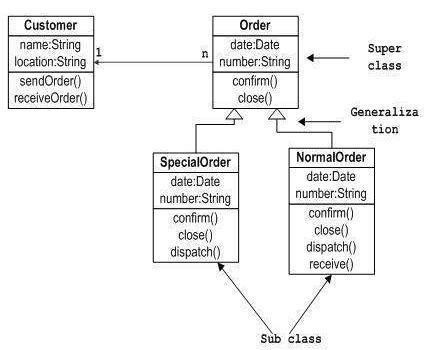
* 1. **Realization** can be defined as a relationship in which two elements are connected. One element describes some responsibility, which is not implemented and the other one implements them. This relationship exists in case of interfaces.



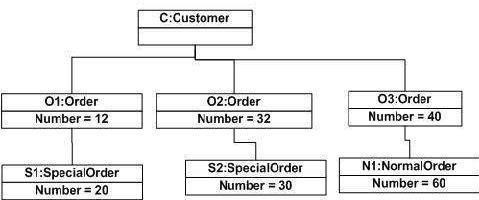
1. **UML Diagrams**

**Structural Diagrams** represent the static aspect of the system. These static aspects represent those parts of a diagram, which forms the main structure and are therefore stable.

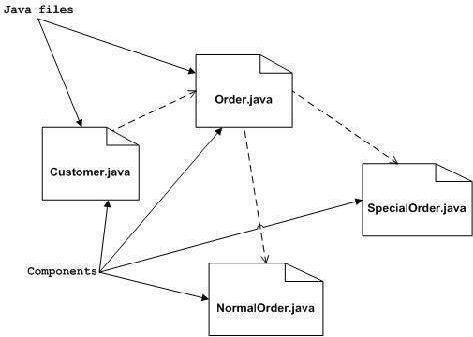
1. **Class diagrams** consist of classes, interfaces, associations, and collaboration.



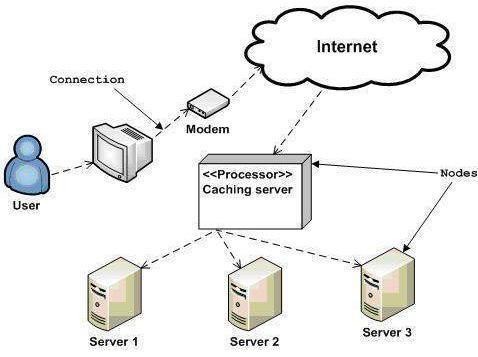
1. **Object diagrams** are a set of objects and their relationship is just like class diagrams.



1. **Component diagrams** represent a set of components and their relationships.

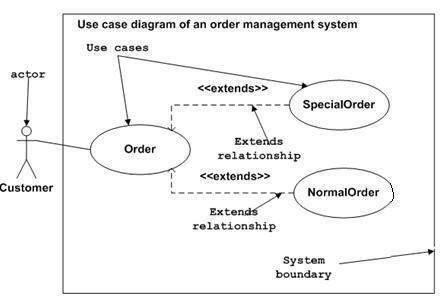


1. **Deployment diagrams** are a set of nodes and their relationships. These nodes are physical entities where the components are deployed.

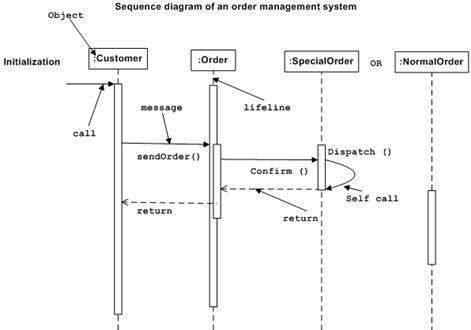


**Behavioral Diagrams** basically capture the dynamic aspect of a system. Dynamic aspect can be further described as the changing/moving parts of a system.

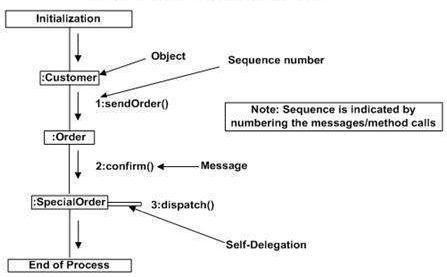
1. **Use case Diagrams** are a set of use cases, actors, and their relationships. They represent the use case view of a system.



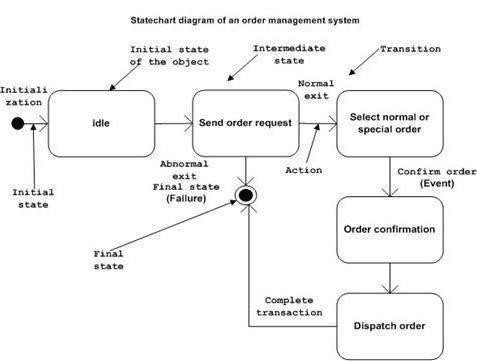
1. **Sequence diagram** deals with some sequences, which are the sequence of messages flowing from one object to another.

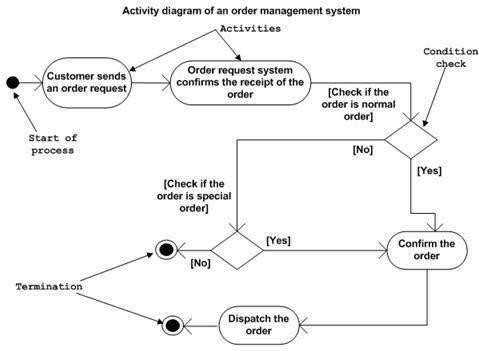


1. **Collaboration diagram** represents the structural organization of a system and the messages sent/received. Structural organization consists of objects and links.



1. **State-chart Diagram** 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.



1. **Activity diagram** describes the flow of control in a system. It consists of activities and links

## Problem Statements

**Case Studies**

## Library Management System

* It is a support system for a library.
* The library lends books and magazines to borrowers, who are registered in the system, as are the books and magazines.
* The library handles the purchase of new titles for the library. Popular titles are bought in multiple copies. Old books and magazines are removed when they are out of date or in poor condition.
* The librarian is an employee of the library who interacts with the customers (borrowers) and whose work is supported by the system.
* A borrower can reserve a book or magazine that is not currently available in the library, so that when it’s returned or purchased by the library, that borrower is notified. The reservation is canceled when the borrower checks out the book or magazine or through an explicit canceling procedure.
* The librarian can easily create, update, and delete information about the titles, borrowers, loans, and reservations in the system.
* The system can run on all popular Web browser platforms (Internet Explorer 5.1+, Netscape 4.0+, and so on).
* The system is easy to extend with new functionality.

## Point-Of-Sale Terminal

* POS system is a computerized application used to record sales and handle payments; it is typically used in a retail store. It includes hardware components such as a computer and bar code scanner, and software to run the system.
* It interfaces to various service applications, such as a third-party tax calculator and inventory control. These systems must be relatively fault- tolerant; that is, even if remote services are temporarily unavailable, they must still be capable of capturing sales and handling at least cash payments.
* POS system increasingly must support multiple and varied client-side terminals and interfaces. These include a thin-client Web browser terminal, a regular personal computer with something like a Java Swing graphical user interface, touch screen input, wireless PDAs, and so forth.
* Furthermore, we are creating a commercial POS system that we will sell to different clients with disparate needs in terms of business rule processing.
  + - Each client will desire a unique set of logic to execute at certain predictable points in scenarios of using the system, such as when a new sale is initiated or when a new line item is added.

