

P.T.Lee Chengalvaraya Naicker College of Engineering & Technology

(Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai)
Vallal P.T.Lee Chengalvaraya Naicker Nagar, Oovery, Kanchipuram - 631 502



Department of Computer Science and Engineering

Regulation - 2021

CCS356 – OBJECT ORIENTED SOFTWARE ENGINEERING

Record

Name :

Reg. No :

Year/ Semester :

P.T.Lee Chengalvaraya Naicker College of Engineering & Technology

(Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai)

Vallal P.T.Lee Chengalvaraya Naicker Nagar, Ooveri,
Kanchipuram – 631 502.

Department of Computer Science and Engineering



BONAFIDE CERTIFICATE

Reg.No: _____

Certified that this is a bonafide record of work done by the Candidate
Mr\Mrs. _____ in **CCS356-Object Oriented Software
Engineering** during the academic year _____.

Lab-in-charge

Head of the Department

Submitted for the Practical Examination held on _____.

Internal Examiner

External Examiner

CCS356-OBJECT ORIENTED SOFTWARE ENGINEERING Laboratory

LIST OF EXPERIMENTS

Draw standard UML diagrams using an UML modeling tool for a given case study and map design to code and implement a 3 layered architecture. Test the developed code and validate whether the SRS is satisfied.

1. Identify a software system that needs to be developed.
2. Document the Software Requirements Specification (SRS) for the identified system.
3. Identify use cases and develop the Use Case model.
4. Identify the conceptual classes and develop a Domain Model and also derive a Class Diagram from that.
5. Using the identified scenarios, find the interaction between objects and represent them using UML Sequence and Collaboration Diagrams
6. Draw relevant State Chart and Activity Diagrams for the same system.
7. Implement the system as per the detailed design
8. Test the software system for all the scenarios identified as per the usecase diagram
9. Improve the reusability and maintainability of the software system by applying appropriate design patterns.
10. Implement the modified system and test it for various scenarios

SUGGESTED DOMAINS FOR MINI-PROJECT:

1. Passport automation system.
2. Book bank
3. Exam Registration
4. Stock maintenance system.
5. Online course reservation system
6. Airline/Railway reservation system
7. Software personnel management system
8. Credit card processing
9. e-book management system
10. Recruitment system
11. Foreign trading system
12. Conference Management System
13. BPO Management System
14. Library Management System
15. Student Information System

LIST OF EQUIPMENT FOR A BATCH OF 30 STUDENTS

Suggested Software Tools:

Windows 7 or Higher ,Argo UML that supports UML 1.4 and higher,

Software Tools 30 user License

ArgoUML

Selenium ,JUnit or Apache JMeter

Eclipse IDE JUnit PC's 30

Ex.No		Title	Page.No	Marks	Sign
1		Study of ARGOUML & UML Diagrams			
2		PROBLEM STATEMENT - QUIZ SYSTEM			
3		SRS DOCUMENT & USE CASE MODELING			
4		UML Class Diagram			
5		UML Interaction Diagram			
6		UML State Chart Diagram and Activity Diagram			
7		Project Implementation			
8		Project Testing			
9		Applying Design Patterns , Implementing & Testing			

MINI PROJECT

Ex.No		Title	Page.No	Marks	Sign
1		Passport automation system			
2		Book bank			
3		Stock maintenance system			
4		Foreign trading system			
5		BPO Management System			

Ex. No: 1	STUDY OF ARGOUML & UML DIAGRAMS
Date:	

ArgoUML

ArgoUML is an open source Unified Modeling Language (UML) modeling tool that includes support for all standard UML 1.4 diagrams. It runs on any Java platform and is available in ten languages.

ArgoUML is different: i) it makes use of ideas from cognitive psychology, ii) it is based on open standards; iii) it is 100% pure Java; and iv) it is an open source project.

CASE Tools

Computer-aided software engineering (CASE) is the scientific application of a set of tools which is meant to result in high-quality, defect-free, and maintainable software products.

FEATURES

- Support open standards extensively: UML, XMI, SVG, OCL and others.
- 100% Platform independent thanks to the exclusive use of Java
- Open Source, which allows extending or customizing.
- Cognitive features like
 - reflection-in-action
 - Design Critics
 - Corrective Automations (partially implemented)
 - "To Do" List
 - User model (partially implemented)
 - opportunistic design
 - "To Do" List
 - Checklists
 - Comprehension and Problem Solving
 - Explorer Perspectives
 - Multiple, Overlapping Views
 - Alternative Design Representations: Graphs, Text, or Table

UML Diagrams by ArgoUML

- Use Case Diagrams
- Class Diagrams
- Behavior Diagrams
 - State chart Diagrams
 - Activity Diagrams
 - Interaction Diagrams
 - » Sequence Diagrams
 - » Collaboration Diagrams

- Implementation Diagrams
 - Component Diagrams
 - Deployment Diagrams

Use Case Diagrams

- Present a high-level view of system usage
- These diagrams show the functionality of a system or a class and how the system interacts with the outside world.
- During analysis to capture the system requirements and to understand how the system should work.

During the design phase, use-case diagrams specify the behavior of the system as implemented.

Class Diagram

- Helps you visualize the structural or static view of a system.
- Class diagrams show the relationships among class.
- Foundation for component and deployment diagrams.

Sequence Diagram

- Illustrates object interacts arranged in a time sequence
- This shows step-by-step what has to happen to accomplish something in the use case and emphasize the sequence of events.

Collaboration Diagram

- Provides a view of the interactions or structured relationships between objects in current model.
- Emphasizes relation between objects.
- Used as the primary vehicle to describe interactions that express decision about system behavior.

State Chart Diagram

- To model the dynamic behaviors of individual classes or objects
- Used to model the discrete stages of an objects lifetime.

Activity Diagram

- Model the workflow of a business process and the sequence of activities in a process.
- Similar to a flowchart, l a workflow from activity to activity or from activity to state.
- It is help to understand the overall process.

Component Diagram

- A physical view of the current model and Show the organization and dependencies of software components, including source code, binary code, and executable components

Deployment Diagram

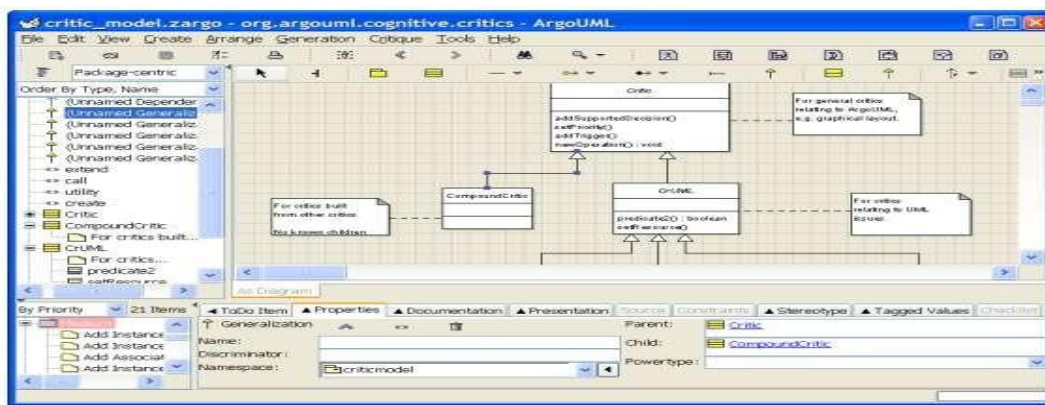
- Each model contains a single deployment diagram that shows the mapping of processes to hardware.

THE ArgoUML USER INTERFACE

Overview of the Window

The title bar of the window shows the following 4 parts of information, separated from each other by a dash.

- The current filename.
- The name of the currently active diagram.
- The name “ArgoUML”.
- An asterisk (*). This item is only present if the current project file is “dirty”, i.e. it is altered, but not yet saved. In other words, if the asterisk is absent, then the current file has not been altered.



The window comprises four sub-windows or panes namely,

- The Explorer pane,
- The Editing pane
- The Details Pane and
- The To-Do Pane.

THE EDITING PANE






The Toolbar

The toolbar at the top of the editing pane provides the main functions of the pane.





The Toolbar consists of the following menus:

- File operations
- Edit operations
- View operations
- Create operations

File operations

-  New...
 -  Open Project...
 -  Save Project
 -  Project Properties
-
-  Print

Edit operations

-  Remove From Diagram
gram".
-  Delete From Model
-  Configure Perspectives
-  Settings

View operations

-  Find...
-  Zoom

Create operations

-  New Use Case Diagram
-  New Class Diagram
-  New Sequence Diagram
-  New Collaboration Diagram
-  New Statechart Diagram
-  New Activity Diagram
-  New Deployment Diagram

The tools fall into four categories.

- ***Layout tools.*** Provide assistance in laying out model elements on the diagram.
- ***Annotation tools.*** Used to annotate model elements on the diagram.
- ***Drawing tools.*** Used to add general graphic objects to diagrams.
- ***Diagram specific tools.*** Used to add UML model elements specific to a particular diagram type to the diagram.
- ***Drawing tools.*** Used to add general graphic objects to diagrams.
- ***Diagram specific tools.*** Used to add UML model elements specific to a particular diagram type to the diagram.
- **Layout Tools**

↑ **Select** - This tool provides for general selection of model elements on the diagram.

⌘ **Broom Tool** – This tool is used to sweep all model elements along.

- **Layout Tools**

↑ **Select** - This tool provides for general selection of model elements on the diagram.

↓ **Broom Tool** - This tool is used to sweep all model elements along.









- **Annotation Tools**



Annotation tool - It is used to add a comment to a selected UML model element.

- **Drawing Tools**



-  **Rectangle**. Provides a rectangle.
-  **Rounded Rectangle**. Provides a rectangle with rounded corners. There is no control over the degree of rounding.
-  **Circle**. Provides a circle.
-  **Line**. Provides a line.
-  **Text**. Provides a text box. The text is entered by selecting the box and typing. Text is centered horizontally and after typing, the box will shrink to the size of the text. However it can be re-sized by dragging on the corners.
-  **Polygon**. Provides a polygon. The points of the polygon are selected by button 1 click and the polygon closed with button 1 double click (which will link the final point to the first point).
-  **Spline**. Provide an open spline. The control points of the spline are selected with button 1 and the last point selected with button 1 double click.
-  **Ink**. Provide a polyline. The points are provided by button 1 motion.

USE CASE DIAGRAM SPECIFIC TOOLS



Actor - Add an actor to the diagram.








Use Case – Add a use case to the diagram.

Association - Add an association between two model elements selected using button 1 motion (from the first model element to the second).

The association tool selector
















-  **Dependency**. Add a dependency between two model elements selected using button 1 motion (from the dependent model element).
-  **Generalization**. Add a generalization between two model elements selected using button 1 motion (from the child to the parent).
-  **Extend**. Add an extend relationship between two model elements selected using button 1 motion (from the extended to the extending use case).
-  **Include**. Add an include relationship between two model elements selected using button 1 motion (from the including to the included use case).
-  **Add Extension Point**. Add an extension point to a selected use case.






CLASS DIAGRAM SPECIFIC TOOLS


Class diagrams are used for only one of the UML static structure diagrams, the class diagram itself. Object diagrams are represented on the ArgoUML deployment diagram.


ArgoUML uses the class diagram to show model structure through the use of packages.

-  **Package.** Add a package to the diagram.
-  **Class.** Add a class to the diagram. For convenience, when the mouse is over a selected class it displays two handles to left and right which may be clicked or dragged to form association relationships (or composition in case SHIFT has been pressed) and two handles top and bottom which may be dragged or clicked to form generalization and specialization relationships respectively.
-  **Association.** Add an association between two model elements selected using button 1 motion (from the first model element to the second). There are 2 types of association offered here, `bidirectional` or `unidirectional`.
-  **Aggregation.** Add an aggregation between two model elements selected using button 1 motion (from the first model element to the second). There are 2 types of aggregation offered here, `bidirectional` or `unidirectional`.
-  **Composition.** Add an composition between two model elements selected using button 1 motion (from the first model element to the second). There are 2 types of composition offered here, `bidirectional` or `unidirectional`.
-  **Association-end.** Add another end to an already existing association using button 1 (from the association middle to a class, or vice versa). This is the way to create so-called N-ary associations.
-  **Generalization.** Add a generalization between two model elements selected using button 1 (from the child to the parent).
-  **Interface.** Add an interface to the diagram. For convenience, when the mouse is over a selected interface it displays a handle at the bottom which may be dragged to form a realization relationship (the target being the realizing class).
-  **Realization.** Add a realization between a class and an interface selected using button 1 motion (from the realizing class to the realized interface).
-  **Dependency.** Add a dependency between two model elements selected using button 1 motion (from the dependent model element). There are also 2 special types of dependency offered here, `Permission` () and `Usage` (). A `Permission` is created by default with stereotype `Import`, and is used to import elements from one package into another.
-  **Attribute.** Add a new attribute to the currently selected class. The attribute is given the default name `newAttr` of type `int` and may be edited by button 1 double click and using the keyboard, or by selecting with button 1 click (after the class has been selected) and using the property tab.






SEQUENCE DIAGRAM SPECIFIC TOOLS

-  Classifier Role. Add a classifierrole to the diagram.
-  Message with Call Action. Add a call message between two classifierroles selected using button 1 motion (from the originating classifierrole to the receiving classifierrole).
-  Message with Return Action. Add a return message between two classifierroles selected using button 1 motion (from the originating classifierrole to the receiving classifierrole).
-  Message with Create Action. Add a create message between two classifierroles selected using button 1 motion (from the originating classifierrole to the receiving classifierrole).
-  Message with Destroy Action. Add a destroy message between two classifierroles selected using button 1 motion (from the originating classifierrole to the receiving classifierrole).
-



 Add Vertical Space to Diagram. Add vertical space to a diagram by moving all messages below this down. Click the mouse at the point where you want the space to be added and drag down the screen vertically the distance which matches the height of the space you'd like to have added.

-  Remove Vertical Space in Diagram. Remove vertical space from diagram and move all elements below up vertically. Click and drag the mouse vertically over the space that you want deleted.










COLLABORATION DIAGRAM SPECIFIC TOOLS

-  Classifier Role. Add a classifier role to the diagram.
-  Association Role. Add an association role between two classifier roles selected using button 1 motion (from the originating classifier role to the receiving classifier role). There are 6 types of association roles offered here, see Figure 12.4, "The association tool selector.": association, aggregation and composition, and all these three can be bidirectional or unidirectional.
-  Generalization. Add a generalization between two model elements selected using button 1 (from the child to the parent).
-  Dependency. Add a dependency between two model elements selected using button 1 motion (from the dependent model element).
-  Add Message. Add a message to the selected association role.









STATE CHART DIAGRAM SPECIFIC TOOLS

-  Simple State. Add a simple state to the diagram.
-  Composite State. Add a composite state to the diagram.

ACTIVITY DIAGRAM SPECIFIC TOOLS

-  **Action State.** Add an action state to the diagram.
-  **Transition.** Add a transition between two action states selected using button 1 motion (from the originating action state to the receiving action state).
-  **Initial.** Add an initial pseudostate to the diagram.
-  **Fork.** Add a fork pseudostate to the diagram.
-  **Final State.** Add a final state to the diagram.
-  **Junction.** Add a junction (decision) pseudostate to the diagram.
-  **Join.** Add a join pseudostate to the diagram.
-  **CallState.** Add a callstate to the diagram. A call state is an action state that calls a single operation. Hence, the name of the operation being called is put in the symbol, along with the name of the classifier that hosts the operation in parentheses under it.
-  **ObjectFlowState.** Add an objectflowstate to the diagram. An objectflowstate is an object that is input to or output from an action.

DEPLOYMENT DIAGRAM SPECIFIC TOOLS

-  **Node.** Add a node to the diagram. For convenience, when the mouse is over a selected node it displays four handles to left, right, top and bottom which may be dragged to form association relationships.
-  **Node Instance.** Add a node instance to the diagram. For convenience, when the mouse is over a selected node instance it displays four handles to left, right, top and bottom which may be dragged to form link relationships.
-  **Component.** Add a component to the diagram. For convenience, when the mouse is over a selected component it displays four handles to left, right, top and bottom which may be dragged to form dependency relationships.
-  **Component Instance.** Add a component instance to the diagram. For convenience, when the mouse is over a selected component instance it displays four handles to left, right, top and bottom which may be dragged to form dependency relationships.
-  **Generalization.** Add a generalization between two model elements selected using button 1 (from the child to the parent).
-  **Realization.** Add a realization between a class and an interface selected using button 1 motion (from the realizing class to the realized interface).
-  **Dependency.** Add a dependency between two model elements selected using button 1 motion (from the dependent model element).
-  **Association.** Add an association between two model elements

INTRODUCTION TO SELENIUM TESTING TOOL

Introduction

- Selenium is a robust set of tools that supports rapid development of test automation for web-based applications.
- Works anywhere JavaScript is supported
- Hooks for many other languages - Java, Ruby, Python
- Can simulate a user navigating through pages and then assert for specific marks on the pages.
- Selenium IDE is a plug-in to Firefox to record and playback tests (like WinRunner, QTP).

Selenium Components

- Selenium-IDE
- Selenium-RC (Remote Control)
- Selenium-Grid

Steps to start with Selenium!

1) Begin: write and run tests in Firefox.

Selenium IDE is a Firefox add-on that records clicks, typing, and other actions to make a test, which you can play back in the browser.

2) Customize: your language, your browser.

Selenium Remote Control (RC) runs your tests in multiple browsers and platforms. Tweak your tests in your preferred language.

3) Deploy: scale out, speed up

Selenium Grid extends Selenium RC to distribute your tests across multiple servers, saving you time by running tests in parallel.

Ex. No: 2	PROBLEM STATEMENT - QUIZ SYSTEM
Date:	

Aim: To identify a software system (Quiz System) that needs to be developed

PROBLEM STATEMENT

Online Quiz system is to be designed for the students of XYZ college of Technology. The questions should be of objective type with multiple options. The system should be able to handle errors such as 2 options for one answer. After answering the questions in the preliminary round, the system verifies whether the student has answered the minimum number of questions in each level. If not, a message is displayed to make the student answer the required number of questions within an allotted time. If the student qualifies the prelims, he/she can proceed to the finals in the same way. At the end of the quiz the users' score along with information whether he has been selected or not has to be displayed. The quiz system has two levels of questions.