# Customer Support System

# The objective of the system is to integrate all front-office sales and customer related activities.

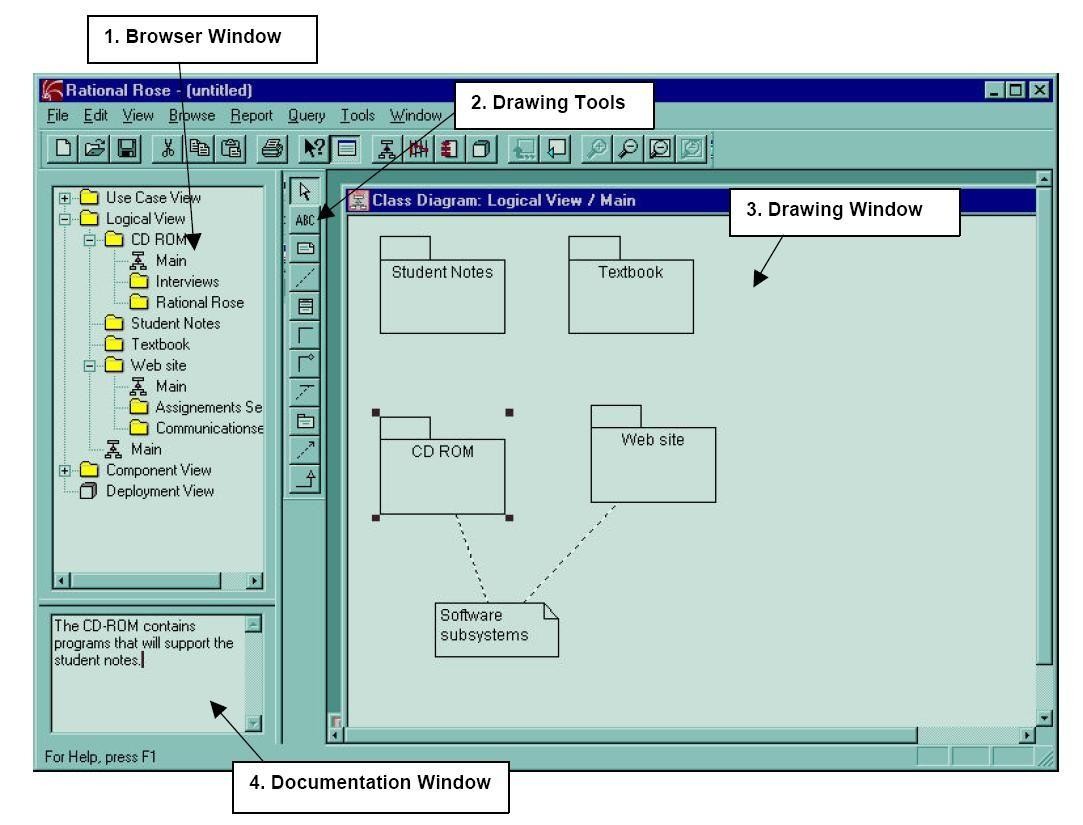
# There is a need for a comprehensive sales and customer support system to provide integrated support for all sales channels.

# To operate high levels of efficiency and to take advantage of market opportunities, the company needs a new comprehensive system that is tightly integrated with the SCM system. This new system will be called the customer support system.

# 

# Experiment1: Familiarization with Rational Rose or Umbrella

Rational Rose is an object-oriented Unified Modeling Language (UML) software design tool intended for visual modeling and component construction of enterprise-level software applications. Two popular features of Rational Rose are its ability to provide iterative development and round-trip engineering.



## 

## 1.Browser

Browser contains a list of all the modeling elements in the current model. It contains a tree view of all the elements in the current Rose model. This presents a hierarchical view of the analysis and design model, including all the diagrams and all the individual elements that make up a diagram.

## 

## Documentation Window

Documentation Window may be used to create, review, and modify for any selected modeling element. If nothing is selected, the window displays the string “No selection”. The contents of the Documentation Window will change as different modeling elements are selected. It is strongly recommended that each element added to a diagram have documentation to accompany it. To add documentation, right click on the element, select specification, and fill in the documentation field.The documentation will then be shown in the documentation window each time the mouse is clicked on the element.

## Diagram Windows

Diagram windows allow creating and modifying graphical views of the current model. Each icon on a diagram represents a modeling element. This is the place where the diagram is actually created.

## 

## Drawing Tools

This tool presents a set of icons that indicate the different elements that can be added to a diagram. The elements that can be used will change, depending on the type of diagram being created. Different diagram types have different sets of icons.

## Views in Rational Rose

There are four views for a model created in Rational Rose, each representing the system from a different point of view.

## 

## Use Case View

The use case view contains the diagrams used in analysis, and all the elements that contain these diagrams. This view looks at actors and use cases along with their interactions. The diagrams in this view are use case diagrams, sequence diagrams and collaboration diagrams. The purpose of the use case view is to visualize what the system must do, without dealing with the specifics of how it will be implemented. More recent versions of Rational Rose also allow for additional documentation in the form of word-processed documents and/or URLs to Web- based materials.

Packages in the use case view can contain actors, use cases, sequence diagrams, and/or collaboration diagrams. To create a package:

* + Right-click on the parent modeling element (use case view or another package) to make the shortcut menu visible.
  + Select the New: Package menu command. This will add a new package called New Package to the browser.
  + While the new package is still selected, enter its name.

Once a package is created, modeling elements may be moved to the package.

To move a modeling element to a package:

* + Click to select the modeling element to be relocated.
  + Drag the modeling element to the package.

## 

## Logical View

The logical view contains the diagrams used in object design. The diagrams in this view are class diagrams and state transition diagrams. It offers a detailed view of how the system visualized in the use case view will be implemented. The basic element in this view is the class, which includes an outline of its attributes and operations. This directly corresponds to a class created in our chosen implementation language.

From the logical view, skeletal code can be generated for implementation into a computer language. More recent versions of Rational Rose not only can generate skeletal code for Visual C++, Visual Java, or Visual BASIC, but also reverse engineer programs created in these languages into Rational Rose models. This allows existing components to be included in documented models, if there is access to the source code. In addition, changes that need to be made during implementation can be reflected in the documentation of the design model.

## 

## Component View

The component view is a step up from the logical view and contains diagrams used in system design. This view contains only component diagram. This includes information about the code libraries, executable programs, runtime libraries, and other software components that comprise the completed systems.

Components can be pre-existing; for example, a Windows program in Visual C++ will utilize Microsoft Foundation Class to provide the framework for the Windows interface. Components that do not exist and need to be created by the developers will have to be designed in the logical view.

Rose automatically creates one component diagram called Main. To create an additional component diagram:

* + Right-click on the owning package (either the component view itself or a user created package) to make the shortcut menu visible.
  + Select the New: Component Diagram menu command. This will place a new component diagram called New Diagram in the browser.
  + While the new diagram is still selected, enter its name. Rose will automatically add the new diagram to the browser.

## To open a component diagram:

* + - * Double-click on the diagram in the browser.

## To create a component:

* + - * Click to select the package specification icon from the toolbar.
      * Click on the component diagram to place the component.
      * While the component is still selected, enter its name. Rose will automatically add the new component to the browser.

## 

## To create a dependency relationship:

* + - * Click to select the dependency icon from the toolbar.
      * Click on the package or component representing the client.
      * Drag dependency arrow to the package or component representing the supplier.

## Deployment View

The deployment view illustrates how the completed system will be physically deployed. This view contains only deployment diagram.

**Experiment 2**

## Identify and analyze events

Event is an action or occurrence recognized by software that may be handled by the software. Computer events can be generated or triggered by the system, by the user or in other ways. A source of events includes the user, who may interact with the software by way of, for example, keystrokes on the keyboard. Another source is a hardware device such as a timer. Software can also trigger its own set of events into the event loop, e.g. to communicate the completion of a task. Software that changes its behavior in response to events is said to be event- driven, often with the goal of being interactive. The first step in use-case modeling is to define actors who interact with the system and their events.

## Library Management System:

The actors in the library are identified as follows:

* + **Librarian actor**: is user of the system and have management capability to add borrowers, titles, and items.

## Events of Librarian

1. Login to the system
2. Manage (create, update, delete) Borrowers
3. Manage (create, update, delete) Titles (Books or Magazines)
4. Manage (create, update, delete) Items
5. Search Titles
6. Browse Titles
7. Assume Identity of Borrower
   * **Borrowers actor**: is also user of the system and Borrower is people who check out and reserve books and magazines. Occasionally a librarian or another library can be a borrower.

## Events of Borrower

1. Register in the system
2. Login to the system
3. Make reservation of Titles (Books or Magazines)
4. Remove reservation
5. Search Titles
6. Browse Titles
7. Checkout Titles
8. Return Titles
   * **Master Librarian actor**: this role is capable of managing the librarians as well. It is possible to add a title to the system before the library has a copy (an item), to enable borrowers to make reservations.

## Events of Master Librarian

1. Login to the system
2. Manage Librarians
3. Add Titles (Books or Magazines)

## Point-Of-Sale Terminal:

There are three types of actors can be identified as follows:

* + **Primary actor:** has user goals fulfilled through using services. Find user goals to drive the use cases. Ex: Cashier, System Administrator

## Events of Cashier

1. Process Sale
2. Handle Returns
3. Process Rental
4. Cash In

## Events of System Administrator

1. Manage Security
2. Manage Users
   * **Supporting actor**: provides a service (e.g., information). Often a computer system, but could be an organization or person. The purpose is to clarify external interfaces and protocols. Ex: Sales Activity System

## Events of Sales Activity System

1. Analyze Activity

* + **Offstage actor**: has an interest in the behavior of the use case, but is not primary or supporting. Ex: Payment Authorization Service like Tax Calculator, Accounting System, HR System

## Events of Tax Calculator

1. Process Sale
2. Process Rental

## Events of Accounting System

1. Process Sale
2. Process Rental

## Events of HR System

1. Cash In

## Identify Use cases

Next step in use-case modeling is to define Use-cases of system that describe what the library system provides in terms of functionality to the actors and the functional requirements of the system. While identifying the use cases, we must read and analyze all specifications, as well as discuss the system with potential users of the system and all stakeholders.

## Library Management System:

The use cases in the library management system are as follows:

* + Login
  + Search
  + Browse
  + Make Reservation
  + Remove Reservation
  + Checkout Item
  + Return Item
  + Manage Titles
  + Manage Items
  + Manage Borrowers
  + Manage Librarians
  + Assume Identity of Borrower

## Point-Of-Sale Terminal:

The use cases in the POS terminal system are as follows:

* + Login
  + Process Sale
  + Handle Returns
  + Process Rental
  + Cash In
  + Analyze Activity
  + Manage Security
  + Manage Users

## Develop event table

An event table is a catalogue of use cases that lists events in rows and key pieces of information about each event in columns. It includes row and columns representing events and their details, respectively. Each column in the event table represents a section as follows:

* + **Trigger**: a signal that tells the system that an event has occurred, either the arrival of data needing processing or a point in time.
  + **Source**: an external agent or actor that supplies data to the system
  + **Response**: an output, produced by the system, that goes to a destination
  + **Destination**: an external agent or actor that receives data from the system

## Library Management System:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Event / Use case** | **Trigger** | **Source** | **Response** | **Destination** |
| Login to the system | Login | Registered User | Authentication of the user | Registered User |
| Search Titles | Search | Registered User | Showing searched details about Titles  using keywords | Registered User |
| Browse Titles | Browse | Registered User | Showing browsed  details about Titles | Registered User |
| Manage Borrowers | Create / Update/ Delete  Borrower | Librarian | Insertion / Updation / Deletion of Borrower details | Librarian |
| Manage Titles | Create /  Update/ Delete Titles | Librarian | Insertion / Updation  / Deletion of Title details | Librarian |
| Manage Items | Create / Update/  Delete Items | Librarian | Insertion / Updation  / Deletion of Item details | Librarian |
| Assume  Identity of Borrower | Identify | Librarian | Getting identify of the borrower | Librarian |
| Make  reservation of Titles | Make  Reservation | Borrower | Adding reservation  record for particular borrower | Borrower |
| Remove Reservation | Remove Reservation | Borrower | Removing reservation record for particular  borrower | Borrower |
| Checkout Titles | Checkout | Borrower | Showing reservation  status of particular borrower | Borrower |
| Return Titles | Return | Borrower | Updation of loan details | Borrower |
| Manage Librarians | Create / Update/  Delete Librarians | Master Librarian | Insertion / Updation/ Deletion of Librarian details | Master Librarian |
| Add Titles | Add Title | Master  Librarian | Insertion of new Title details | Master  Librarian |
| Enabling borrowers to make  Reservation | Enable reservation | Master Librarian | Enabling of make reservation option for borrower | Master Librarian |

1. **Point-Of-Sale Terminal:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Event / Use case** | **Trigger** | **Source** | **Response** | **Destination** |
| Process Sales | Process Sale | Tax Calculator / Accounting System | Generation of sales / inventory bills | Cashier |
| Handle Returns | Handle  Returns | Cashier | Recording of returned  items by the Cashier | Cashier |
| Process Rental | Process Rental | Tax Calculator / Accounting  System | Generation third party service bills | Cashier |
| Cash In | Cash In | HR System | Receiving ash from HR  System | Cashier |
| Manage Security | Create  /Update/  Delete Security | System Administrator | Insertion / Updation  /Deletion of Security related information | System Administrator |
| Manage Users | Create  /Update/ Delete Users | System Administrator | Insertion / Updation  /Deletion of Users details | System Administrator |
| Analyze Activity | Analyze Activity | Sales Activity System | Showing of analysis of sales activities | Sales Activity System |

**Experiment 3**

**a) Identify & analyze domain classes**

Objects that represent domain entities are called entities or domain objects. The classes from which domain objects can be instantiated are called Domain classes. An early domain model is useful to establish a core set of classes that represent the things in the problem space of the system to be built.

## Library Management System:

The use cases in the library management system are as follows:

* Borrower
* Title
* Book Title
* Magazine Title
* Item
* Reservation
* Loan

## Point-Of-Sale Terminal:

The domain classes in the POS terminal system are as follows:

* Sales Line Item
* Item
* Sale
* Store
* Payment
* Register

## b) Represent use cases and a domain class diagram using Rational Rose

Design class diagrams (DCD) show software class definitions. Classes are shown with their simple attributes and methods listed. Some attributes are depicted using associations (relationships) rather than actually being listed in the class block.

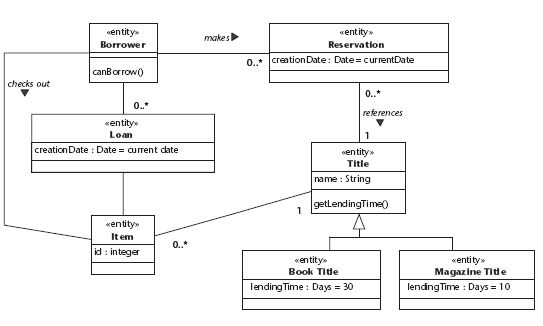
These associated attributes refer to complex objects which should also be shown in the diagram. The collaboration diagram indicates methods to be contained in a class with methods posted as relationships.

## Library Management System:

Representation of domain class

|  |  |  |
| --- | --- | --- |
| **Domain**  **Class** | **Use cases** | **Other Operations** |
| Borrower | Register, Login, Search, Browse, Make Reservation, Remove Reservation, Checkout Item, Return Titles | Get borrowed titles,  Get borrowed items,  Get reservation details |
| Title | Search Title, Browse Title | Get title quantity |
| Book Title | Search Title, Browse Title | Get book quantity |
| Magazine  Title | Search Title, Browse Title | Get magazine quantity |
| Item | Search Item, Browse Item, Checkout  Item | Get item quantity |
| Reservation | Make Reservation, Remove  Reservation | Get reservation details |
| Loan | Checkout Item, Return Titles | Get loan details |

Domain class diagram for the library management system:

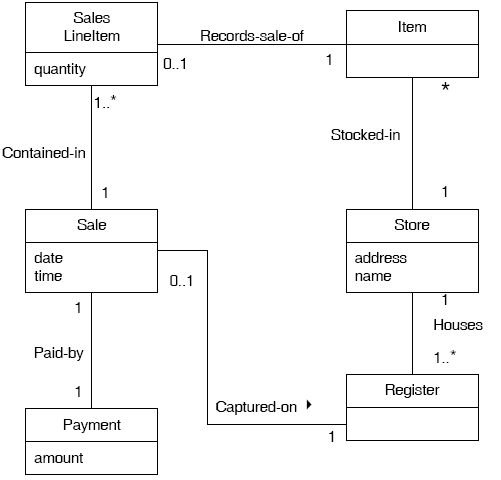


## Point-Of-Sale Terminal:

Representation of domain class with use cases:

|  |  |  |
| --- | --- | --- |
| **Domain Class** | **Use cases** | **Other Operations** |
| Sales Line Item | Process Sale, Handle  Returns | Get sales details, Get item details |
| Item | Handle Returns | Get item quantity, Get item details |
| Sale | Cash In, Process Sale | Get sales details |
| Store | Handle Items | Get store details |
| Payment | Process Rental | Generate bill, Print bill |
| Register | Process Sale | Get sales details, Get store details |

Domain class diagram for POS terminal system



## c) Develop CRUD matrix to represent relationships between use cases and problem domain classes

A CRUD matrix is a table showing the functions in an application that affects parts of a database. The CRUD Matrix is an excellent technique to identify the Tables in a database which are used in any User interaction. CURD means:

* + Create - INSERT - to store new data
  + Read - SELECT - to retrieve data
  + Update - UPDATE - to change or modify data.
  + Delete - DELETE - delete or remove data

CRUD Matrix identifies the Tables involved in any CRUD operation. The analysis helps to identify any Tables which are used.

1. Library Management System:

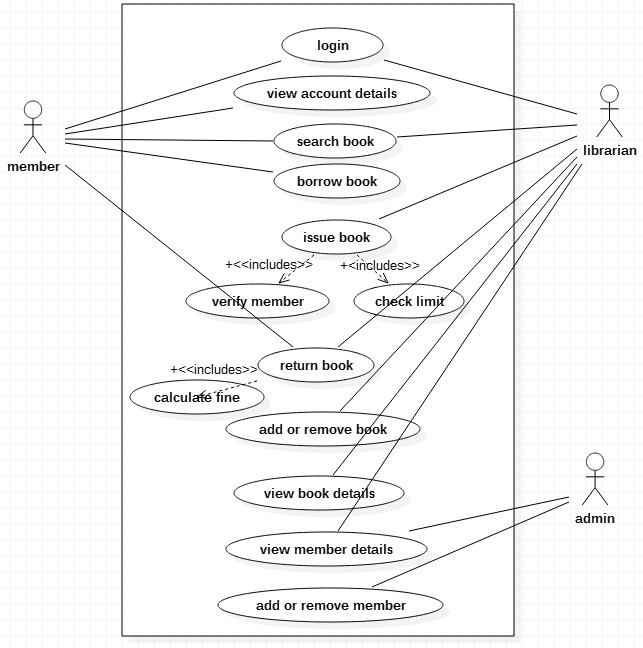
CRUD matrix for library management system:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Event / Use case** | **Borrower** | **Title** | **Item** | **Reservation** | **Loan** |
| Login to the system | R |  |  |  |  |
| Search Titles |  | R |  |  |  |
| Browse Titles |  | R |  |  |  |
| Manage Borrowers | C,U,D |  |  |  |  |
| Manage Titles |  | C,U,D |  |  |  |
| Manage Items |  |  | C,U,D |  |  |
| Assume Identity of Borrower | R |  |  |  |  |
| Make reservation of Titles |  | R |  | C,U |  |
| Remove reservation |  |  |  | D |  |
| Checkout Titles |  | R |  |  |  |
| Return Titles |  | U |  | U | U |
| Manage Librarians |  |  |  |  |  |
| Add Titles |  | C |  |  |  |
| Enabling borrowers to make reservation | U |  |  |  |  |