LEVELS	TOTAL NO. OF QUESTIONS	MINIMUM NO. OF QUESTIONS TO BE ATTENDED	MARKS FOR EACH QUESTION	DURATION
LEVEL1- PRELIMINARY	30	25	1	15 Min
LEVEL2- FINALS	15	10	2	10 Min

RESULT:

The above program was executed successfully

Ex. No: 3	SRS DOCUMENT & USE CASE MODELING
Date:	

Aim: To document the Software requirement specification for the identified QUIZ system and To identify Use Cases and develop the Use Case model for QUIZ system

Description:

A Software requirements specification document describes the intended purpose, requirements and nature of a software to be developed. Follow the Instructions and prepare SRS document using given table of contents . Below given Table of Contents for Quiz System.

Table of Contents

Table of Contents	
Table of Figures	
1.0. Introduction	Error! Bookmark not defined.
1.1. Purpose.....	
1.2. Scope of Project	
1.3. Glossary.....	
1.4. References	
1.5. Document overview	
2.0. Overall description	
2.1. System environment.....	
2.2. Functional requirements definitions	
2.3. Use cases	
2.3.1. Use Case: Login.....	
2.3.2. Use Case: Play Level 1 /Level 2.....	
2.3.3. Use Case: Display Questions.....	
2.3.4. Use Case: Submit Answers.....	
2.3.5. Use Case: Qualify to Level 2.....	
2.3.6. Use Case: Display Score:.....	
2.4. Non-functional requirements	
3.0. Requirement specifications	
3.1. External interface specifications	
3.2. Functional Requirements.....	
3.2.1. Use Case :Login	
3.2.2. Use Case: Play Level 1 /Level 2	
3.2.3. Use Case: Display Questions	
3.2.4. Use Case: Submit Answers	
3.2.5. Use Case: Qualify to Level 2	
3.2.6. Use Case: Display Score	
3.3. Detailed non-functional requirements	

3.3.1. Functionality	
3.3.2. Usability	
3.3.3. Reliability	
3.3.4. Performance	
3.3. 5. Supportability	
3.3. 6. Security	
3.3.7. Design Constraints.....	
3.4 System Evolution	
4.0. Index	

Table of Figures

Figure 1 System Design	
Figure 2 Second Figure Name.....	
Figure 3 Third Figure Name.....	
Figure 4 Fourth Figure Name	

1.0. Introduction

1.1. Purpose

The purpose of this document is to present a detailed description of the Online Quiz System. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate and how the system will react to external stimuli. This document is intended for both the Stakeholder and the developers of the system .

1.2. Scope of Project

This software system will be a Online Quiz System for Students Society. The scope of designing a Quiz system is to carry out the process faster and maintaining accuracy. The system should be able to handle errors such as two options for one answer. It must be efficient in such a way that is satisfies all the participants

1.3. Glossary

Term	Definition
Participant	Anyone using this s/w to test their knowledge
Database	Collection of all the information monitored by this system.
Level	It Shows the complexity of the question
Stakeholder	Any person with an interest in the project who is not a developer.
Coordinator	Person who conducts the quiz
Login Form	Checks for User Name and password of the participant.
Questions for the Quiz	Questions for the quiz to selected from 3 different tables namely Technical, Logic and GK

Coordinator	Person who conducts the quiz
Timer Display	Timer Display which displays the time left for answering.
Level Selection System	Level Selection System, which takes care for selecting questions randomly from the tables

1.4. References

None

1.5. Document overview

The next chapter, the Overall Description section, of this document gives an overview of the functionality of the product. It describes the informal requirements and is used to establish a context for the technical requirements specification in the next chapter.

The third chapter, Requirements Specification section, of this document is written primarily for the developers and describes in technical terms the details of the functionality of the product.

Both sections of the document describe the same software product in its entirety, but are intended for different audiences and thus use different language.

2.0. Overall description

In a quiz system we perform the following

- 1) Student registration
- 2) Quiz Participation
- 3) Result Processing

Student Registration

The inputs to the quiz system initially is the student details for the quiz which involves the following details like

- i. name
- ii. year
- iii. branch
- iv. college name

When a student enters these details it will be updated in a database. After that the student will be allowed to participate.

Quiz Participation

When the participant begins to answer the questions, a clock is maintained to keep track of the time.

The correct answers for the questions are stored in a separate database .After answering the questions in the preliminary round,

the system verifies whether the student has answered the minimum number of questions in each level. If not, a message is displayed to make the student answer the required number of questions within an allotted time. If the student qualifies the prelims, he/she can proceed to the finals in the same way.

Result Processing

After all the students have participated in the quiz, the system processes the marks scored by all the students. The system sorts the marks and generates a final output, which displays the name and scores.

2.1 System Environment

The Quiz system will be operated from any node. When a participant login to the Quiz system, it updates the information into the database to the Server. Then Server will display the questions to the participant through user Interface Screens, which allows the participant to submit their answers to a database.

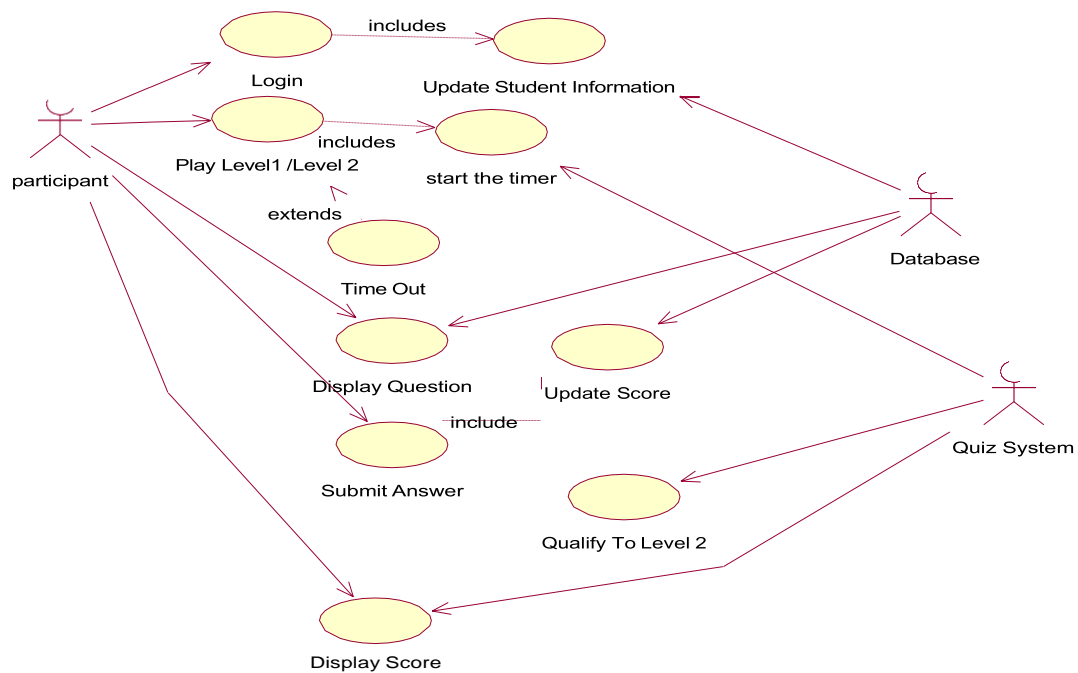


Figure 1 System Design

2.2 Functional Requirements definitions

Functional Requirements are those that refer to the functionality of the system, i.e., what services it will provide to the user. Nonfunctional (supplementary) requirements pertain to other information needed to produce the correct system and are detailed separately.

2.3 Use cases

The system will consist of Login screen to authenticate the participants whose information is updated in the database. On starting the quiz , timer is started to maintain the timings. In Level 1 /Level 2 Questions with four options are displayed sequentially .User select the answer and move to the next question . Finally he/she selects the Submit answers which updates the marks and displays the score to the participants. If he is playing Level 1 and qualified for level 2 , then next level questions are displayed otherwise Not Qualified Message is displayed.

2.3.1. Use Case: Login

Brief Description

The use case describes how a Participant logs into the Quiz System. The System requests the actor to enter his/her name and password. The actor enters his/her name and password. The System validates the entered name and password and logs the actor into the System. Reference SRS 3.2.1

2.3.2. Use Case: Play Level 1 /Level 2 (provide the content same as above

2.3.3. Use Case: Display Questions (provide the content same as above)

2.3.4. Use Case: Submit Answers (provide the content same as above)

2.3.5. Use Case: Qualify to Level 2 (provide the content same as above)

2.3.6. Use Case: Display Score (provide the content same as above)

2.4. Non-functional requirements

Non-functional requirements are often called qualities of a system. Other terms for non-functional requirements are "constraints", "quality attributes", "quality goals", "quality of service requirements" and "non-behavioral requirements".

Qualities, that is, non-functional requirements, can be divided into two main categories:

1) Execution qualities, such as security and usability, which are observable at run time.

2) Evolution qualities, such as testability, maintainability, extensibility and scalability, which are embodied in the static structure of the software system.

3.0. Requirement specifications

3.1. External interface specifications

None

3.2. Functional Requirements

3.2.1. Login

Use Case Section	Comment
Use Case Name	Login

Scope	Quiz System
Level	"user-goal"
Primary Actor	Participant
Stakeholders and Interest list	- Participant: logs into the Quiz System ...
Preconditions	None
Success Guarantee/ Post condition	If the use case was successful, the actor is now logged into the system. If not, the system State is unchanged.
Main Success Scenario	1. The System requests the actor to enter his/her name and password 2. The actor enters his/her name and password 3. The System validates the entered name and password and logs the actor into the System
Extensions	3a.If the pwd is wrong, user is allowed for 3 attempts
Special Requirements	Timer
Technology and Data Variations List	-
Frequency of Occurrence	Could Be nearly Continuous
Miscellaneous	If the connection is terminated before the form is submitted, the fields are cleared and the Server is returned to the wait state. The Administrator can make the system not to get updated by others.

3.2.2. Use Case: Play Level 1 /Level 2 (provide the content same as above)

3.2.3. Use Case: Display Questions (provide the content same as above)

3.2.4. Use Case: Submit Answers (provide the content same as above)

3.2.5. Use Case: Qualify to Level 2 (provide the content same as above)

3.2.6. Use Case: Display Score (provide the content same as above)

3.3. Detailed non-functional requirements

3.3.1. Functionality

Multiple users must be able to perform their work concurrently. If the participant has completed 30 Minutes allotted for him/her, he or she should be notified with the message "your time slot is over".

3.3.2. Usability

The desktop user-interface shall be Windows 95/98/2000/xp compliant.

3.3.3. Reliability

The System should function properly for allotted time slot and produces score card with no more than 10% down time.

3.3.4. Performance

1. The System shall support up to 100 simultaneous users against the central database at any given time and up to 100 simultaneous users against the local servers at any one time.
2. The System must be able to complete 80% of all transactions within 2 minutes.

3.3. 5. Supportability

None.

3.3. 6. Security

1. The System should secure so that only registered participants can take part in Quiz.
2. Once the participant had submitted a answer, he/she can't change the answer later.

3.3.7. Design Constraints.

The system shall provide a window-based desktop interface.

3.4. System Evolution

In the future this system will be updated to allow students to select their Subject Domain for attending the Quiz .

4.0. Index

Use Case, 3, 5, 6, 7, 8

List of words list of Page Numbers

USECASE MODELING

Guideline: Prepare Usecase diagram and place inside the SRS document.

The guidelines to prepare usecase diagram is given below.

To Find Use Cases

Use cases are defined to satisfy the goals of the primary actors. Hence, the basic procedure is:

Step 1: Choose the system boundary. Is it just a software application, the hardware and application as a unit, that plus a person using it, or an entire organization?

Step 2: Identify the primary actors those that have goals fulfilled through using services of the system. The following questions help identify others that may be missed:

Who starts and stops the system?	Who does system administration?
Who does user and security management?	Is "time" an actor because the system does something in response to a time event?
Is there a monitoring process that restarts the system if it fails?	Who evaluates system activity or performance?
How are software updates handled? Push or pull update?	Who evaluates logs? Are they remotely retrieved?
In addition to human primary actors, are there any external software or robotic systems that call upon services of the system?	Who gets notified when there are errors or failures?

Step 3: Identify the goals for each primary actor.

For example, use this table to prepare actor goal list

Act or	Goal
Participant	Play level 1 Play level 2 Submit answers
...	...

Step 4: Define Use Cases

In general, define one use case for each user goal. Name the use case similar to the user goal. Start the name of use cases with a verb.

A common exception to one use case per goal is to collapse CRUD (create, retrieve, update, delete) separate goals into one CRUD use case, idiomatically called Manage <X>. For example, the goals "edit user," "delete user," and so forth are all satisfied by the Manage Users use case.

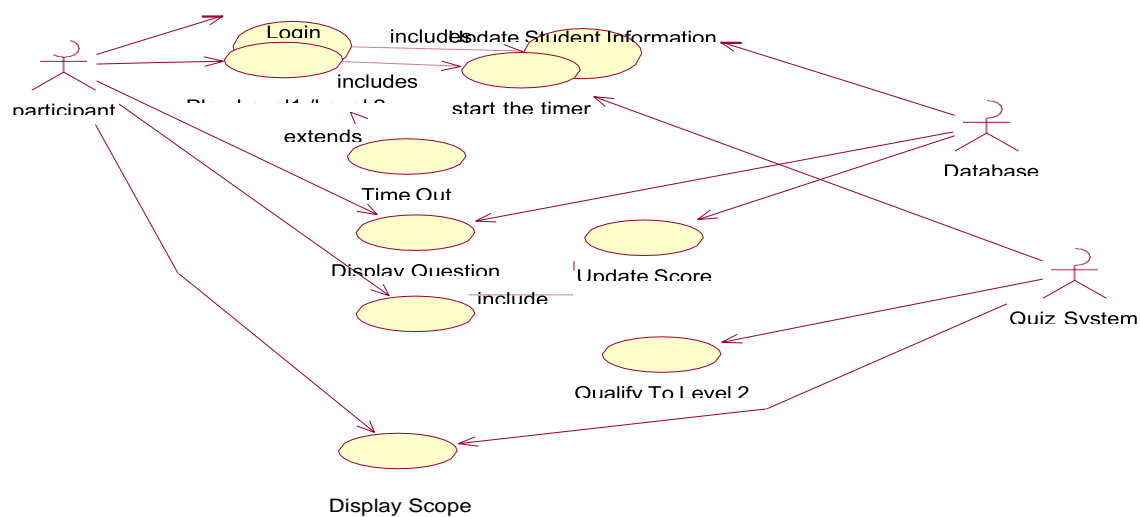
UML Notation:

Actor : An Actor, as mentioned, is a user of the system, and is depicted using a stick figure. The role of the user is written beneath the icon. Actors are not limited to humans. If a system communicates with another application, and expects input or delivers output, then that application can also be considered an actor

Use Case: A Use Case is functionality provided by the system, typically described as verb+object (e.g. Register Car, Delete User). Use Cases are depicted with an ellipse. The name of the use case is written within the ellipse. **Association:** Associations are used to link Actors with Use Cases, and indicate that an Actor participates in the Use Case in some form. Associations are depicted by a line connecting the Actor and the Use Case.



Use-case Diagram: Quiz System



Use cases

Guideline: Prepare the table format for each usecase & place inside the SRS document
 . For Example login usecase given below

Use Case: Login

Brief Description

The use case describes how a Participant logs into the Quiz System

Use Case Section	Comment
Use Case Name	Login
Scope	Quiz System
Level	"user-goal"
Primary Actor	Participant
Stakeholders and Interest list	- Participant: logs into the Quiz System ...
Preconditions	None
Success Guarantee/ Post condition	If the use case was successful, the actor is now logged into the system. If not, the system State is unchanged.
Main Success Scenario	1. The System requests the actor to enter his/her name and password 2. The actor enters his/her name and password 3. The System validates the entered name and password and logs the actor into the System
Extensions	3a.If the pwd is wrong, user is allowed for 3 attempts
Special Requirements	Timer
Technology and Data Variations List	-
Frequency of Occurrence	Could Be nearly Continuous
Miscellaneous	If the connection is terminated before the form is submitted, the fields are cleared and the Server is returned to the wait state. The Administrator can make the system not to get updated

RESULT:

The above program was executed successfully

Ex. No: 4	UML Class Diagram
Date:	

Aim: To Identify the Conceptual Classes and to develop a domain model and derive UML class diagram for QUIZ system

Description

A class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, and the relationships between the classes.

The class diagram is the main building block in object oriented modeling. They are being used both for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. The classes in a class diagram represent both the main objects and or interactions in the application and the objects to be programmed. In the class diagram these classes are represented with boxes which contain three parts

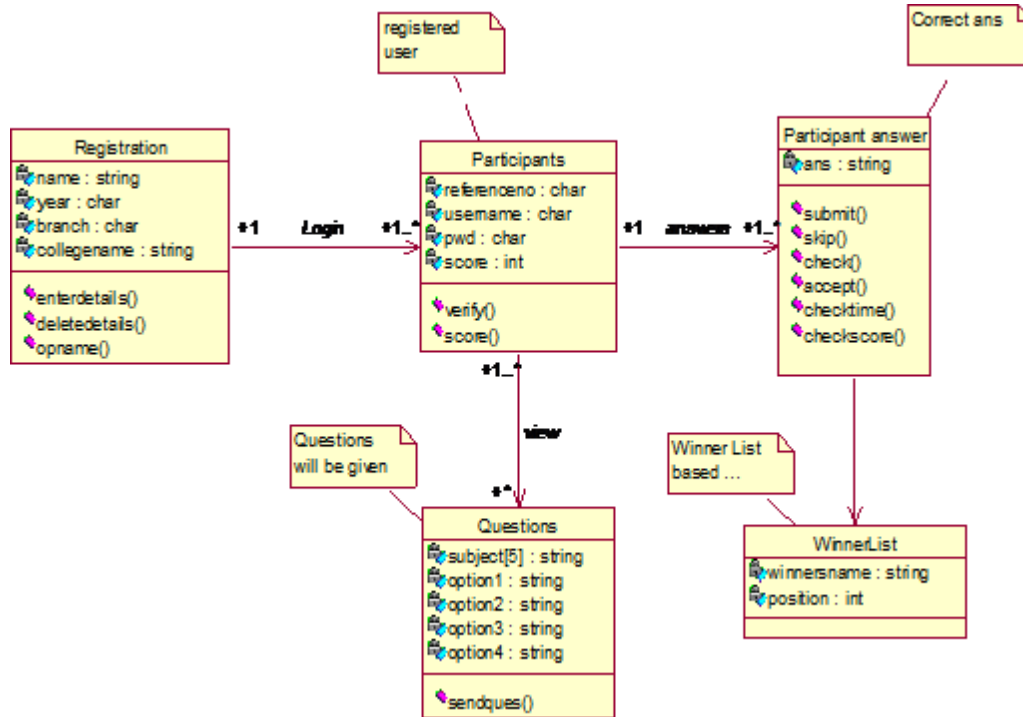


A class with three sections.