## 2. Point-Of-Sale Terminal:

CRUD matrix for POS terminal system:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Event / Use case** | **Sales Line Item** | **Item** | **Sale** | **Payment** | **Register** |
| Process Sales | U | R | C,U |  |  |
| Handle Returns | U | U |  |  | U |
| Process Rental |  |  |  | C,U |  |
| Cash In | U |  |  |  |  |
| Manage Security |  |  |  |  |  |
| Manage Users |  |  |  |  |  |
| Analyze Activity | R | R | R | R | R |



**Experiment 4**

# For each case study

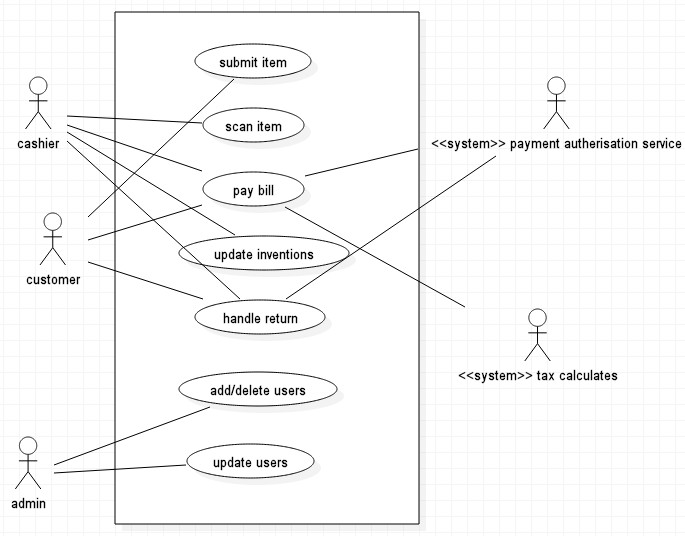
## Develop Use case diagrams

1. **Library Management System:**

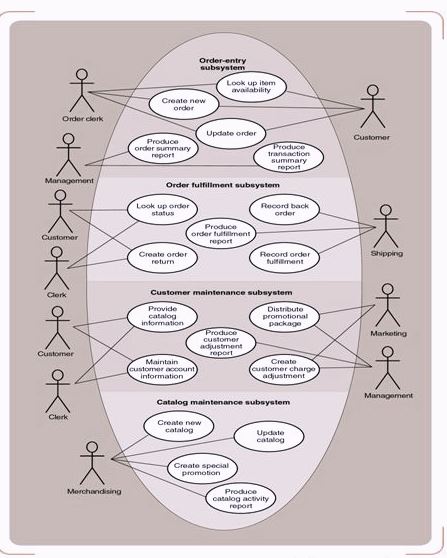
**Use case diagram of library management system**

1. **Point-Of-Sale Terminal:**

**Use case diagram of point of sale terminal**



**3.CUSTOMER SUPPORT SYSTEM USE CASE DIAGRAM**



## Develop elaborate Use case descriptions & scenarios

**Scenario** is a specific sequence of actions and interactions between actors and the system under discussion; it is also called a use case instance. **Use case** is a collection of related success and failure scenarios that describe actors using a system to support a goal.

* + - 1. **Library Management System:**

**Make Reservation Use case along with its scenarios**

|  |  |
| --- | --- |
| **Primary Actor:** | Borrower |
| **Stakeholders & Interests:**   1. Borrower: fast & easy environment for making reservation 2. Librarian: easy management of borrower, reservation & load details 3. Master Librarian: easy management of librarian, borrowers | |
| **Pre-conditions:** | Borrower login, Enable for making reservation |
| **Post-conditions:** | Reservation details for book are saved, Taking necessary actions when book is not available |
| **Main Success Scenarios:**   1. Borrower login to the system 2. Borrower search for the required title 3. Selects one of the books form the list of searched results 4. Selects for make reservation option for the book 5. If book is available book will be issued to the respective borrower 6. Borrower loan details will be updated to indicate issue of book | |
| **Extensions:**   1. At any time, System fails:    * To support recovery and correct accounting, ensure all transaction sensitive state and events can be recovered from any step of the scenario. 2. Invalid borrower credentials:    * System signals error and rejects entry. 3. If selected book is not available:    * Borrower need to check until book is returned by other borrower.    * Borrower will be notified if new book is add to library. | |

* + - 1. **Point-Of-Sale Terminal:**

**Process Sale Use case along with its scenarios**

|  |  |  |
| --- | --- | --- |
| **Primary Actor:** | | Cashier |
| **Stakeholders & Interests:**   1. Cashier: wants accurate, fast entry, and no payment errors 2. Sales Person: wants sales commissions updated 3. Customer: wants purchase and fast service with minimal effort 4. Company: wants accurately record transactions and satisfy customer interests 5. Manager: wants to be able to quickly perform override operations 6. Government Tax agencies: want to collect tax from every sale 7. Payment Authorization Service: want to receive digital authorization request in correct format and protocol | | |
| **Pre-conditions:** | Cashier Identified and Authenticated | |
| **Post-conditions:** | Sale is saved, Tax is correctly calculated, Accounting and Inventory are update,  Commissions recorded, Receipt is generated, Payment authentication approvals are recorded | |
| **Main Success Scenarios:**   1. Customer arrives at POS checkout with items 2. Cashier starts new sale 3. Cashier enters item identifiers 4. System records sales line item & presents item details Cashier repeats steps 3-4 until indicates done. 5. System presents total with taxes and asks for payment 6. Customer pays and system handles payment 7. System logs completed sales and sends payment information to accounting and inventory systems 8. System presents receipt 9. Customer leaves with receipt and items | | |
| **Extensions:**   1. At any time, System fails:    * To support recovery and correct accounting, ensure all transaction sensitive state and events can be recovered from any step of the scenario. 2. Invalid identifier:    * System signals error and rejects entry. 3. System detects failure to communicate with external tax calculation system service:    * System restarts the service on the POS node, and continues.    * Cashier may manually calculate and enter the tax, or cancel the sale. 4. Customer says they intended to pay by cash but don’t have enough cash:    * Customer uses an alternate payment method.    * Customer tells Cashier to cancel sale. Cashier cancels sale on System. | | |

* + - 1. **Customer Supporting System:**

**Add items to cart Use case along with its scenarios**

|  |  |
| --- | --- |
| **Primary Actor:** | Customer |
| **Stakeholders & Interests:**   1. Customer: fast & easy environment for placing orders 2. Company: accurate entry of product, payment, shipment details 3. System Administrator: easy maintenance of product details | |
| **Pre-conditions:** | Customer login |
| **Post conditions:** | Selected items/products are successfully added to cart |
| **Main Success Scenarios:**   1. Customer login to the system 2. Borrower search for the required item/product 3. Selects one of the item/product form the list of searched results 4. Add selected item/product to the cart   Customer repeats 2,3,4 steps until required items/products added to cart   1. System displays list of items/products added to the cart | |

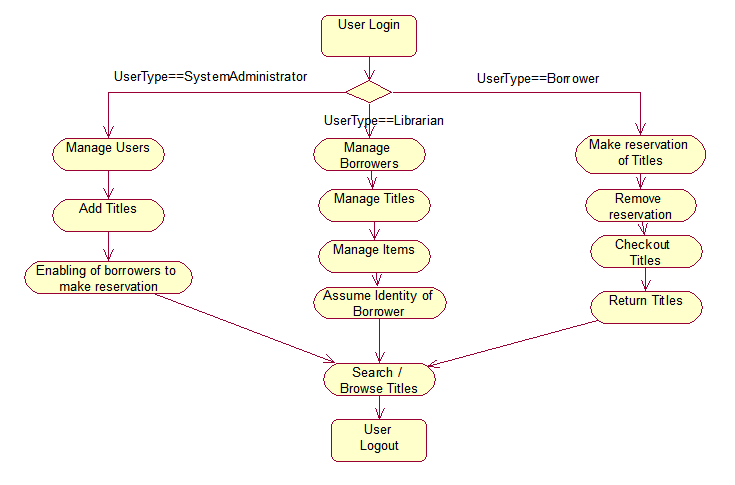
**Extensions:**

1. At any time, System fails:
   * To support recovery and correct accounting, ensure all transaction sensitive state and events can be recovered from any step of the scenario.
2. Invalid customer credentials:
   * System signals error and rejects entry.
3. If customer is unable to get cart details:
   * Customer need to re-login.
   * Customer need to refresh for getting updated cart details.