- The upper part holds the name of the class
- The middle part contains the attributes of the class
- The bottom part gives the methods or operations the class can take or undertake

In the system design of a system, a number of classes are identified and grouped together in a class diagram which helps to determine the statically relations between those objects. With detailed modeling, the classes of the conceptual design are often split in a number of subclasses.

Class Diagram



RESULT:

The above program was executed successfully

Ex. No: 5	UML Interaction Diagram
Date:	

Aim: To Identify the interaction between objects using the identified scenarios and represent using UML Interaction diagram for QUIZ system

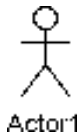
Description

Sequence diagrams document the interactions between classes to achieve a result, such as a use case. Because UML is designed for object- oriented programming, these communications between classes are known as messages. The Sequence diagram lists objects horizontally, and time vertically, and models these messages over time.

Notation In a Sequence diagram, classes and actors are listed as columns, with vertical lifelines indicating the lifetime of the object over time.

Object :Objects are instances of classes, and are arranged horizontally. The pictorial representation for an Object is a class (a rectangle) with the name prefixed by the object name (optional) and a semi-colon.

Actor:Actors can also communicate with objects, so they too can be listed as a column. An Actor is modeled using the ubiquitous symbol, the stick figure.



Lifeline:The Lifeline identifies the existence of the object over time. The notation for a Lifeline is a vertical dotted line extending from an object.

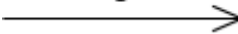


Activation:Activations, modeled as rectangular boxes on the lifeline, indicate when the object is performing an action.

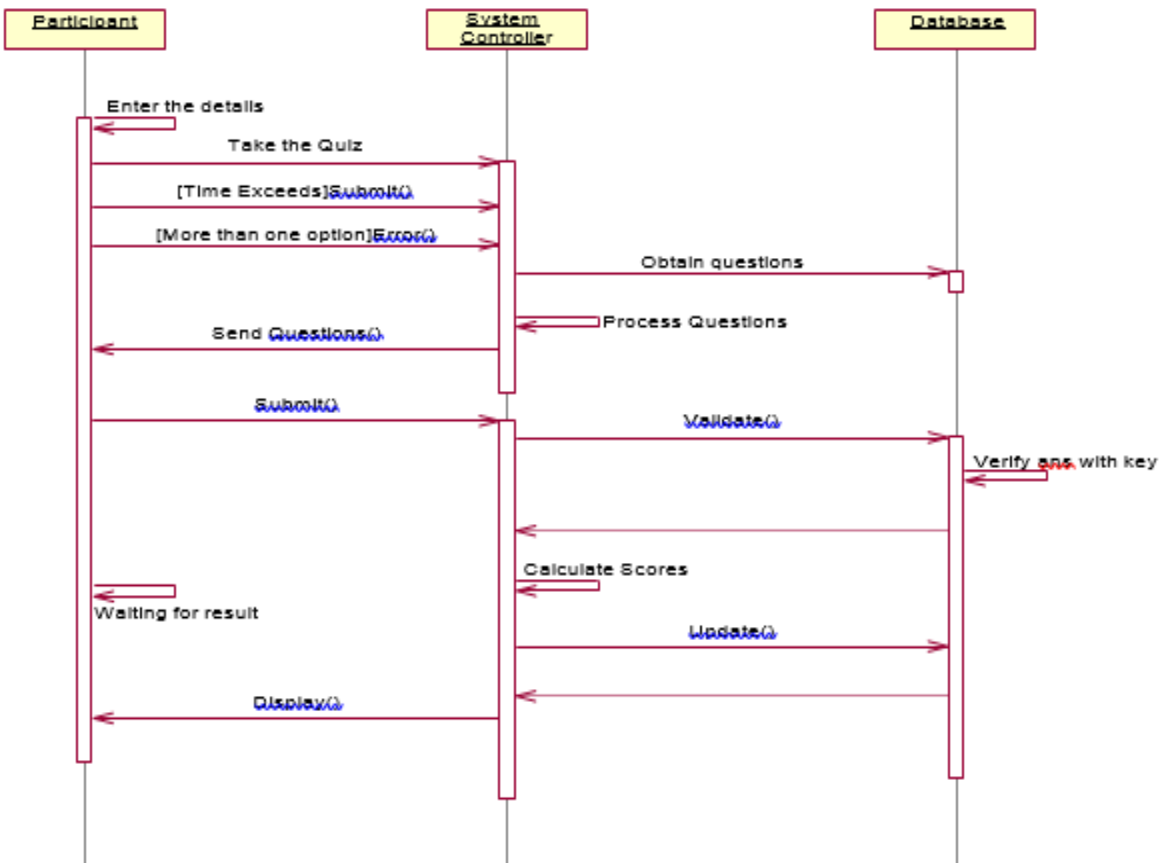


Message

Messages, modeled as horizontal arrows between Activations, indicate the

communications between objects. 

Sequence diagram



RESULT:

The above program was executed successfully

Ex. No: 6	UML State Chart Diagram and Activity Diagram
Date:	

Aim: To develop UML State Chart diagram and Activity diagrams for QUIZ system

State Chart Diagram Description

UML preserves the general form of the traditional state diagrams. The UML state diagrams are directed graphs in which nodes denote states and connectors denote state transitions. For example, Figure 1 shows a UML state diagram corresponding to the computer keyboard state machine. In UML, states are represented as rounded rectangles labeled with state names. The transitions, represented as arrows, are labeled with the triggering events followed optionally by the list of executed actions. The initial transition originates from the solid circle and specifies the default state when the system first begins. Every state diagram should have such a transition, which should not be labeled, since it is not triggered by an event. The initial transition can have associated actions.

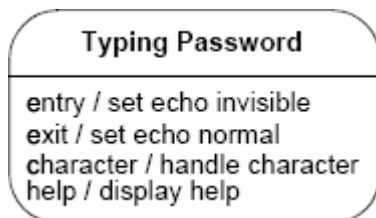
The main purposes of using Statechart diagrams:

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

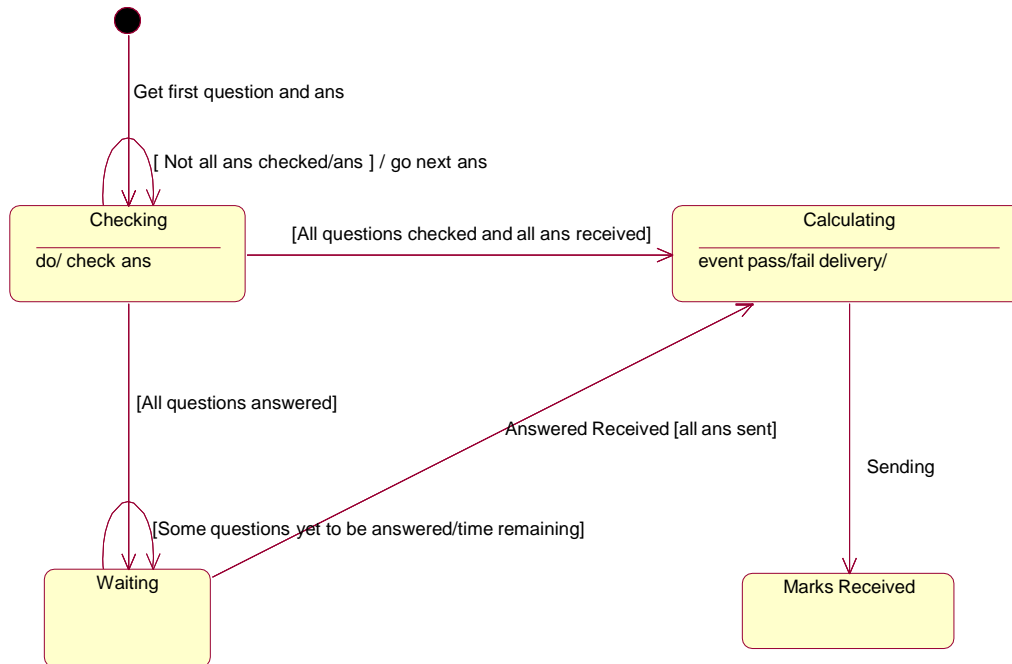
Steps to prepare state chart diagram

- Identify important objects to be analyzed.
- Identify the states.
- Identify the events.

State Notation:



State Chart Diagram



Activity Diagram Description

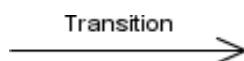
Activity diagrams are used to document workflows in a system, from the business level down to the operational level. Activity diagram is a variation of the state diagram where the "states" represent operations, and the transitions represent the activities that happen when the operation is complete. The general purpose of Activity diagrams is to focus on flows driven by internal processing vs. external events.

Activity States

Activity states mark an action by an object. The notations for these states are rounded rectangles, the same notation as found in State chart diagrams.



Transition: When an Activity State is completed, processing moves to another Activity State. Transitions are used to mark this movement. Transitions are modeled using arrows.



Swim lane: Swim lanes divide activities according to objects by arranging objects in column format and placing activities by that object within that column. Objects are listed at the top of the column, and vertical bars separate the columns to form the swim lanes.

Initial State: The Initial State marks the entry point and the initial Activity State. The notation for the Initial State is the same as in State chart diagrams,

a solid circle. There can only be one Initial State on a diagram.

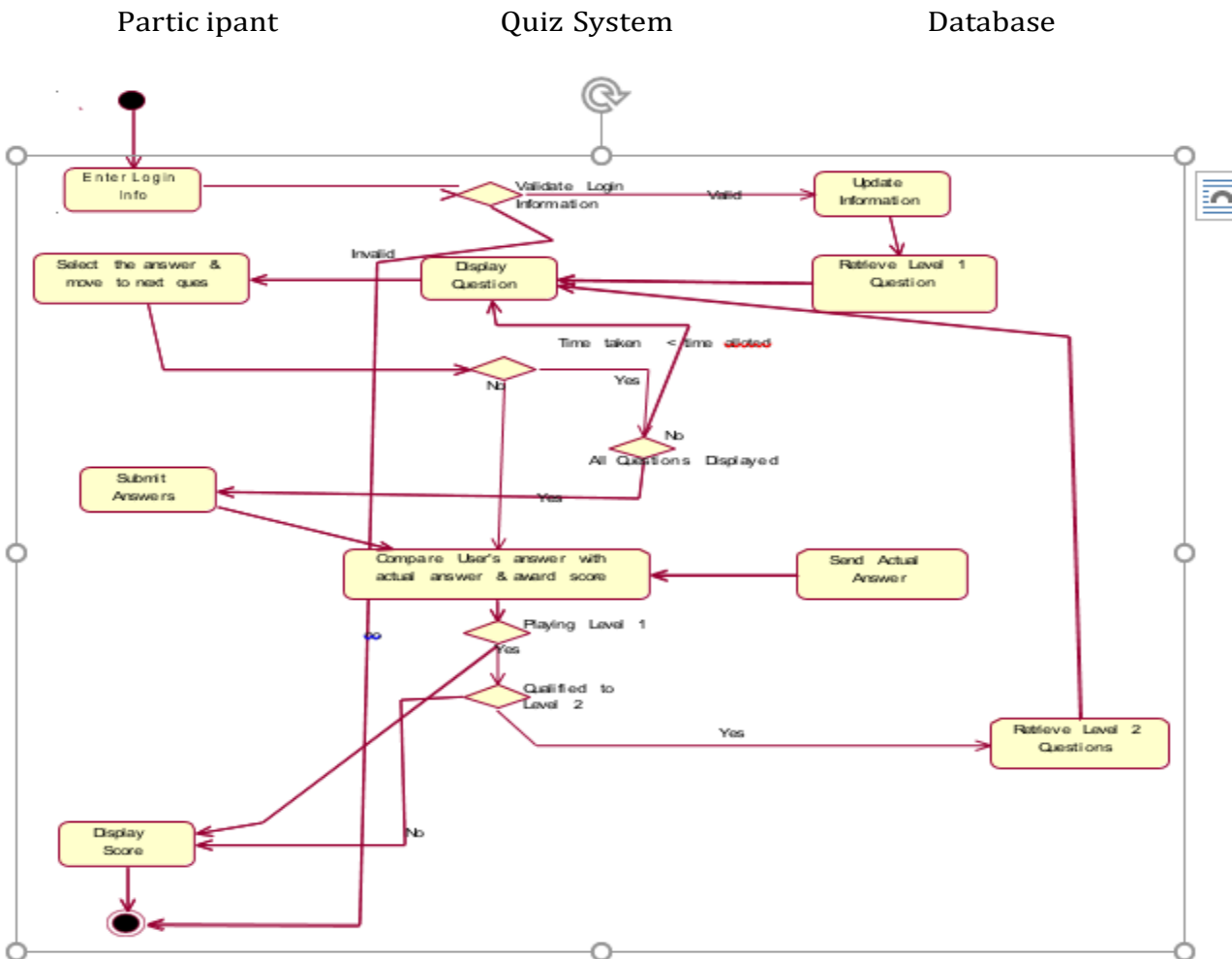
Final State: Final States mark the end of the modeled workflow. There can be multiple Final States, and these are modeled using a solid circle surrounded by



another circle.

Synchronization Bar: Activities often can be done in parallel. To split processing ("fork"), or to resume processing when multiple activities have been completed ("join"), Synchronization Bars are used. These are modeled as solid rectangles, with multiple transitions going in and/or out.

Activity diagram : Quiz System



RESULT:

The above program was executed successfully

Ex. No: 7	Project Implementation - QUIZ SYSTEM
Date:	

Aim: To implement the Quiz system as per the detailed design

Description:

In a quiz system we perform the following

- 1) Student registration
- 2) Quiz Participation
- 3) Result Processing

1) Student Registration

The inputs to the quiz system initially is the student details for the quiz which involves the following details like

- 1) name
- 2) year
- 3) branch
- 4) college name

When a student enters these details it will be updated in a database. After that the student will be allowed to participate.

2) Quiz Participation

When the participant begins to answer the questions, a clock is maintained to keep track of the time.

The correct answers for the questions are stored in a separate database .After answering the questions in the preliminary round, the system verifies whether the student has answered the minimum number of questions in each level. If not, a message is displayed to make the student answer the required number of questions within an allotted time.

If the student qualifies the prelims, he/she can proceed to the finals in the same way.

3) Result Processing

After all the students have participated in the quiz, the system processes the marks scored by all the students. The system sorts the marks and generates a final output, which displays the name and scores

SOFTWARE DEVELOPMENT

TECHNICAL SERVICES LAYER : design & create the Database in the given backend tool.

Table Structure:

Student Registration

	Field Name	Data Type
▶	Name	Text
	Branch	Text
	College	Text

Level 1-Questions and Answers

	Field Name	Data Type
▶	qno	Number
	question	Text
	option1	Text
	option2	Text
	option3	Text
	option4	Text
	ans	Text
	rightans	Text

Level 2: Questions and answers

	Field Name	Data Type
▶	qno	Number
	question	Text
	option1	Text
	option2	Text
	option3	Text
	option4	Text
	ans	Text
	rightans	Text

UI LAYER AND DOMAIN LAYER : do Project Coding in specific Language as per the design

RESULT:

The above program was executed successfully

Ex. No: 8	Project Testing - QUIZ SYSTEM
Date:	

Aim: To test Software System for all scenarios identified as given in use case diagram of Quiz System

Description

Identify the usecases given in the use case diagram and prepare the test report for each use case by giving input , expected output , actual output and result. Prepare the Test report by executing the project and giving all possible inputs and testing . The sample Test Report is given below.

Test case Template

Test Report : Quiz_001

Product : Quiz System **Date** : 2/8/2019 **Tester** : XXX

Use Case : Login & Update Student Information

S.No	Input	Expected Result	Actual Result	Pass/Fail
1	Give New User Option	Enters into Registration Mode , System ready to take userid ,pwd	System ready to take userid ,pwd	Pass
2	Enter user id " Ram" Password "Ram" Confirm the password	Creates the new user	New user Successfully created	Pass
3

Test Report : Quiz-002

Product : Quiz System **Date** : 2/8/2019 **Tester** : XXX

UseCase : Play Level 1 / Level 2

S.No	Input	Expected Result	Actual Result	Pass/Fail
1.	Select the suitable option for the displayed question.	The ans is accepted and next question is displayed	-do-	Pass
2.

Test Report : Quiz-003**Product** : Quiz System**Date** : 2/8/2019**Tester** : XXX**UseCase** : Display Question

S.No	Input	Expected Result	Actual Result	Pass/Fail
1.	User login id " Ram" Password "Ram"	Question is displayed	-do-	Pass
2.

Test Report : Quiz-004**Product** : Quiz System**Date** : 2/8/2019**Tester** : XXX**UseCase** : Submit Answer

S.No	Input	Expected Result	Actual Result	Pass/Fail
1.	User select the submit option	System calculate the score and display	-do-	Pass
2.