## Develop prototypes (without functionality)

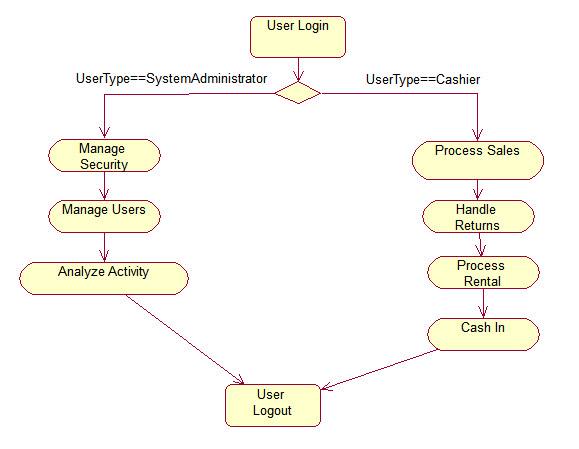
1. **Library Management System:**

**Prototype of LMS**



1. **Point-Of-Sale Terminal:**

**Prototype of POS**



**Experiment 5**

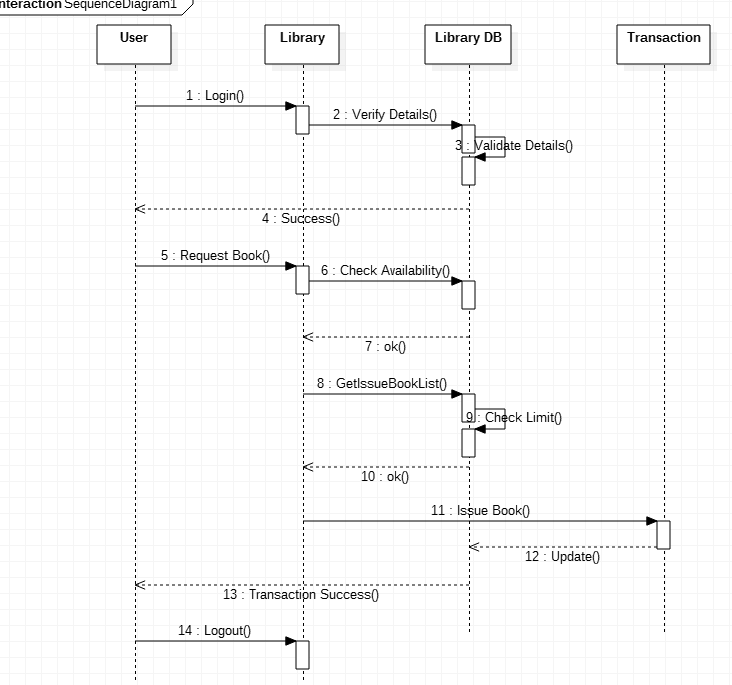
**For each case Study**

## Develop system sequence diagrams

A system sequence diagram (SSD) is a picture that shows, for a particular scenario of a use case, the events that external actors generate their order, and inter-system events. All systems are treated as a black box; the emphasis of the diagram is events that cross the system boundary from actors to systems.

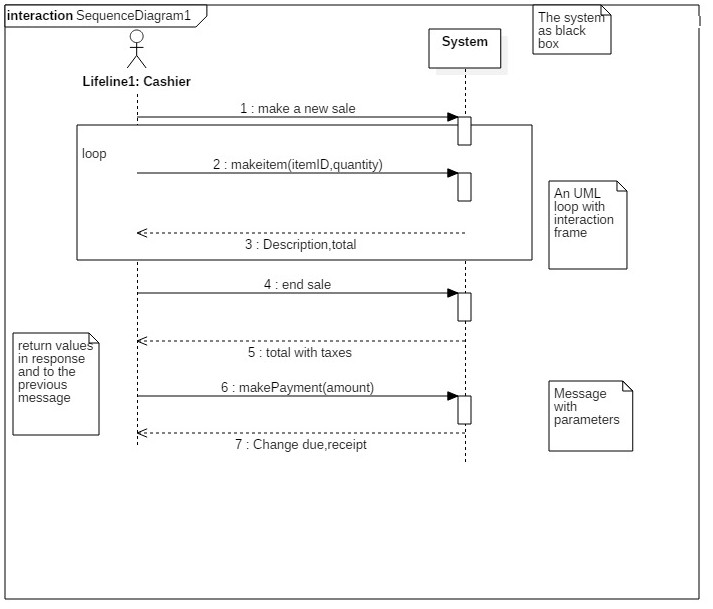
* + - 1. **Library Management System:**

**System sequence diagram for library management system**

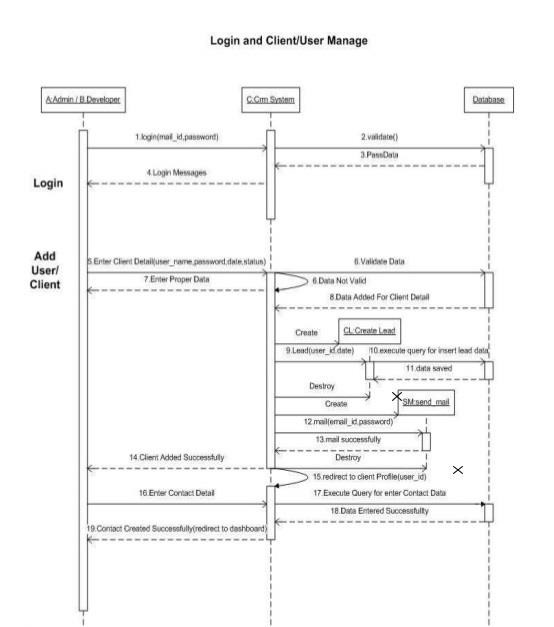


**2.Point-Of-Sale Terminal:**

**System sequence diagram for POS terminal system**



**3.SEQUENCE DIAGRAM FOR CUSTOMER SUPPORT SYSTEM:**



# 

1. **Identify MVC classes / objects for each use case 1. Library Management System:**

**MVC classes/objects of LMS**

|  |  |  |  |
| --- | --- | --- | --- |
| **Use case** | **Model Classes** | **View Classes** | **Controller Classes** |
| Make Reservation | Loan | BorrowerFrame | Reservation |
| Remove Reservation |
| Checkout Item | Titles Items |
| Return Titles/Items |
| Manage Titles/Items | TitlesModel | LibrarianFrame |
| Manage Librarians | UsersModel | Librarian |
| Manage Borrowers | Borrower |
| Assume Identity of  Borrower |

## Point-Of-Sale Terminal:

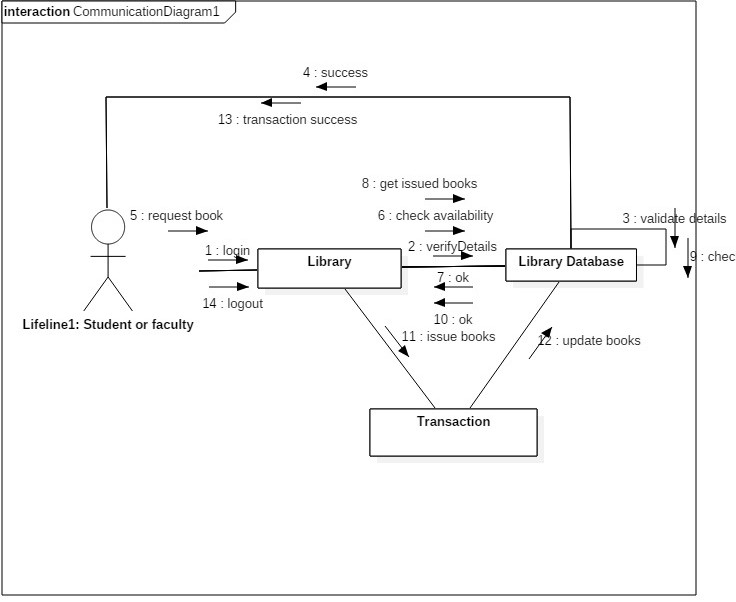
**MVC classes/objects of POS**

|  |  |  |  |
| --- | --- | --- | --- |
| **Use case** | **Model Classes** | **View Classes** | **Controller Classes** |
| Process Sale | SalesModel TaxCalculator  HRSystem Store | ProcessSaleFrame ProcessSaleConsole | Register Sale Payment  SalsLineItem |
| Handle Returns |
| Process Rental |
| Cash In |
| Analyze Activity | ActivityModel | ActivityFrame | Analyze |
| Manage Security | UserModel | LoginFrame | Users |
| Manage Users |
| Login |

## Develop Detailed Sequence Diagrams / Communication diagrams for each use case showing interactions among all the three-layer objects