Test Report : Quiz-005**Product** : Quiz System**Date** : 2/8/2019**Tester** : XXX**UseCase** : Display Score

S.No	Input	Expected Result	Actual Result	Pass/Fail
1.	User select the display score option	System display the score	-do	Pass
2.

RESULT:

The above program was executed successfully

Ex. No: 9	Applying Design Patterns , Implementing & Testing - QUIZ SYSTEM
Date:	

Aim : To improve the reusability and maintainability apply design pattern in Quiz System , Change the design ,implement and test the system

How to Apply Design Patterns :

Design patterns can speed up the development process by providing tested, proven development paradigms. Effective software design requires considering issues that may not become visible until later in the implementation. There are 3 types of design patterns.

1. Creational design patterns

These design patterns are all about class instantiation. This pattern can be further divided into class-creation patterns and object-creational patterns. While class-creation patterns use inheritance effectively in the instantiation process, object-creation patterns use delegation effectively to get the job done.

- **Abstract Factory**
Creates an instance of several families of classes
- **Builder**
Separates object construction from its representation
- **Factory Method**
Creates an instance of several derived classes
- **Object Pool**
Avoid expensive acquisition and release of resources by recycling objects that are no longer in use
- **Prototype**
A fully initialized instance to be copied or cloned
- **Singleton**
A class of which only a single instance can exist

2. Structural design patterns

These design patterns are all about Class and Object composition. Structural class-creation patterns use inheritance to compose interfaces. Structural object-patterns define ways to compose objects to obtain new functionality.

- **Adapter**
Match interfaces of different classes
- **Bridge**
Separates an object's interface from its implementation

- **Composite**
A tree structure of simple and composite objects
- **Decorator**
Add responsibilities to objects dynamically
- **Facade**
A single class that represents an entire subsystem
- **Flyweight**
A fine-grained instance used for efficient sharing
- **Proxy**
An object representing another object

3. Behavioral design patterns

These design patterns are all about Class's objects communication. Behavioral patterns are those patterns that are most specifically concerned with communication between objects.

- **Chain of responsibility**
A way of passing a request between a chain of objects
- **Command**
Encapsulate a command request as an object
- **Interpreter**
A way to include language elements in a program
- **Iterator**
Sequentially access the elements of a collection
- **Mediator**
Defines simplified communication between classes
- **Memento**
Capture and restore an object's internal state
- **Null Object**
Designed to act as a default value of an object
- **Observer**
A way of notifying change to a number of classes
- **State**
Alter an object's behavior when its state changes
- **Strategy**
Encapsulates an algorithm inside a class
- **Template method**
Defer the exact steps of an algorithm to a subclass
- **Visitor**
Defines a new operation to a class without change

Apply the relevant design pattern in quiz system , change the design , implementation and test the system.

MINI PROJECT

Ex. No: 1	PASSPORT AUOMATION SYSTEM
Date:	

AIM:

To Design, Implement and Test the Passport Automation described in the given problem statement

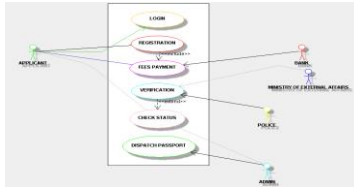
PROBLEM STATEMENT:

Passport Automation System is used in the effective dispatch of passport to all the applicants. This system adopts a comprehensive approach to minimize the work load of both the applicant and the persons who are involved in verification process. The main aim of the system to create a online registration form that includes details about the applicant like name, permanent address and etc along with the proof are being uploaded in it for verification. The amount must be paid to government after all the personal details being uploaded.

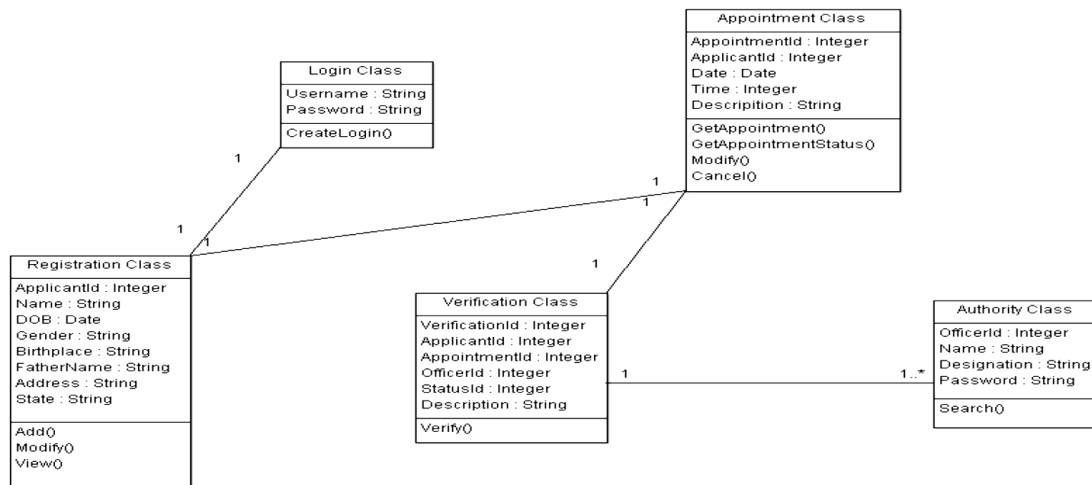
After testing the genuineness of the documents that are being uploaded by the applicant in the passport automation system and the information is sent to regional administrator's office. If all the documents that have been provided are genuine an appointment date is sent as a message to the applicant for further verification of the documents and also a person is assigned to check for personal verification. While verification if any forfeiting is identified, the applicant is liable for penalty and further action will be taken.

After the passport being issued its validity is for 10 years and for renewing process a request must be give to the regional administrator's office and a new passport with validity for the next 5 years will be provided after the above process are done for verification.

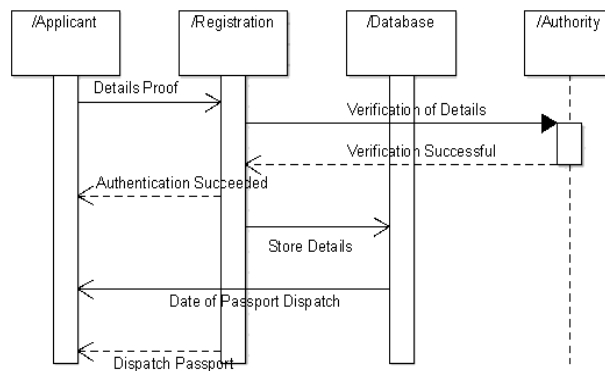
USE CASE DIAGRAM



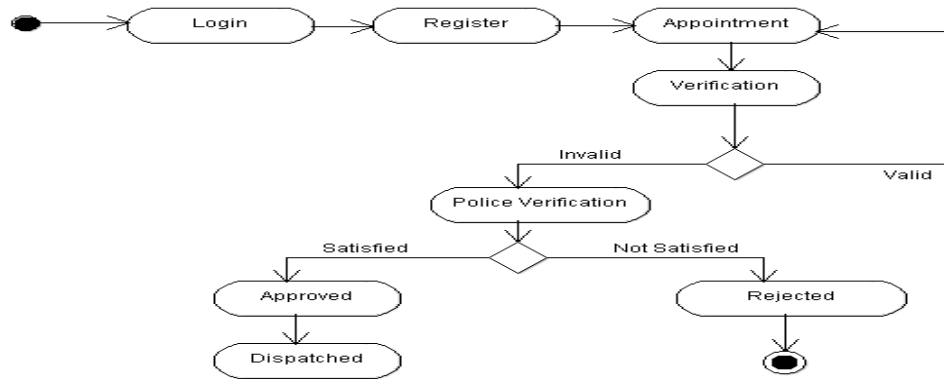
CLASS DIAGRAM



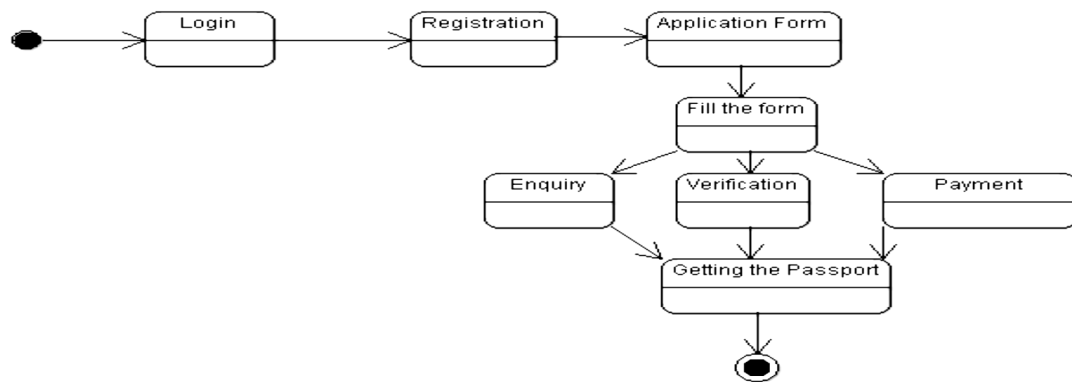
SEQUENCE DIAGRAM:



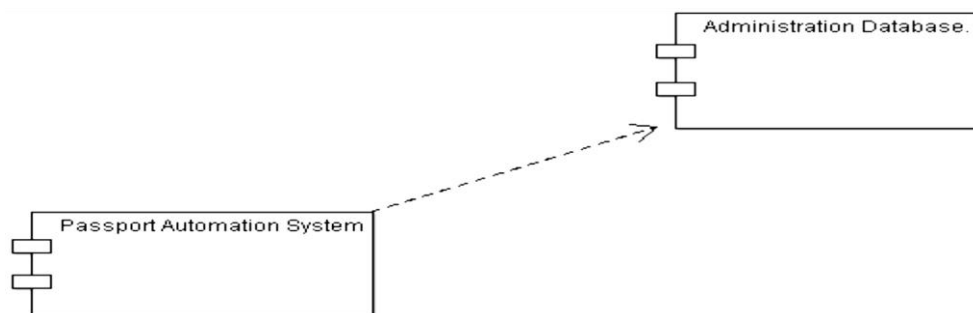
ACTIVITY DIAGRAM



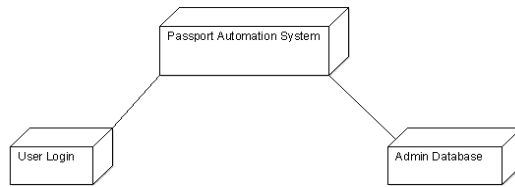
STATE CHART DIAGRAM



COMPONENT DIAGRAM



DEPLOYMENT DIAGRAM



IMPLEMENTATION AND TESTING: PASSPORT AUTOMATION SYSTEM

Person.java

```
package passport;
import passport.Police;
import java.util.*;
class Person
{
public static void main(String args[])
{ Scanner sc= new Scanner(System.in);
  int e ;
  do
  {
    System.out.println("*****passort automation system*****");
    System.out.println(" 1.Apply passport\n 2.Police verification \n 3.passport generation");
    int n = sc.nextInt();
    switch(n)
    {
    case 1: Passport p = new Passport();
      System.out.println(" wlcome passport application !");
      p.apply();
      break;
    case 2: Police g = new Police();
      System.out.println("passport verification");
      g.verify( );
      break;
    case 3 : Passport p1 = new Passport();
      p1.generate();
      break;
    default :
      System.out.println("kindly do the previous operations");

    }
    System.out.println("press 1 for home page");
    e = sc.nextInt();
  }while(e ==1);
}}
```

Passport.java

```

package passport;
import static java.lang.System.exit;
import passport.Police;
import java.util.Random;
import java.util.*;
class Passport {
    public static String name ,gender,mailId,x ;
    public static String id,DOB;
    public static long phone_no;
    public static int c;
    Scanner s1 = new Scanner(System.in);
    public void apply()
    {
        System.out.println( " Enter the your name " );
        name = s1.next();
        System.out.println( " Enter the your gender " );
        gender = s1.next();
        System.out.println( " Enter the your birth date (dd/mm/yyyy) " );
        DOB = s1.next();
        System.out.println( " Enter the your Mail Id " );
        mailId = s1.next();
        System.out.println("Eneter your Aadhar Id (12 digit) number");
        id = s1.next();
        int i=id.length();
        while(i!=12)
        {
            System.out.println("Renter the id .");
            id = s1.next();
            i= id.length();
        }
        System.out.println("Eneter your mobile number");
        phone_no = s1.nextLong();
        System.out.println("your application id is been generating");
        Random r = new Random();
        c = (r.nextInt( 0x171)+ 1000);
        System.out.println("application id is "+c);

        System.out.println("\n \n Press y to proceed next section ...\n");
        String f = s1.next();
        if( "y".equals(f))
        {
            Police g = new Police();
            g.verify( );
        }
        x = ("A54"+Integer.toString(r.nextInt()));
    }
    public void generate(){
        System.out.println("*****passport generation*****");
    }
}

```

```

System.out.println("****PASSPORT ****");
System.out.println("*** GOV OF INDIA ***");
System.out.println("PASSPORT NO :"+x);
System.out.println("NAME      :"+name);
System.out.println("DOB       :"+DOB);
System.out.println("GENDER    :"+gender);
System.out.println("MAILID    :"+mailId);
System.out.println("PH NO     :"+phone_no);
exit(0);
}

}

```

Police.java

```

package passport;
import java.util.Scanner;
class Police {
    public void verify()
    { Scanner s2 = new Scanner(System.in);
      System.out.println("\n Police verifying your passport application ...");
      System.out.println("Enter your name");
      String sname=s2.next();
      System.out.println("\n Enter your application id to complete verification");
      int i = s2.nextInt();

      if((i!=Passport.c)&&(sname.equals(Passport.name)))
      {
          System.out.println("unauthenticate user");
      }
      else
      {
          System.out.println("verified sucessfully");
          System.out.println("press y to proceed next section ...");
          String f = s2.next();
          if( "y".equals(f))
          {
              Passport p1 =new Passport();
              p1.generate( );
          }
      }
    }
}

```

OUTPUT

run:

*****passport automation system*****

- 1.Apply passport
- 2.Police verification
- 3.passport generation

1

welcome passport application !

Enter the your name

Manish

Enter the your gender

Male

Enter the your birth date (dd/mm/yyyy)

12/09/1999

Enter the your Mail Id

manishraghu@yahoo.com

Enter your Aadhar Id (12 digit) number

740550346166

Enter your mobile number

7358682980

your application id is been generating

application id is 1169

Press y to proceed next section ...

y

Police verifying your passport application ...

Enter your name

Manish

Enter your application id to complete verification

1169

verification successful

press y to proceed next section ...

n

press 1 for home page

1

*****passport automation system*****

- 1.Apply passport
- 2.Police verification
- 3.passport generation

*****passport generation*****

****PASSPORT ****

** GOV OF INDIA **

PASSPORT NO :A54-1300338742

NAME : Manish

DOB : 12/09/1999

GENDER :Male

MAILID :manishmoni1209@gmail.com

PH NO :358682980

BUILD SUCCESSFUL (total time: 2 minutes 3 seconds)

TEST REPORT 1

Product : Passport Automation System

Use Case : Applying Passport

Test Case ID	Test Case / Action To Perform	Expected Result	Actual Result	Pass/Fail
1.	To apply for passport	Press 1 for passport application	Displays details on what to be uploaded.	Pass
2.	After Entering '1' for Passport Application	Enter the details	After entering you can proceed to next section	Pass
3.	Press 'Y' to move onto next section	You can check whether the Passport is verified or not.	Details about your passport verification.	Pass
4.	Press '1' to move to home page	To go to home page	Displays home page	Pass
5.	Press '3' for Passport Generation	Passport details	Your passport has been generated with your passport id.	Pass

Ex. No: 2

BOOK BANK SYSTEM

Date:

BOOK BANK SYSTEM

AIM:

To Design ,Implement and Test the BOOK BANKSYSTEM described in the given problem statement

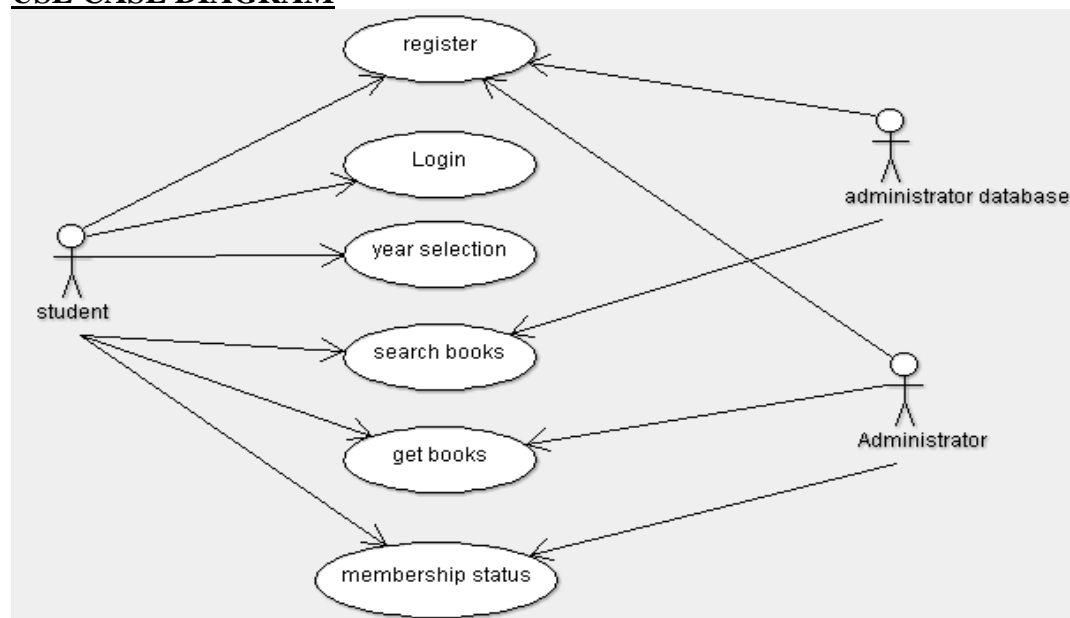
PROBLEM STATEMENT:

The process of members registering and purchasing books from the book bank are described sequentially through following steps:

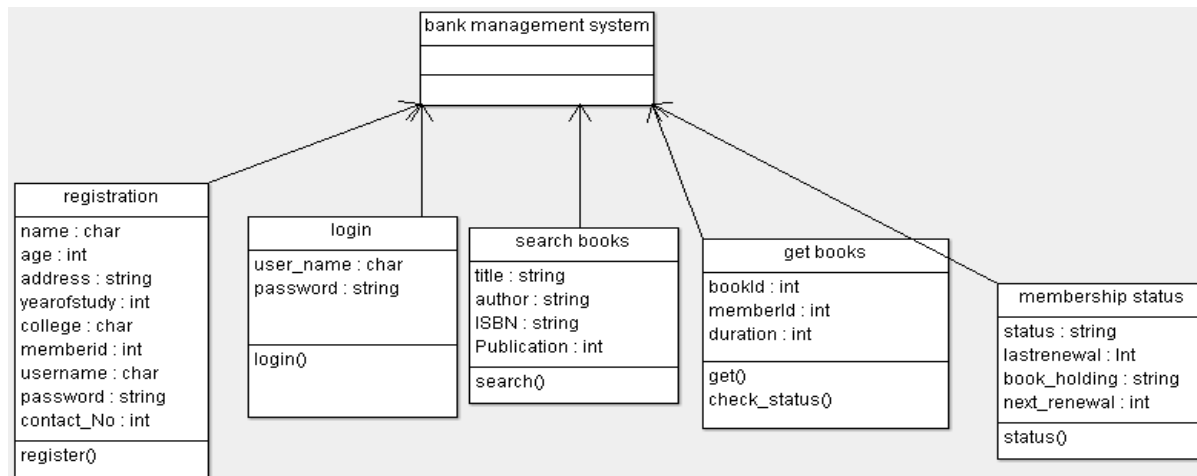
First the member registers himself if he was new to the book bank. Old members will directly select old member button.They select their corresponding year. After selecting the year they fill the necessary details and select the book and he will be directed towards administrator. The administrator will verify the status and issue the book.

The book bank management system is a software in which a member can register themselves and then he can borrow books from the book bank. It mainly concentrates on providing books for engineering students. This system would be used by members who are students of any college to check the availability of the books and borrow the books, and then the databases are updated. The purpose of this document is to analyze and elaborate on the high-level needs and features of the book bank management system. It also tells the usability, reliability defined in use case specification.

USE CASE DIAGRAM

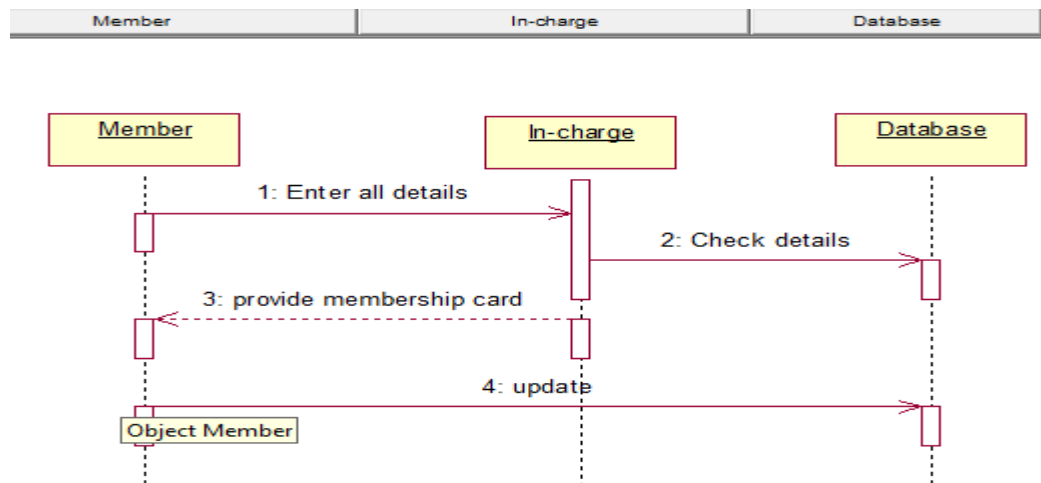


CLASS DIAGRAM

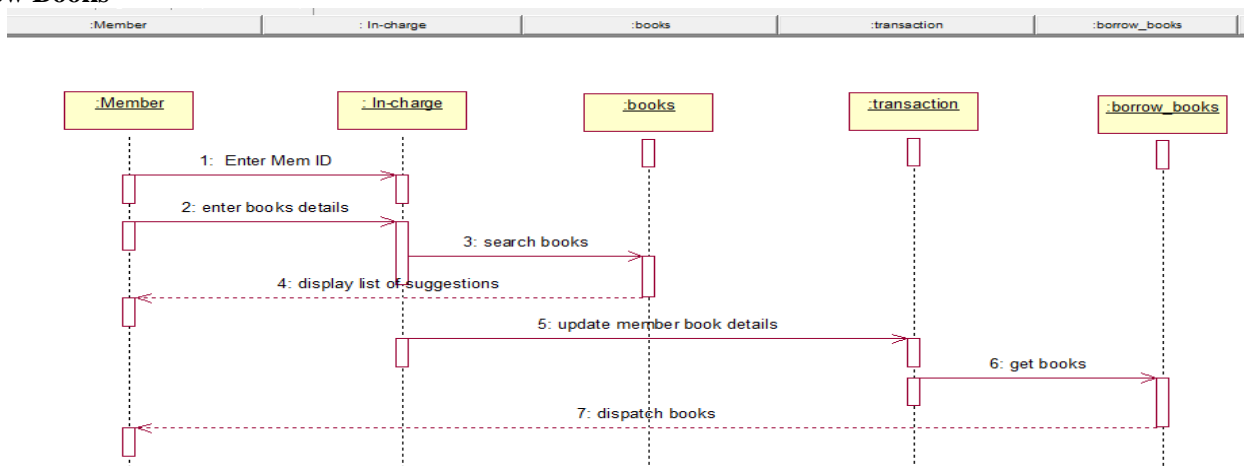


SEQUENCE DIAGRAM:

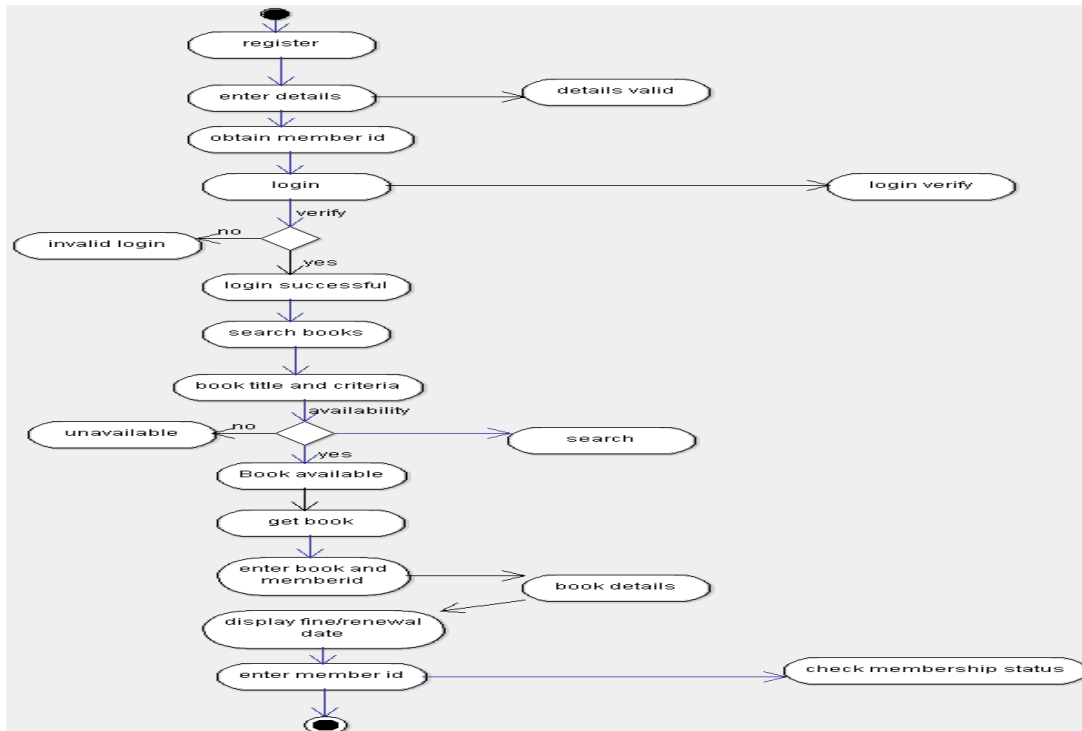
Registration



Borrow Books

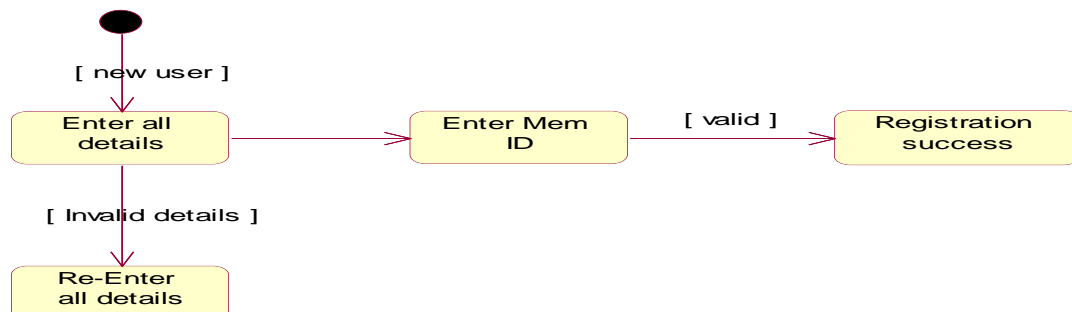


ACTIVITY DIAGRAM

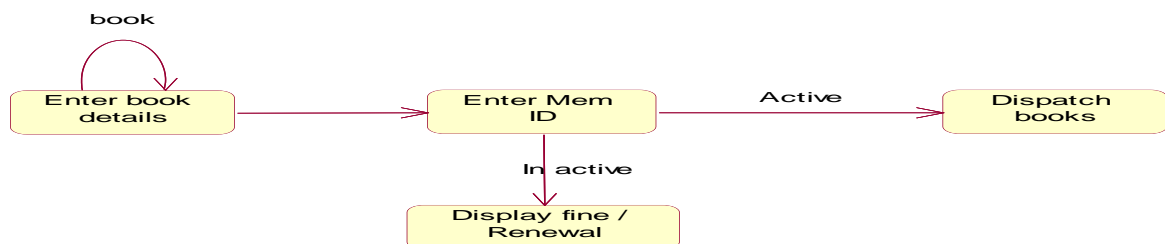


STATE CHART DIAGRAM

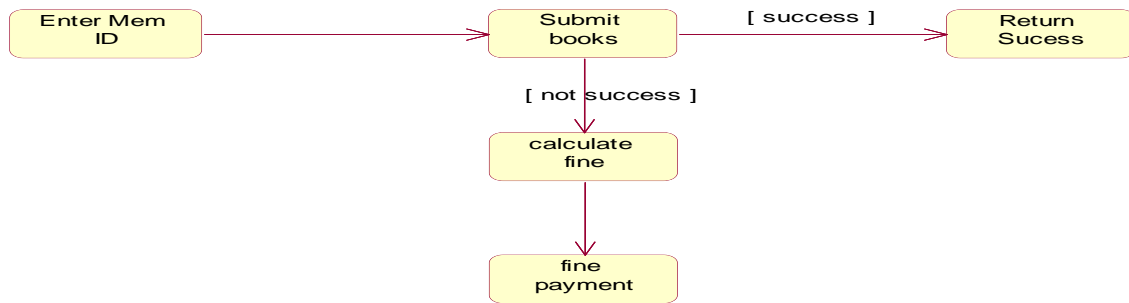
Registration



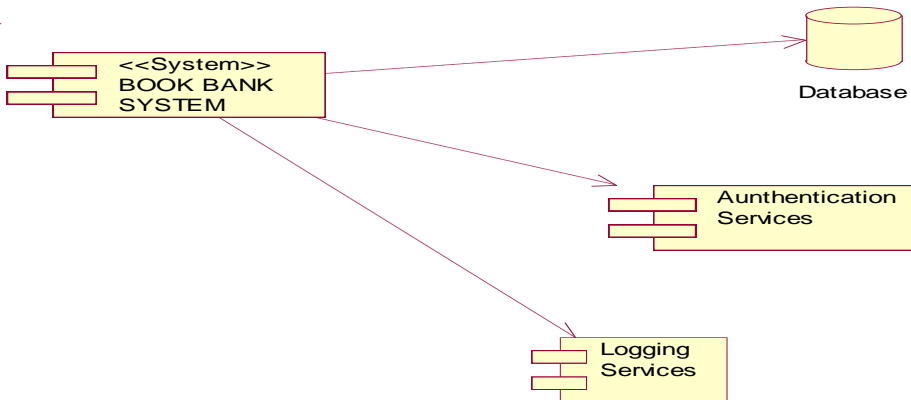
Borrow



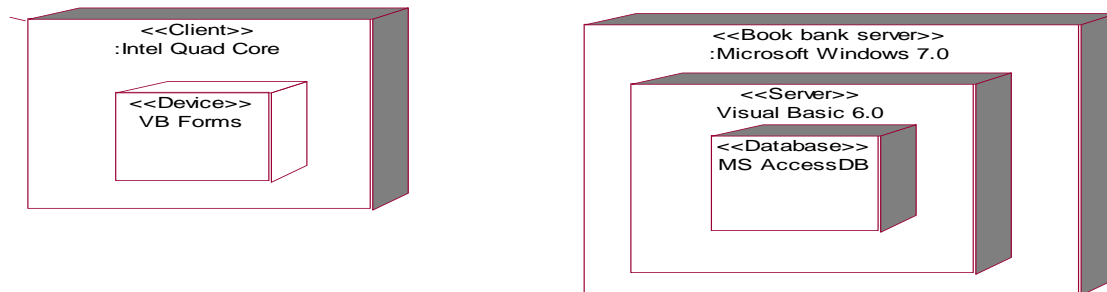
Return



COMPONENT DIAGRAM



DEPLOYMENT DIAGRAM



IMLEMENTATION AND TESTING:

Admin.java

```

package Bookbank;
import java.util.Scanner;
import java.io.*;

```

```

class Admin extends Book
{
    public void operation()
    {
        int a;
        do
        {

```

```

        System.out.println("select an option");
        System.out.println("1.Admin login\t2.Member login\t3.exit");
        Scanner s=new Scanner(System.in);
        int m=s.nextInt();
        switch(m)
        {
            case 1:
            {
                Book p=new Book();
                p.managebook();
                break;
            }
            case 2:
            {
                Member c=new Member();
                c.order();
                break;
            }
            case 3:
                System.exit(0);

        }

        System.out.println("enter 1 to continue to select login");
        a=s.nextInt();
    }while(a==1);
}
}

```

Book.java

```

package Bookbank;

import java.util.*;
import java.io.*;
class Book extends Member
{
    int t;int f;int i=0;int count=0,count1=0;
    public static int[] bid=new int[100];
    public static String[] bname=new String[100];
    public static int[] qty=new int[100];
    public static int[] ord=new int[100];
    public void managebook()
    {
        Scanner s=new Scanner(System.in);

        do
        {

```

```
System.out.println("1.Add\n 2.display\n 3.delete\n 4.update\n");
System.out.println("enter the option");
```

```
int n=s.nextInt();
```

```
switch(n)
```

```
{
```

```
case 1:
```

```
System.out.println("enter the number of Books to be added");
```

```
int no=s.nextInt();
```

```
System.out.println("enter the id,name,qty of the Book");
```

```
no=no-1;
```

```
int f=i+no;
```

```
for(;i<=f;i++)
```

```
{
```

```
    bid[i]=s.nextInt();
```

```
    bname[i]=s.next();
```

```
    qty[i]=s.nextInt();
```

```
}
```

```
i=i;
```

```
System.out.println("Books added successfully");
```

```
break;
```

```
case 2:
```

```
System.out.println("enter the number of Books to be printed");
```

```
int scan=s.nextInt();
```

```
for(int i=0;i<scan;i++)
```

```
{
```

```
    System.out.println("B_id   B_name  qty  ");
```

```
    System.out.println(bid[i]+"\\t"+bname[i]+"\\t"+qty[i]);
```

```
}
```

```
System.out.println("The Books  informations are displayed");
```

```
break;
```

```
case 3:
```

```
System.out.println("enter the id of the Book to be deleted");
```

```
int did=s.nextInt();
```

```
for(int i=0;i<=bid.length-1;i++)
```

```
{
```

```
    if(bid[i]!=0)
```

```
    {
```

```
        count++;
```

```

    }
}

for(i=0;i<=count+1;i++)
{
    if(bid[i]==did)
    {
        bid[i]=bid[count-1];
        bname[i]=bname[count-1];
        qty[i]=qty[count-1];
        bid[count-1]=bid[bid.length-1];
        bname[count-1]=bname[bname.length-1];
        qty[count-1]=qty[qty.length-1];

        break;
    }

}

System.out.println("Boook is successfully deleted");
break;
case 4:
    System.out.println("enter the id,name,qty of the Book to be added");
    int nid=s.nextInt();
    String nname=s.next();
    int nqty=s.nextInt();
    for(int i=0;i<=bid.length;i++)
    {
        if(bid[i]==nid)
        {
            bid[i]=nid;
            bname[i]=nname;
            qty[i]=nqty;
            break;
        }
    }
    System.out.println("Book is updated successfully");
    break;
case 5:
    System.exit(0);

}

System.out.println("enter 1 to continue to manage Books");
t=s.nextInt();
}while(t==1);

}

public static void main(String args[])
{

```

```

        Admin sp=new Admin();
        sp.operation(); }
public void delete(int n)
{
    int did=n;
    for(int i=0;i<=bid.length-1;i++)
    {
        if(i!=0)
        {
            count1++;
        }
    }
    for(int i=0;i<=count1;i++)
    {
        if(bid[i]==did)
        {
            qty[i]=qty[count1];
            break;
        }
    }
}
}