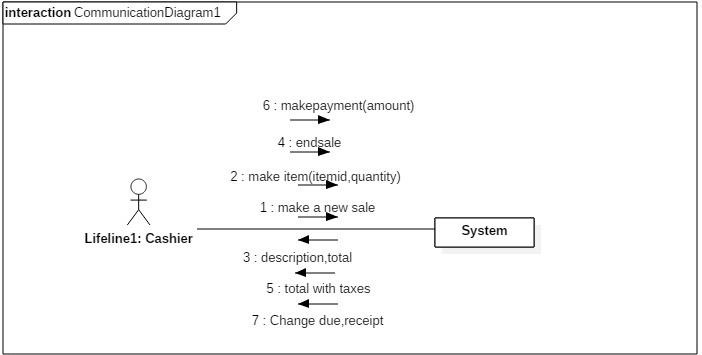
1. **Library Management System:**

**Communication diagrams for LMS**

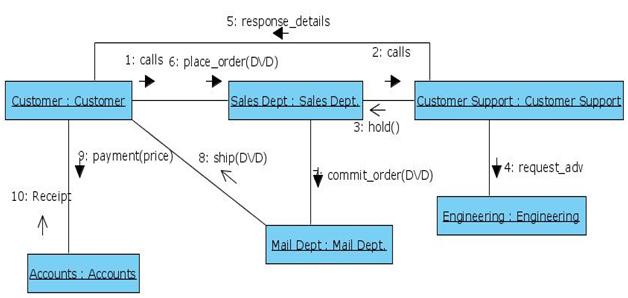


## Point-Of-Sale Terminal:

**Communication diagrams for POS**



1. **Communication diagram for customer support system:**



**Experiment 6**

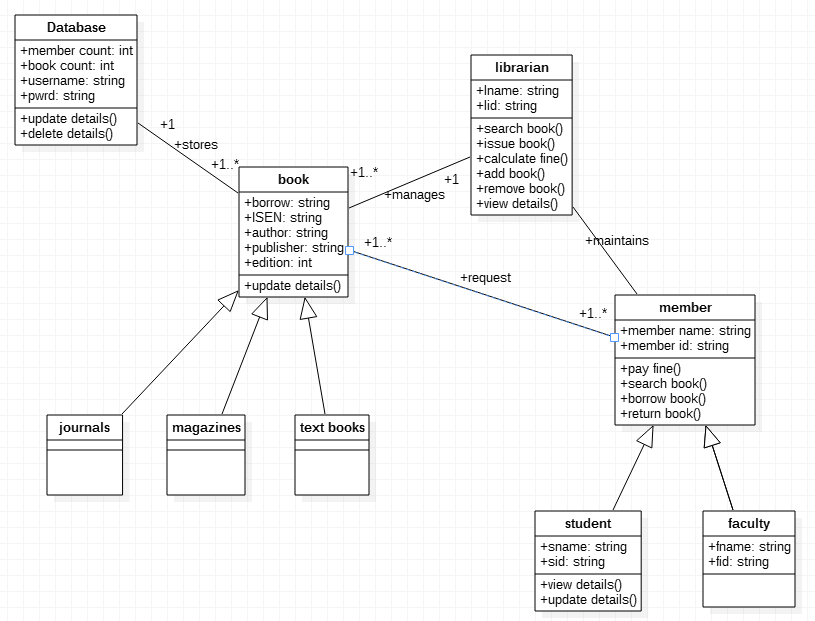
**For each case study**

## Develop detailed design class model (use GRASP patterns for responsibility assignment)

**Class diagrams:** Rose uses class diagrams to graphically describe generic descriptions of the system you’re going to build. Class diagrams contain icons that represent classes and interfaces and their relationships to one another.

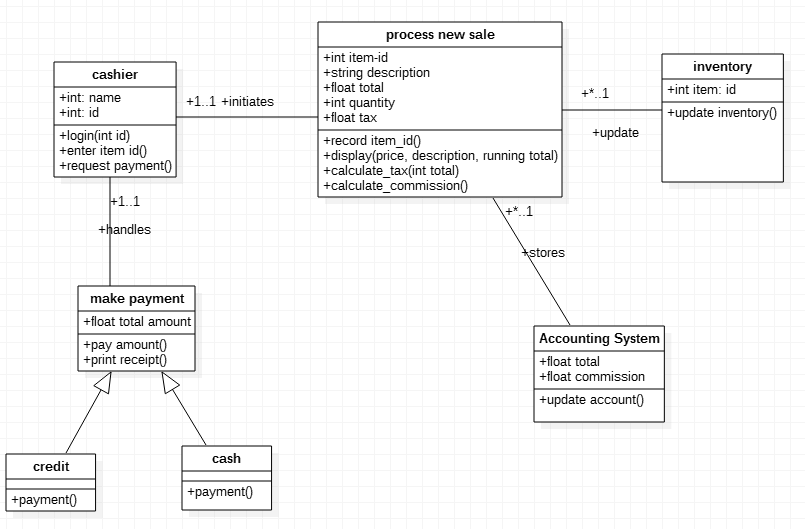
1. **Library Management System:**

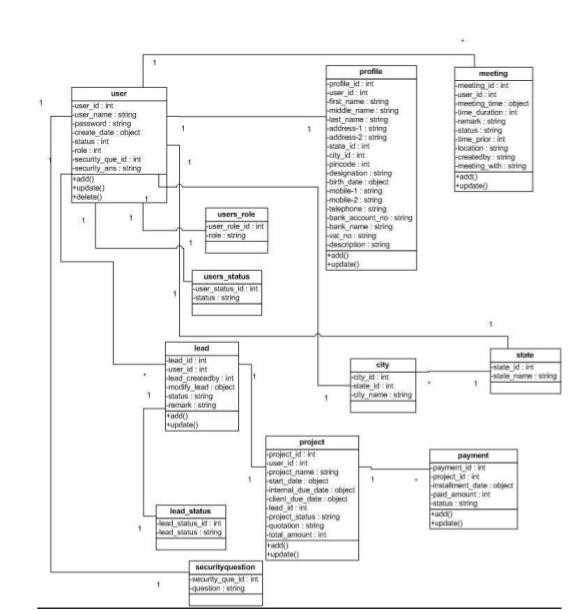
**Design class diagram for LMS**



## Point-Of-Sale Terminal:

**Design class diagram for POS**

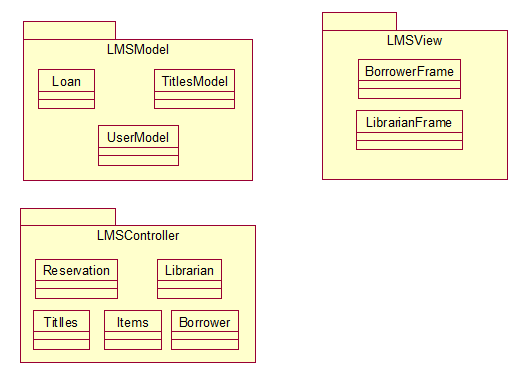


**3. Class Diagram for Customer Support System:**

## b) Develop three-layer package diagrams for each case study

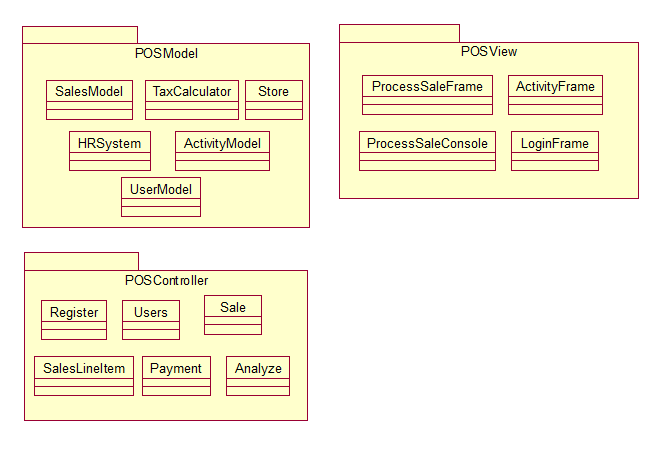
1. **Library Management System:**

**Package diagram for LMS**



1. **Point-Of-Sale Terminal:**

**Package diagram for POS**



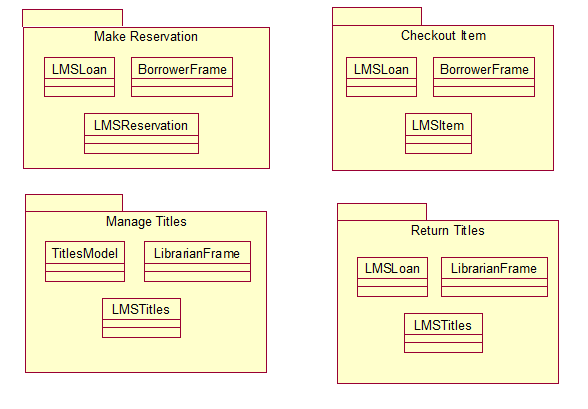
**Experiment 7**

**For each case study**

## Develop Use case Packages

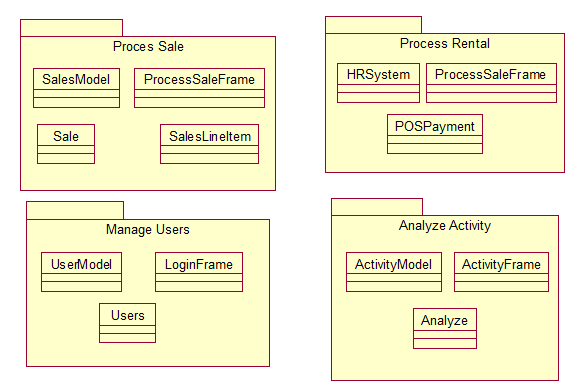
1. **Library Management System:**

**Use case packages for Library Management System**



1. **Point-Of-Sale Terminal:**

**Use case packages for Point-Of-Sale System**

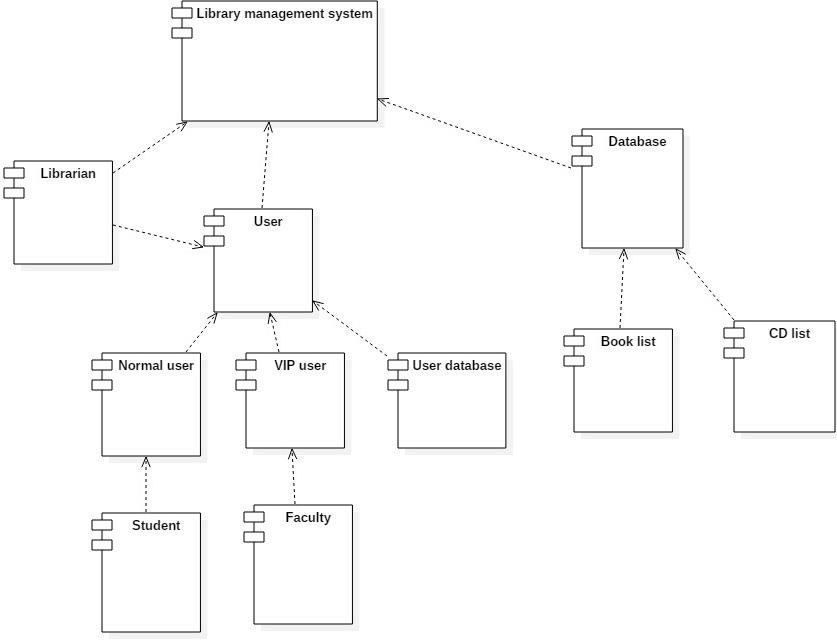


## Develop component diagrams

**Component diagrams:** Rose uses component diagrams to clearly reflect the physical dependency relationships between components (i.e., main program, subprogram, packages, and tasks) and their arrangement in a graphical manner.

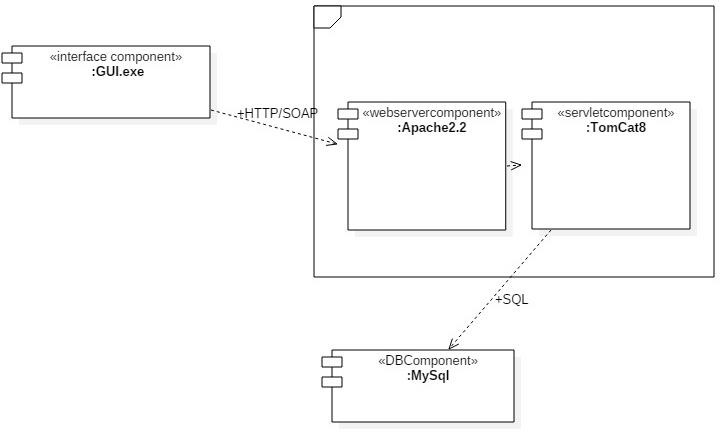
1. **Library Management System:**

**Component diagram for Library Management System**



1. **Point-Of-Sale Terminal:**

**Component diagram for POS**



## 3) Component Diagram for CSS:

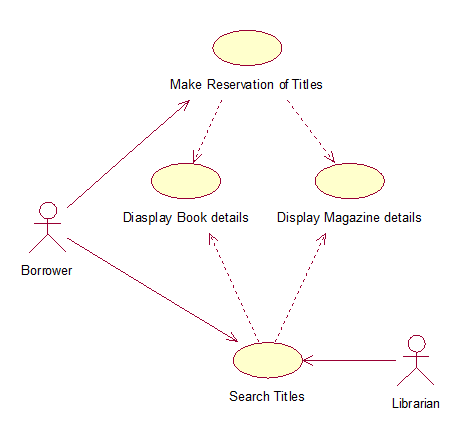
**Component diagram for CSS**

## 

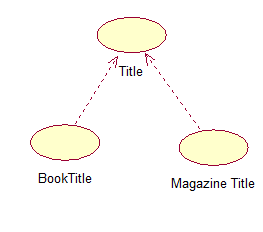
## Identify relationships between use cases and represent them

* + 1. **Library Management System:**

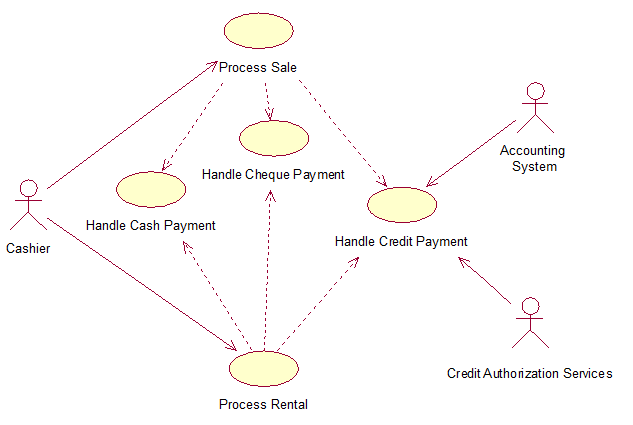
**Include relations for Library Management System**



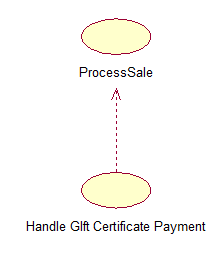
**Extends relations for Library Management System**



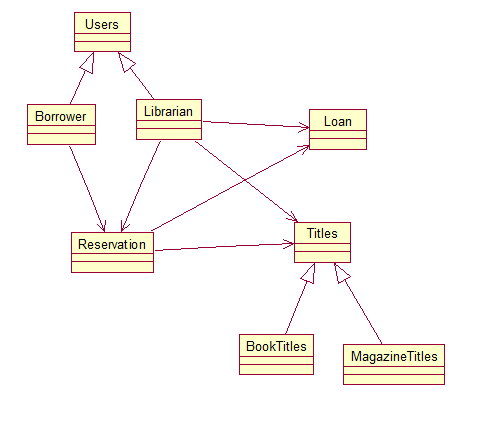
* + 1. **Point-Of-Sale Terminal:**

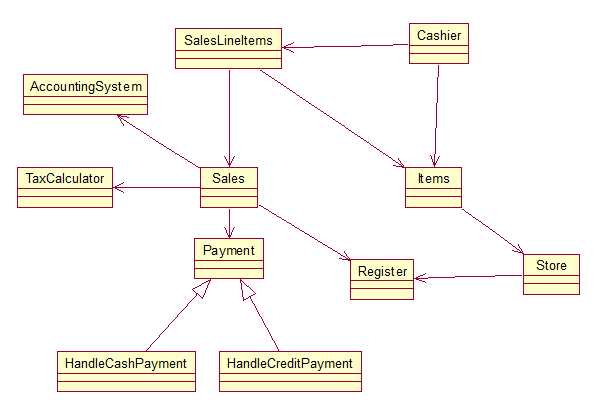


**Extends relations for POS**

**d) Refine domain class model by showing all the associations among classes**

* 1. **Library Management System:**



* 1. **Point-Of-Sale System**

**Experiment 8**

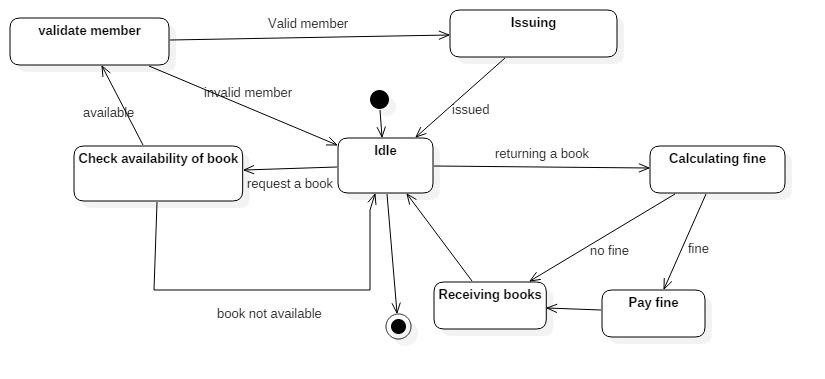
**For each case study**

## Develop sample diagrams for other UML diagrams - state chart diagrams, activity diagrams and deployment diagrams

Rose allows users to use state chart diagrams (which are state-driven) to model the dynamic behavior of individual classes or objects. State chart diagrams are very similar to activity diagrams (which are activity-driven). Basically, these diagrams show you (1) the sequence of states that an object will go through, (2) the events that cause a transition from one activity to another, and (3) any actions that result from the state or activity change.

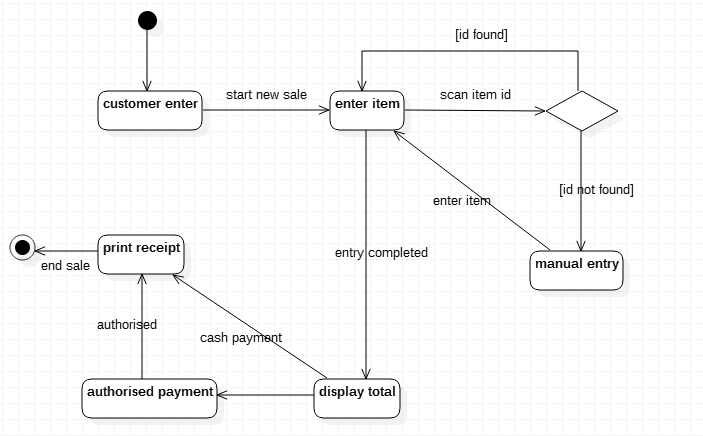
1. **Library Management System:**

**State chart diagram for Library Management System**

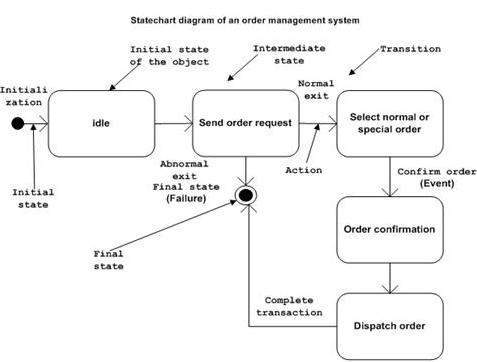


1. **Point-Of-Sale Terminal:**

**State chart diagram for Point - Of - Sale System**

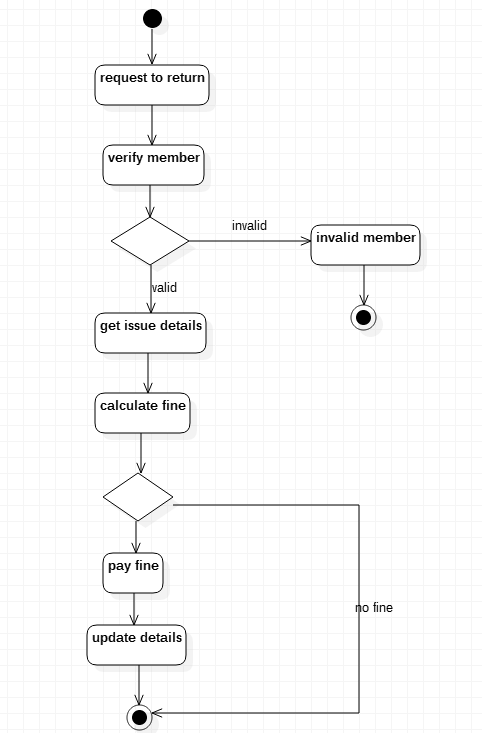


* 1. **Statechart diagram for customer support system:**

****

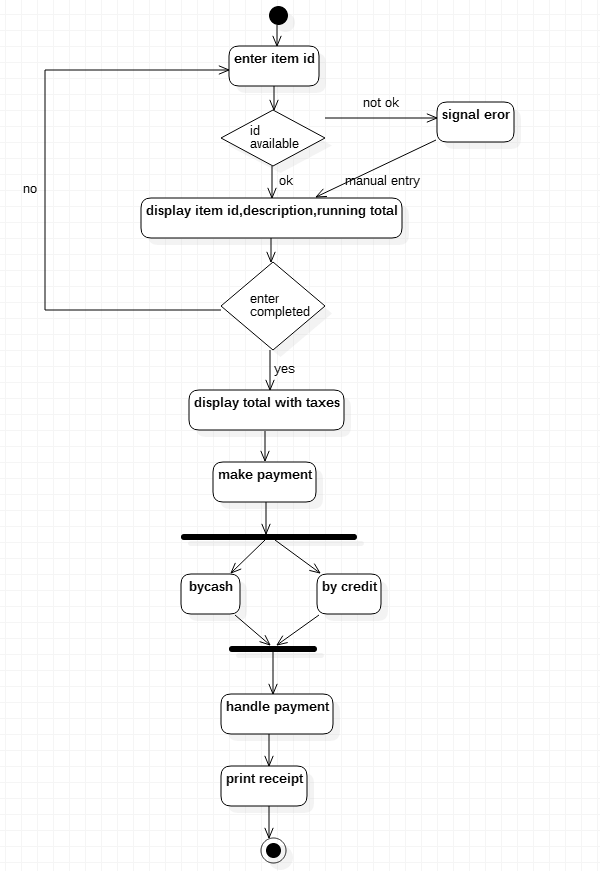
**1.Library Management System:**

**Activity diagram for return a book**

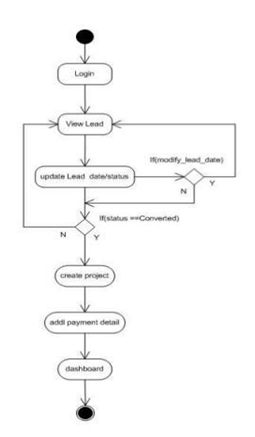


**2. Point-Of-Sale Terminal:**

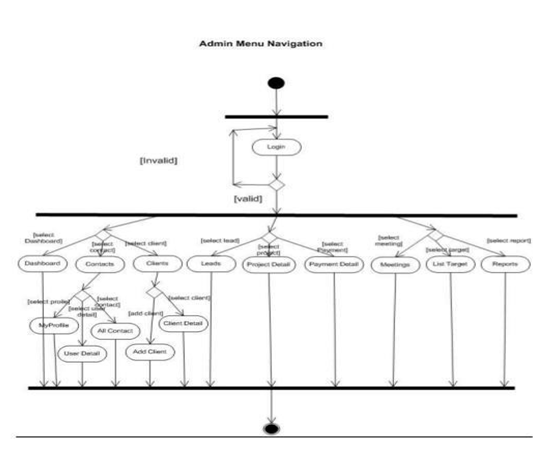
**Activity diagram for POS**



**3. Activity diagram for CSS:**



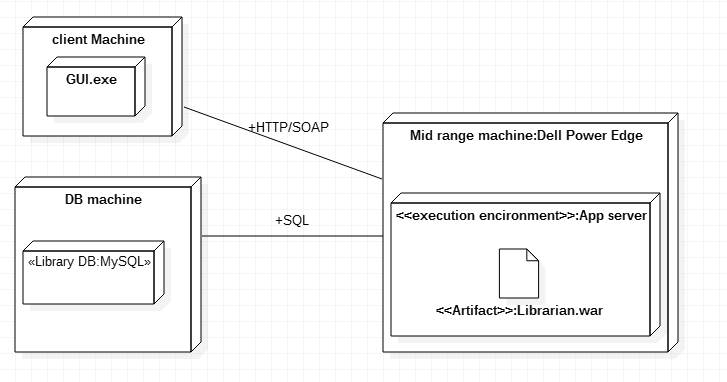
**Activity diagram for the entire system**

****

**Deployment diagrams:** Using the deployment diagram, Rose allows users to graphically show the connections between processors, devices, and connections.

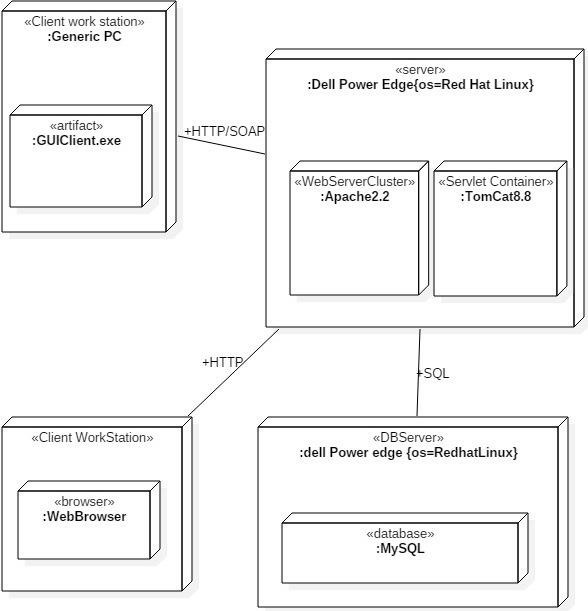
**1.Library Management System:**

**Deployment diagram for Library Management System**

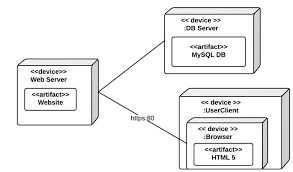


**2.Point-Of-Sale Terminal:**

**Deployment diagram for POS**



**3. Deployment diagram for CSS:**

****