```

Member.java

```

package Bookbank;

import java.util.*;
import java.io.*;
class Member
{
    public void order()
    {
        Scanner s=new Scanner(System.in);
        Book p=new Book();
        System.out.println("enter the Book id to be ordered");
        int search=s.nextInt();
        p.delete(search);
        System.out.println("Book ordered successfully");
        System.out.println("Enter 1 to make payment");
        int b=s.nextInt();
        if(b==1)
        {
            Member c=new Member();
            c.makepayment();
        }
    }
    public void makepayment()
    {
        System.out.println("payment successfull");
    }
}

```

OUTPUT:

Select an option

1. Admin login 2.Member login 3.exit

1

1.Add

2.display

3.delete

4.update

enter the option

1

enter the number of items to be added

3

enter the id, name, qty of the product to be added

1

HarryPotter

3

2

DaVinciCode

4

3

TheAlchemist

5

Items added successfully

enter 1 to continue to manage product

2

enter 1 to continue to select login

1

select an option

1. Admin login 2.Member login 3.exit

1

1.Add

2.display

3.delete

4.update

enter the option

2

enter the number of items to be printed

3

pid	pname	qty
1	HarryPotter	3

pid	pname	qty
2	DaVinciCode	4

pid	pname	qty
3	TheAlchemist	5

the product information is displayed

enter 1 to continue to manage product

1

- 1.Add
- 2.display
- 3.delete
- 4.update

enter the option

3

enter the id of the product to be deleted

1

product is successfully deleted

enter 1 to continue to manage product

1

- 1.Add
- 2.display
- 3.delete
- 4.update

enter the option

4

enter the id , name , qty of the product to be added

2

HarryPotter

2

product is updated successfully

enter 1 to continue to manage product

1

- 1.Add
- 2.display
- 3.delete
- 4.update

enter the option

2

enter the number of items to be printed

2

pid	pname	qty
3	TheAlchemist	5

pid	pname	qty
2	HarryPotter	2

the product information is displayed

enter 1 to continue to manage product

1

- 1.Add
- 2.display
- 3.delete
- 4.update

enter the option

1
enter the number of items to be added
1
enter the id , name , qty of the product to be added
1
DaVinciCode
6
items added successfully
enter 1 to continue to manage product
0
enter 1 to continue to select login
1
select an option
1.Admin login 2.Member login 3.exit
2
enter the product id to be ordered
1
item ordered successfully
enter 1 to make payment
1
payment successfull
enter 1 to continue to select login
1
select an option
1.Admin login 2.Member login 3.exit
1
1.Add
2.display
3.delete
4.update

enter the option
2
enter the number of items to be printed
2
pid pname qty
1 DaVinciCode 0
pid pname qty
2 HarryPotter 2
the product information is displayed
enter 1 to continue to manage product
1
enter 1 to select login
select an option
1.Admin login 2.Member login 3.exit
3
Build successfull...

TEST REPORT 1

Product : Book bank System
Use Case :manage book

Test Case ID	Test Case / Action To Perform	Expected Result	Actual Result	Pass/Fail
1.	After Entering the Book id,name,qty	Displays ”Books added successfully”	Books added successfully	Pass
2.	After Entering the Book ID to be deleted	Displays “book is successfully deleted”	Book is successfully deleted	Pass

TEST REPORT 2

Product :Book Bank System
Use Case : manage book

Test Case ID	Test Case / Action To Perform	Expected Result	Actual Result	Pass/Fail
1.	After selecting “add” option	Displays “Enter the number of books to be added”	Enter the number of books to be added. 3	Pass
2.	After selecting “update” option	Displays “Enter the id,name,qty” of the book to be updated	Enter the id,name,qty of the book to be updated. 3 DaVinciCode 7	Pass

TEST REPORT 3

Product :Book Bank System
Use Case :Make Payment

Test Case ID	Test Case / Action To Perform	Expected Result	Actual Result	Pass/Fail
1.	After books ordered successfully	Displays “Enter 1 to make payment”	Enter 1 to make payment	Pass
2.	After Entering ‘1’ to make payment	Displays “Payment successful”	Payment successful	Pass

Ex. No: 3	STOCK MAINTENANCE SYSTEM
Date:	

AIM:

To Design, Implement and Test the Stock Maintenance System described in the given problem statement .

PROBLEM STATEMENT:

Stock Maintenance System is a real time application used in the merchants day to day system. This is a database to store the transaction that takes places between the manufacturer,dealer and the shop keeper that includes stock inward and stock outward with reference to the dealer.

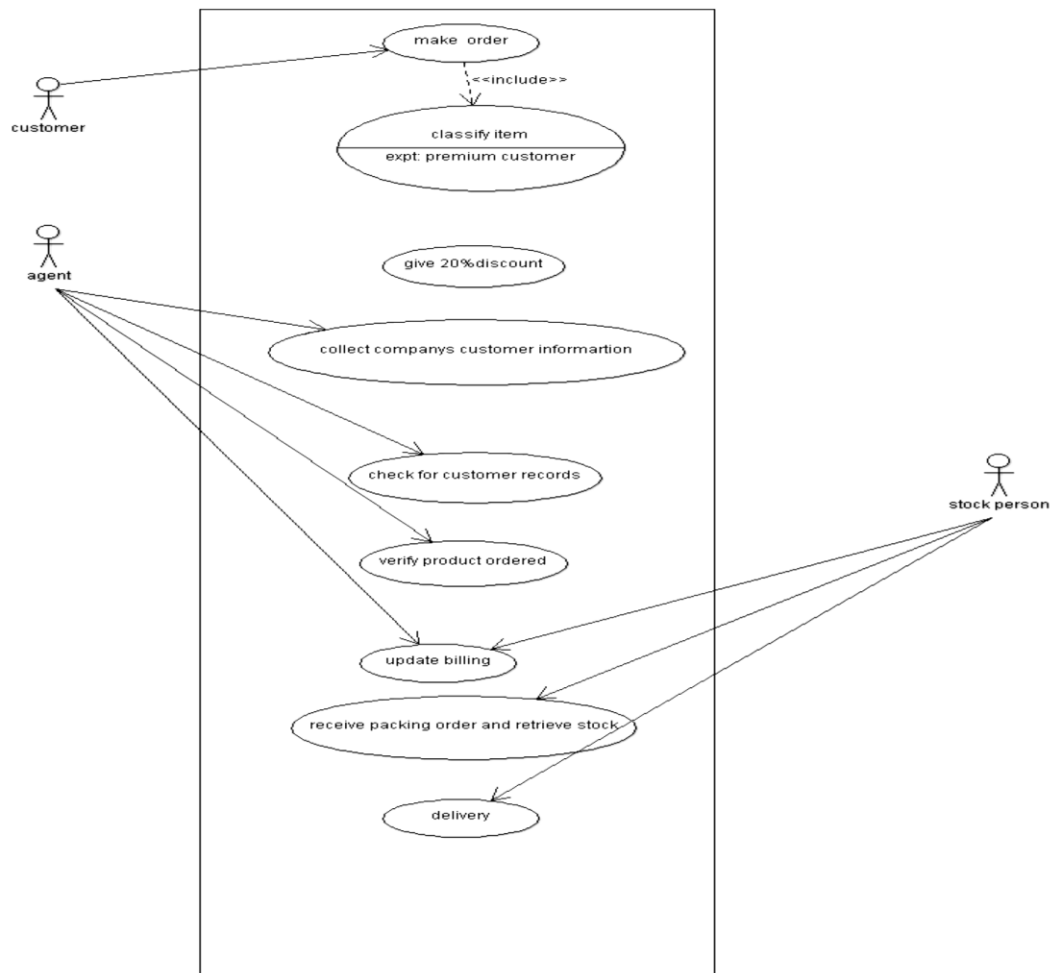
Here we assume ourself as the dealer and proceed with the transaction as follows

- The manufacturer is the producer of the items and it contains the necessary information of the item such as price per item, Date of Manufacture, Best Before Use, Number of item available and their company address.
- The dealer is the secondary source of an item and he purchases item from the manufacture by requesting the required item with its corresponding company name and the number of items required.
- The dealer is only responsible for distribution of the itemn to the retailers in the town of city.
- The shop keeper or retailer is the one who is prime source for selling items in the market.
- The customers get item from the shopkeeper and not directly from the manufacture or the dealer.
- The stock is the database used in our system which records all transactions that takes place between the manufacture and the dealer and the dealer and the retailer.

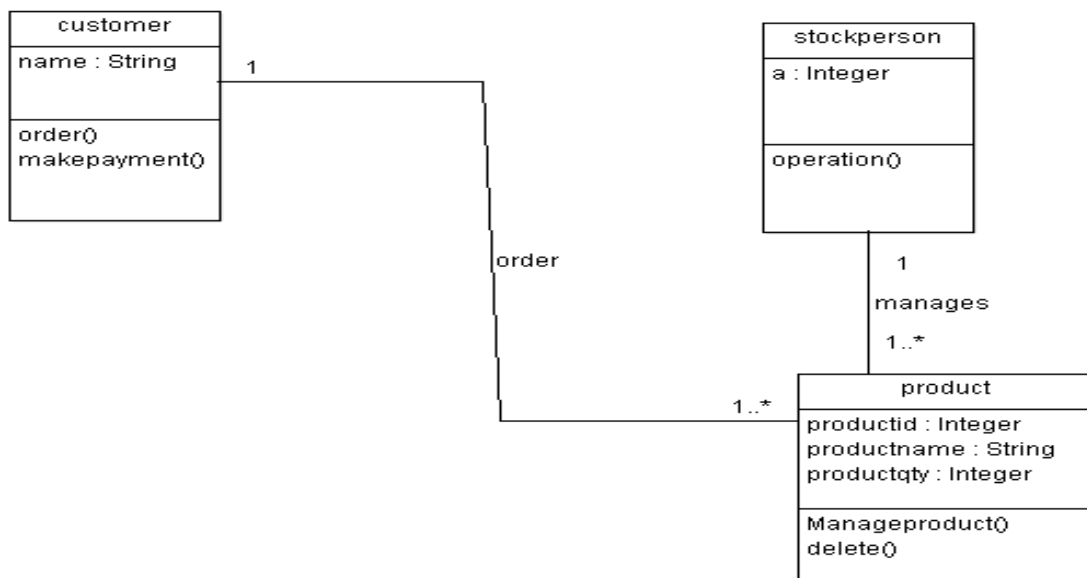
The process of stock maintenance system is that the customer login to the particular site to place that the order for the customer product. The stock maintenance system are described sequentially through steps.

- a) The customer login to the particular site
- b) They fill the customer details
- c) They place the orders for their product.
- d) The vendor login and views the customer details and order

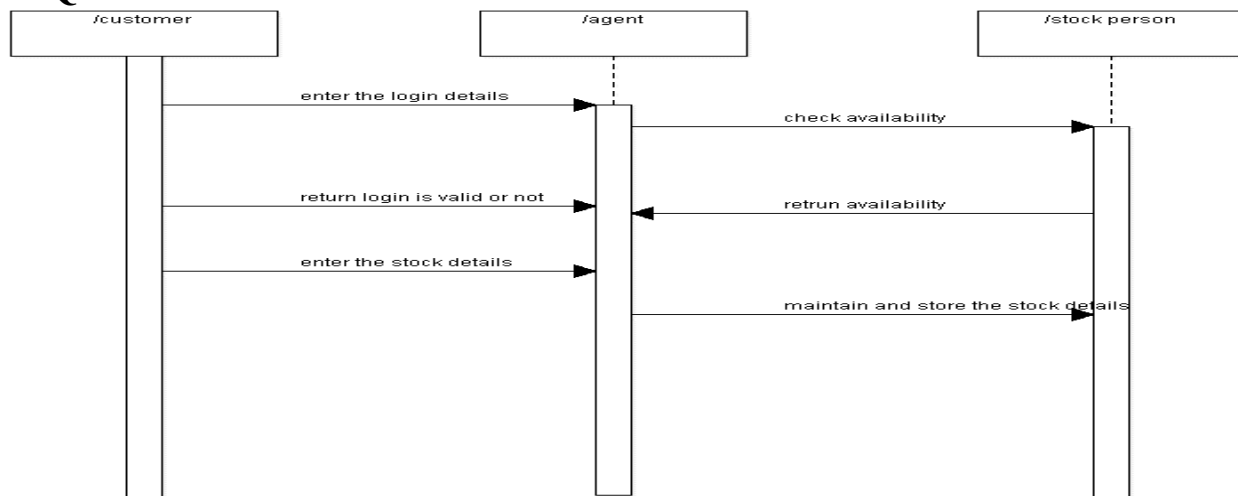
USECASE DIAGRAM:



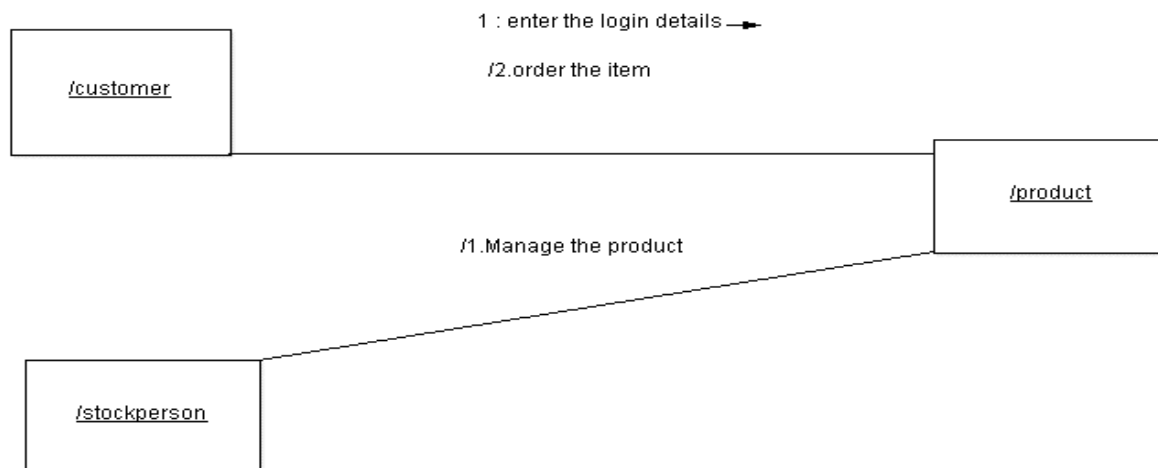
CLASS DIAGRAM:



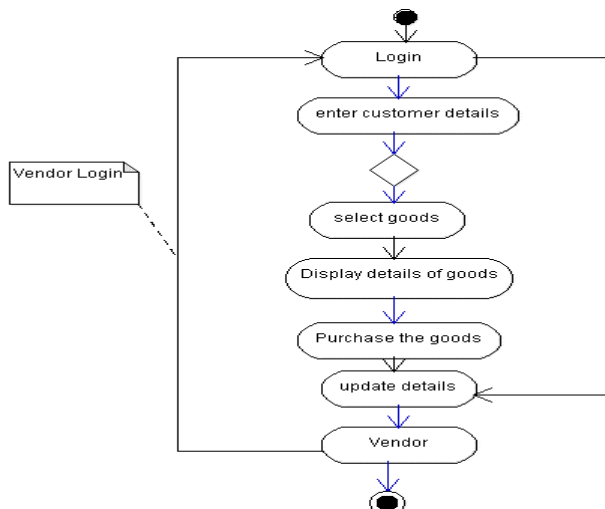
SEQUENCE DIAGRAM:



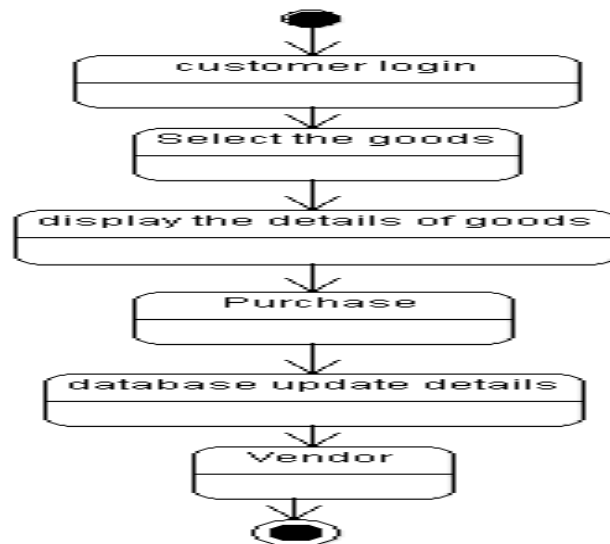
COLLABORATION DIAGRAM:



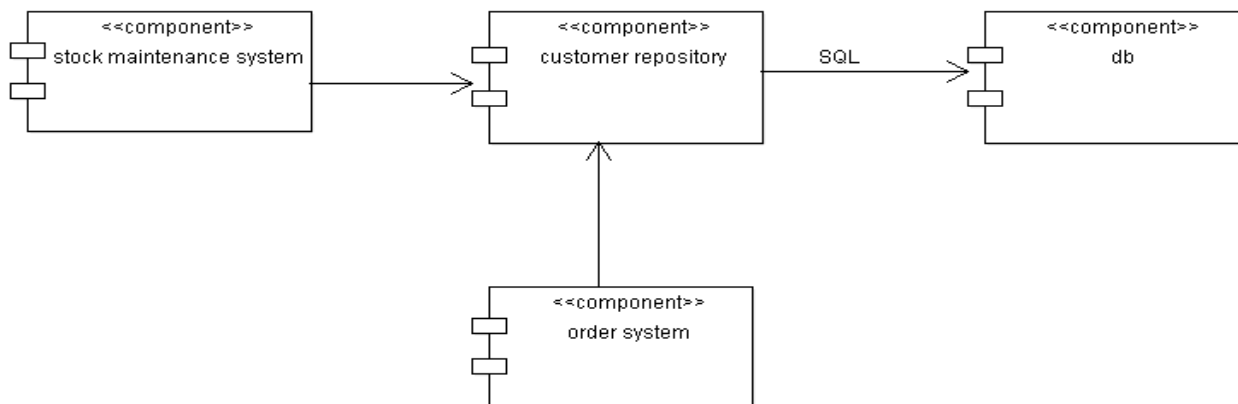
ACTIVITY DIAGRAM:



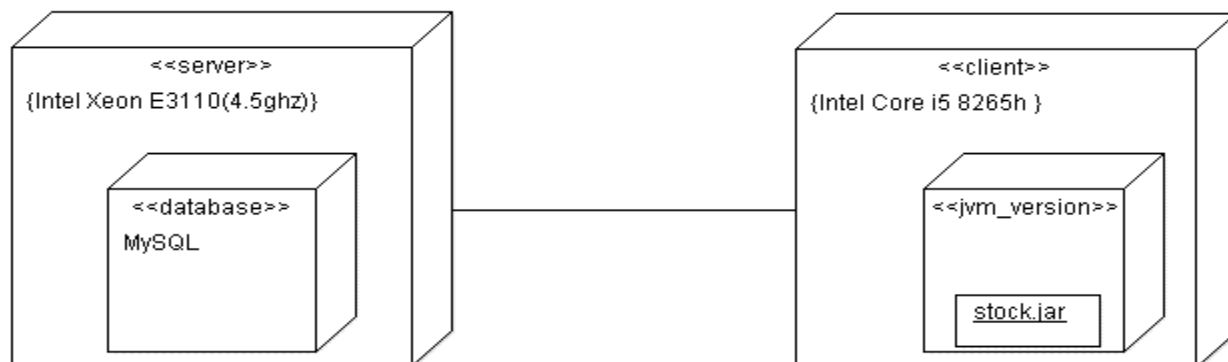
STATE CHART DIAGRAM:



COMPONENT DIAGRAM:



DEPLOYMENT DIAGRAM:



IMPLEMENTATION & TESTING

Product.java:

```

package product;
import java.util.*;
import java.io.*;
class Product extends Customer
{
    int t; int f; int i=0; int count=0, count1=0;
    public static int[] pid=new int[100];
    public static String[] pname=new String[100];
    public static int[] qty=new int[100];
    public static int[] ord=new int[100];
    public void manageproduct()
    {
        Scanner s=new Scanner(System.in);
        do {
            System.out.println("1.Add\n 2.display\n 3.delete\n 4.update\n");
            System.out.println("enter the option");
            int n=s.nextInt();
            switch(n)
            {
                case 1:
                    System.out.println("enter the number of items to be added");
                    int no=s.nextInt();
                    System.out.println("enter the id,name,qty of the product to be added");
                    no=no-1;
                    int f=i+no;
                    for(;i<=f;i++)
                    {
                        pid[i]=s.nextInt();
                        pname[i]=s.next();
                        qty[i]=s.nextInt();
                    }
                    i=i;
                    System.out.println("items added successfully");
                    break;
                case 2:
                    System.out.println("enter the number of items to be printed");
                    int scan=s.nextInt();
                    for(int i=0;i<scan;i++)
                    {
                        System.out.println("pidpnameqty  ");
                        System.out.println(pid[i]+"\\t"+pname[i]+"\\t"+qty[i]);
                    }
                    System.out.println("the product information is displayed");
                    break;
                case 3:
                    System.out.println("enter the id of the product to be deleted");
                    int did=s.nextInt();
                    for(int i=0;i<=pid.length-1;i++)
                    {

```



```

if(pid[i]!=0)
    {
count++;
    }
    }
for(i=0;i<=count+1;i++)
    {
if(pid[i]==did)
    {
pid[i]=pid[count-1];
pname[i]=pname[count-1];
qty[i]=qty[count-1];
pid[count-1]=pid[pid.length-1];
pname[count-1]=pname[pname.length-1];
qty[count-1]=qty[qty.length-1];
break;
    }

    }

System.out.println("product is successfully deleted");
break;
case 4:
System.out.println("enter the id,name,qty of the product to be added");
intnid=s.nextInt();
    String nname=s.next();
intnqty=s.nextInt();
for(inti=0;i<=pid.length;i++)
    {
if(pid[i]==nid)
    {
pid[i]=nid;
pname[i]=nname;
qty[i]=nqty;
break;
    }

    }
System.out.println("product is updated successfully");
break;
case 5:
System.exit(0);

    }
System.out.println("enter 1 to continue to manage product");
t=s.nextInt();
}while(t==1);

}
public static void main(String args[])
    {

```

```

        Stockperson sp=new Stockperson();
        sp.operation();

    }
    public void delete(int n)
    {
        int did=n;
        for(inti=0;i<=pid.length-1;i++)
        {
            if(i!=0)
            {
                count1++;
            }
        }
        for(inti=0;i<=count1;i++)
        {

            if(pid[i]==did)
            {
                qty[i]=qty[count1];
                break;
            }

        }
    }
}

```

Stockperson.java

```

package product;
import java.util.Scanner;
import java.io.*;

class Stockperson extends Product
{
    public void operation()
    {
        int a;
        do
        {
            System.out.println("select an option");
            System.out.println("1.stockperson login\t2.Customer login\t3.exit");
            Scanner s=new Scanner(System.in);
            int m=s.nextInt();
            switch(m)
            {
                case 1:
                {
                    Product p=new Product();
                    p.manageproduct();
                }
            }
        }
    }
}

```

```

break;
    }
case 2:
    {
        Customer c=new Customer();
c.order();
break;
    }
case 3:
System.exit(0);

    }
System.out.println("enter 1 to continue to select login");
        a=s.nextInt();
}while(a==1);
    }
}

```

Customer.java

```

package product;
import java.util.*;
import java.io.*;
class Customer
{
public void order()
    {
        Scanner s=new Scanner(System.in);
        Product p=new Product();
System.out.println("enter the product id to be ordered");
int search=s.nextInt();
p.delete(search);
System.out.println("item ordered successfully");
System.out.println("enter 1to make payment");
int b=s.nextInt();
if(b==1)
    {
        Customer c=new Customer();
c.makepayment();

    }
}
public void makepayment()
    {
System.out.println("payment successfull");
    }

}

```

Output:

select an option

1.stockperson login 2.Customer login 3.exit

1

1.Add

2.display

3.delete

4.update

enter the option

1

enter the number of items to be added

3

enter the id,name,qty of the product to be added

1

tea

3

2

coffee

4

3

boost

5

items added successfully

enter 1 to continue to manage product

2

enter 1 to continue to select login

1

select an option

1.stockperson login 2.Customer login 3.exit

1

1.Add

2.display

3.delete

4.update

enter the option

2

enter the number of items to be printed

3

pidpnameqty

1 tea 3

pidpnameqty

2 coffee 4

pidpnameqty

3 boost 5

the product information is displayed

enter 1 to continue to manage product

1

1.Add
2.display
3.delete
4.update

enter the option

3

enter the id of the product to be deleted

1

product is successfully deleted

enter 1 to continue to manage product

1

1.Add
2.display
3.delete
4.update

enter the option

4

enter the id,name,qty of the product to be added

2

tea

2

product is updated successfully

enter 1 to continue to manage product

1

1.Add
2.display
3.delete
4.update

enter the option

2

enter the number of items to be printed

2

pidpnameqty

3 boost 5

pidpnameqty

2 tea 2

the product information is displayed

enter 1 to continue to manage product

1

1.Add
2.display
3.delete
4.update

enter the option

1

enter the number of items to be added

1
enter the id,name,qty of the product to be added
1
coffee
6
items added successfully
enter 1 to continue to manage product
0
enter 1 to continue to select login
1
select an option
1.stockperson login 2.Customer login 3.exit
2
enter the product id to be ordered
1
item ordered successfully
enter 1to make payment
1
paymentsuccessfull
enter 1 to continue to select login
1
select an option
1.stockperson login 2.Customer login 3.exit
1
1.Add
2.display
3.delete
4.update

enter the option
2
enter the number of items to be printed
2
pidpnameqty
1 coffee 0
pidpnameqty
2 tea 2
the product information is displayed
enter 1 to continue to manage product
1
enter 1 to select login
select an option
1.stockperson login 2.customer login 3.exit
3
Build successful...

TEST REPORT 1**Product : Stock Maintenance System****Use Case :manageproduct**

Test Case ID	Test Case / Action To Perform	Expected Result	Actual Result	Pass/Fail
1.	After Entering the product id,name,qty	Displays "items added successfully"	Items added successfully	Pass
2.	After Entering the product Id to be deleted	Displays "product is successfully deleted"	Product is successfully deleted	Pass

TEST REPORT 2**Product :Stock Maintenance System****Use Case : manageproduct**

Test Case ID	Test Case / Action To Perform	Expected Result	Actual Result	Pass/Fail
3.	After selecting "add" option	Displays "Enter the number of items to be added"	Enter the number of items to be added. 3	Pass
4.	After selecting "update" option	Displays "Enter the id,name,qty" of the product to be updated	Enter the id,name,qty of the product to be updated. 3 Coffe 7	Pass

TEST REPORT 3**Product : Stock Maintenance System****Use Case :Make Payment**

Test Case ID	Test Case / Action To Perform	Expected Result	Actual Result	Pass/Fail
1	After items ordered successfully	Displays "Enter 1 to make payment"	Enter 1 to make payment	Pass
2	After Entering '1' to make payment	Displays "Payment successful"	Payment successful	Pass

Ex. No: 4	FOREIGN TRADING SYSTEM
Date:	

AIM:

To Design, Implement and Test the Exam Registration System described in the given problem statement.

PROBLEM STATEMENT:

Foreign Trading System involves details about the export and import of materials between two or many countries.

The main objective of this project is to ensure globalization and thereby ensuring that the buyers all over the world get the desired product that helps in the development of their globalization.

This software helps the trading system to be easy for the promoters and the buyers to maintain their exports and import details.

The database contains all the details of the export and import procedures.

The system consists of two databases to maintain this software.

- The import session
- The export session

Export Session:

The promoter has to maintain the details of the products that are been exported to other countries.

The bank's certification details are maintained in database are

- Product name that is to be delivered
- The number of items to be exported
- The date and time of the product to be delivered
- The mode of transport through which the goods are sent
- The details of the transaction

Import Session:

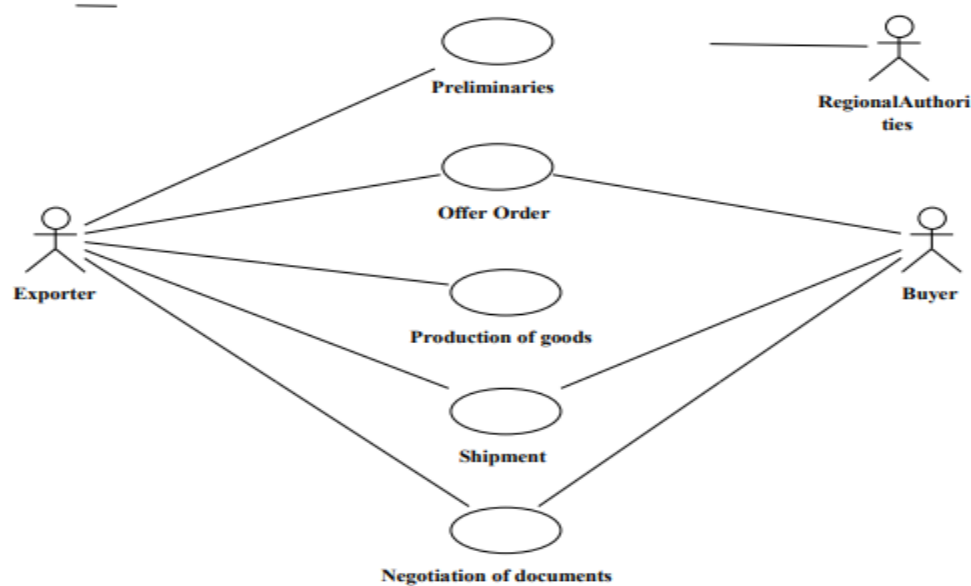
The buyer has to maintain the details of the products brought from the foreign countries.

The bank's certification should be issued to the import products.

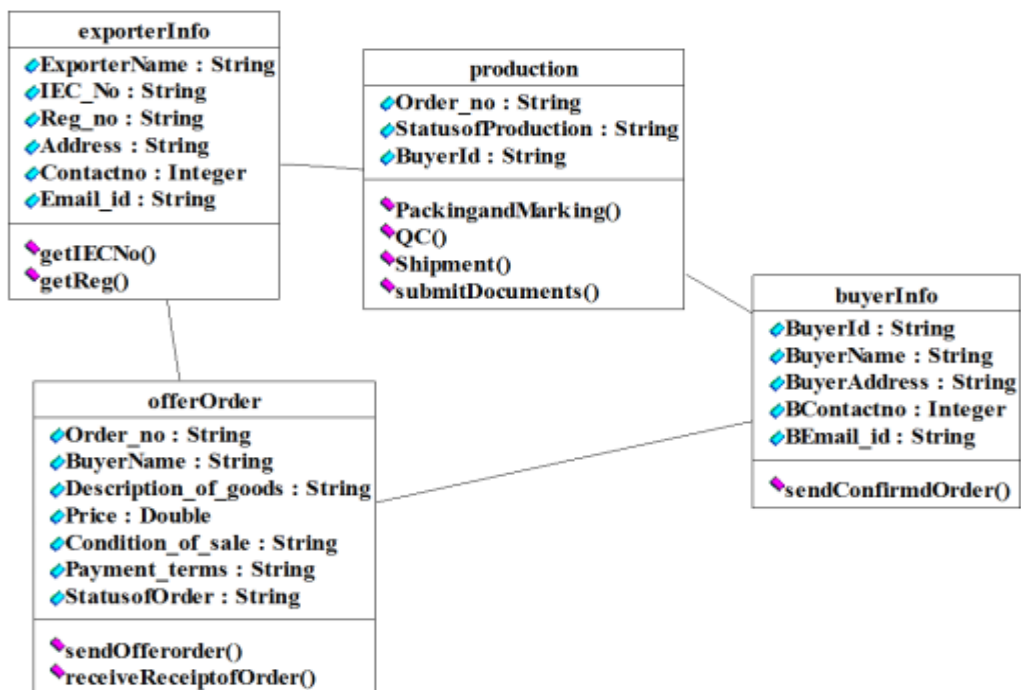
The following details are maintained in database are

- Product name that is to be delivered
- The number of items to be imported
- The date and time of the product to be imported
- The mode of transport through which the goods are received
- The details of the transaction

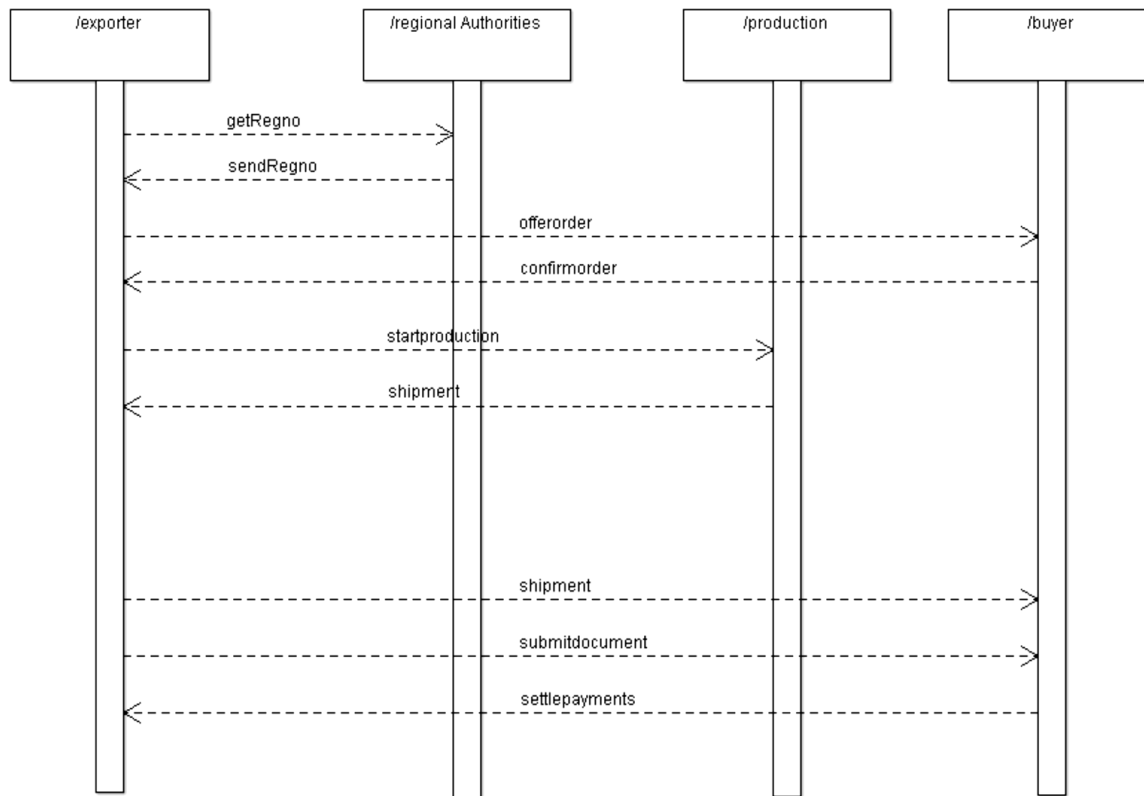
USECASE DIAGRAM:



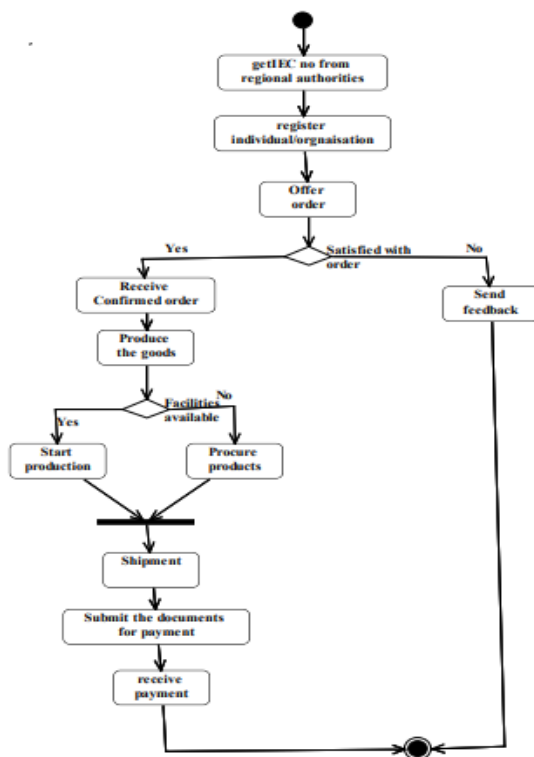
CLASS DIAGRAM:



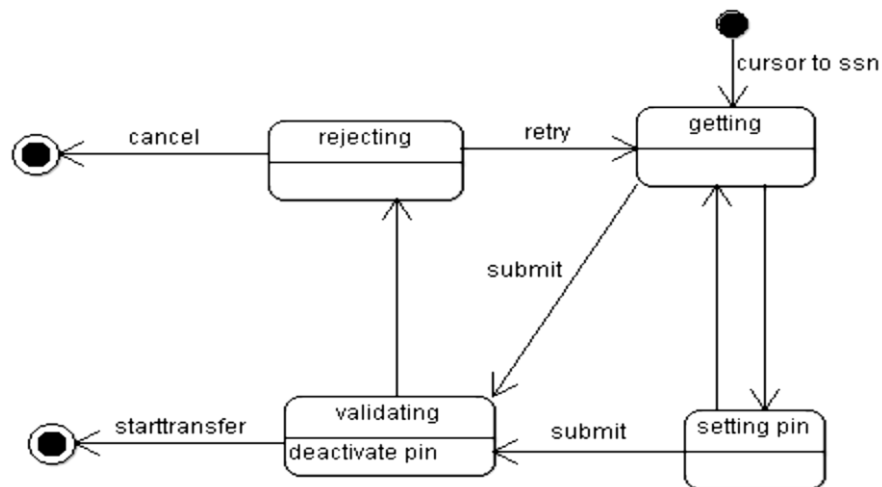
SEQUENCE DIAGRAM:



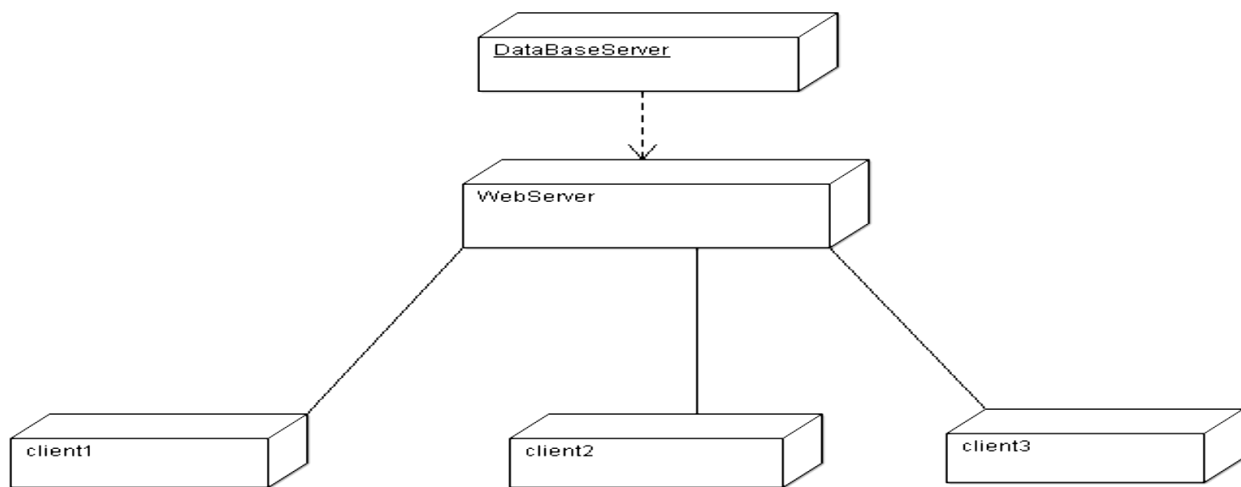
ACTIVITY DIAGRAM:



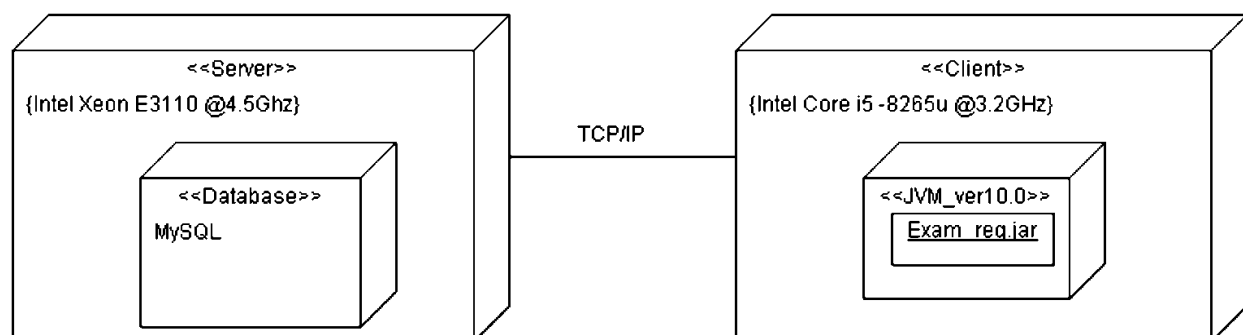
STATE CHART DIAGRAM:



COMPONENT DIAGRAM:



DEPLOYMENT DIAGRAM



IMPLEMENTATION & TESTING

IMPLEMENTATION:

BUYER.java

```
import java.util.*;
class BUYER
{
    String name,country;
    int id;
    int phone;

    void get_info()
    {
        System.out.println("\n*****PROFILE*****");
        System.out.println("\nName : "+name+"\nID : "+id+"\nPhone : "+phone+"\nCountry : "+country);
    }
    void get_payment()
    {
        System.out.println("\n*****Payment*****");
        System.out.println("Transaction made successfully");
    }

    BUYER(String name, String country,intid,int phone)
    {
        this.name=name;
        this.country=country;
        this.id=id;
        this.phone=phone;
    }
}
```

EXPORTER.java

```
class EXPORTER
{
    String name,country;
    int id;
    int phone;

    void get_Offer()
    {
        System.out.println("\n*****OFFERS*****");
        System.out.println("1.Electronics\n2.Food products\n3.Textile");
    }

    void get_Production()
    {
        int rand=(int)(Math.random()*100000000);
        System.out.println("\n*****Shipping*****");
        System.out.println("Shipment Id: "+rand+" has been made");
    }
}
```

```

EXPORTER(String name, String country,intid,int phone)
{
    this.name=name;
this.country=country;
    this.id=id;
this.phone=phone;
}

```

Exchange.java

```

public class exchange {

    public static void main(String[] args) {
        String name,country;
        int id;
        int phone;
        int b,c;
        Scanner sc = new Scanner(System.in);
        int e;
        do
        {
System.out.println("\n*****FORIGN EXCHANGE*****");
System.out.println("SELECT EXCHANGE");
System.out.println("1.EXPORTER \n2.BUYER\n3.EXIT");
        int n = sc.nextInt();
        switch(n)
        {
            case 1 :
System.out.println("Welcome exporter");
System.out.print("\nEnter Your name : ");
                name=sc.nextLine();
                name=sc.nextLine();
System.out.print("\nEnterID : ");
                id=sc.nextInt();
System.out.print("\nEnter Phone no. : ");
                phone=sc.nextInt();
System.out.print("\nEnterCountry : ");
                country=sc.nextLine();
                country=sc.nextLine();
                EXPORTER f = new EXPORTER(name,country,id,phone);
do{
System.out.println("\nMENU");
System.out.println("\n1.Offer order \n2. Production\n3.Quit");
System.out.print("Enter your choice: ");
                b = sc.nextInt();
                switch(b)
                {

                    case 1 :f.get_Offer();

```

```

        break;

        case 2 :f.get_Production();
            break;

    }
}while(b!=3);
    break;

    case 2 :
System.out.println("Welcome Buyer");
System.out.print("\nEnter Your name : ");
    name=sc.nextLine();
    name=sc.nextLine();
System.out.print("\nEnterID : ");
    id=sc.nextInt();
System.out.print("\nEnter Phone no. : ");
    phone=sc.nextInt();
System.out.print("\nEnterCountry : ");
    country=sc.nextLine();
    country=sc.nextLine();
    BUYER d = new BUYER(name,country,id,phone);
do{
System.out.println("\nMENU");
System.out.println("1.View info \n2.Payment\n3.Quit");
System.out.print("Enter your choice: ");

        c = sc.nextInt();
        switch(c)
        {
            case 1 :d.get_info();
                break;

            case 2 :d.get_payment();
                break;

        }
}while(c!=3);
    break;
default :
System.exit(0);
}
System.out.print("press 1 for home page : ");
    e = sc.nextInt();
}while( e ==1);
}
}

```

OUTPUT:

*****FORIGN EXCHANGE*****

SELECT EXCHANGE

- 1.EXPORTER
- 2.BUYER
- 3.EXIT

1

Welcome exporter

Enter Your name : Praveen

Enter ID : 330

Enter Phone no. : 435262

Enter Country : India

MENU

- 1.Offer order
2. Production
- 3.Quit

Enter your choice: 1

*****OFFERS*****

- 1.Electronics
- 2.Food products
- 3.Textile

MENU

- 1.Offer order
2. Production
- 3.Quit

Enter your choice: 2

*****Shipping*****

Shipment Id: 16948196 has been made

MENU

- 1.Offer order
2. Production
- 3.Quit

Enter your choice: 3

press 1 for home page : 1

*****FORIGN EXCHANGE*****

SELECT EXCHANGE

- 1.EXPORTER
- 2.BUYER
- 3.EXIT

2

Welcome Buyer

Enter Your name : Ratheesh

Enter ID : 422

Enter Phone no. : 243652

Enter Country : Pakistan

MENU

1.View info

2.Payment

3.Quit

Enter your choice: 1

*****PROFILE*****

Name : Ratheesh

ID : 422

Phone : 243652

Country : Pakistan

MENU

1.View info

2.Payment

3.Quit

Enter your choice: 2

*****Payment*****

Transaction made successfully

MENU

1.View info

2.Payment

3.Quit

Enter your choice: 3

press 1 for home page : 1

*****FORIGN EXCHANGE*****

SELECT EXCHANGE

1.EXPORTER

2.BUYER

3.EXIT

3

TESTING:

TEST REPORT 1**Product** :FOREIGN EXCHANGE SYSTEM**Use Case** :Exporter

Test Case ID	Test Case / Action To Perform	Expected Result	Actual Result	Pass /Fail
1.	Enter into "Exporter"	Displays the welcome message and prompts for the input from the user	Welcome Exporter	Pass
2.	Provide the correct details	Displays the menu for services	1.Offer order 2. Production 3.Quit Enter your choice: 1	Pass
3.	Enter into offer order	Displays the offers provided	*****OFFERS***** 1.Electronics 2.Food products 3.Textile	Pass

TEST REPORT 2**Product** :Foreign Exchange System**Use Case** :Buyer

Test Case ID	Test Case / Action To Perform	Expected Result	Actual Result	Pass/ Fail
1.	Enter into "Buyer Section"	Displays the welcome message and prompts for input from the buyer	Welcome Buyer	Pass
2.	Provide the correct details	Displays the menu for buyer	1.View info 2.Payment 3.Quit Enter your choice: 1	Pass
3.	Enter into payment section	Initiates the payment	Transaction made successfully	Pass

Ex. No: 5

Date:

BPO MANAGEMENT SYSTEM

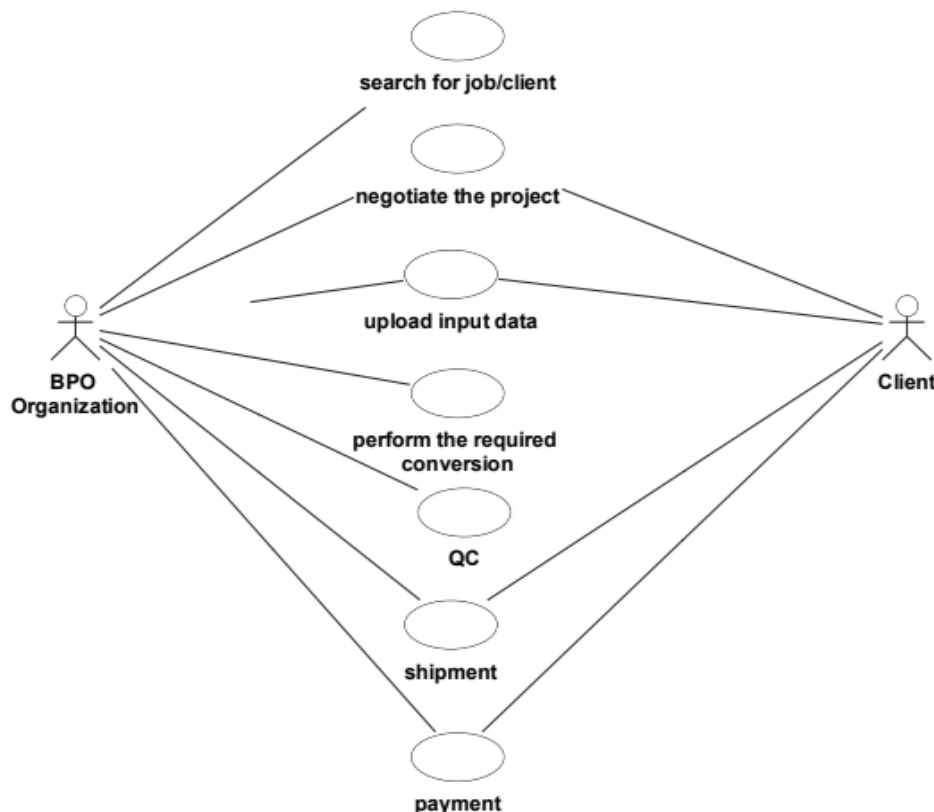
AIM:

To Design ,Implement and Test the BPO Management System described in the given problem statement .

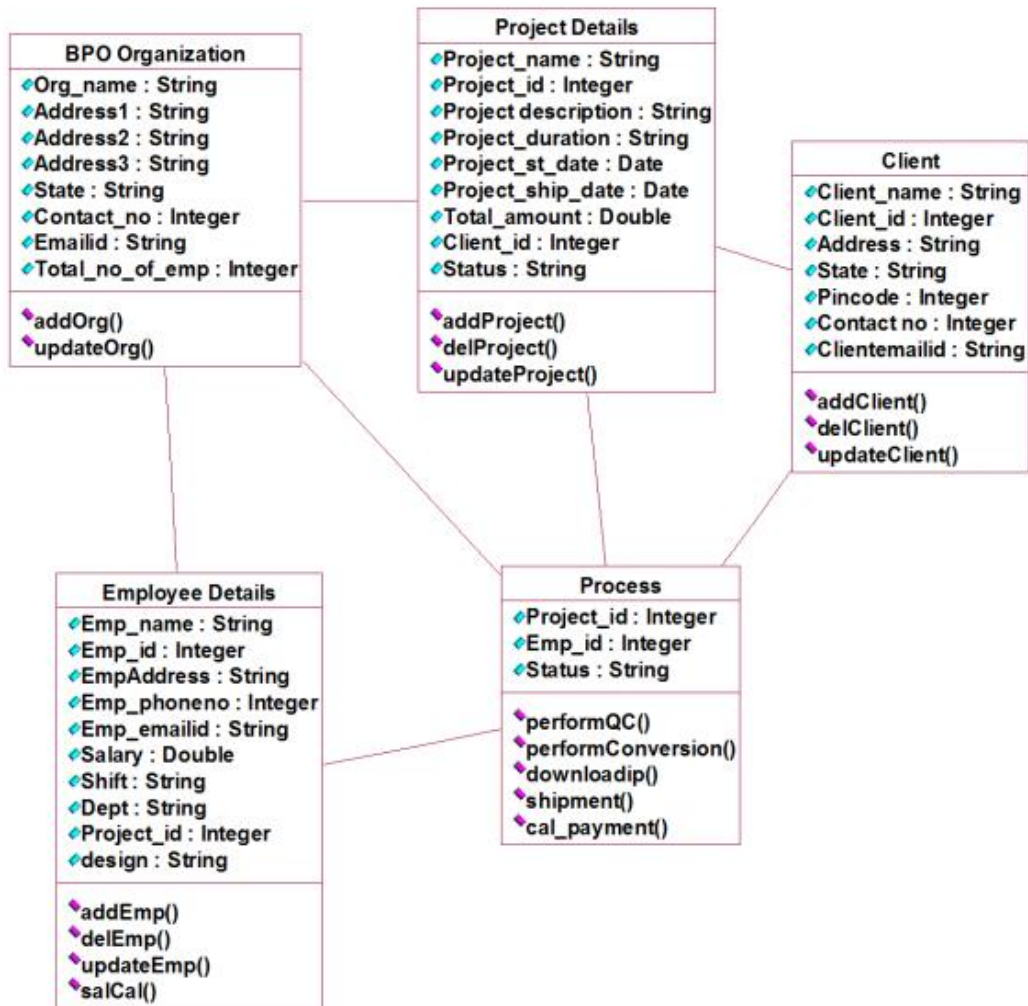
PROBLEM STATEMENT

In this BPO inbound system, the process undergoing is that the agent tries to sell his product so that the agent gets the details of the customer from the database and pitches about his product and makes the sales successful. The communication is done through the telephone. Telephone is the major component used for this customer satisfaction service .This software is designed to know about the processes that were taking place in the BPO office. This system holds the details of the customer who and all approaches to it. It is managed by the central system. With the reduction in communication costs and improved bandwidths and associated infrastructure, BPO as a segment is witnessing a massive growth. One of the key challenges that BPO companies that provide data entry/data validation services is an efficient and effective way of getting the source documents from different customers and accurately route the same to different operators for processing.

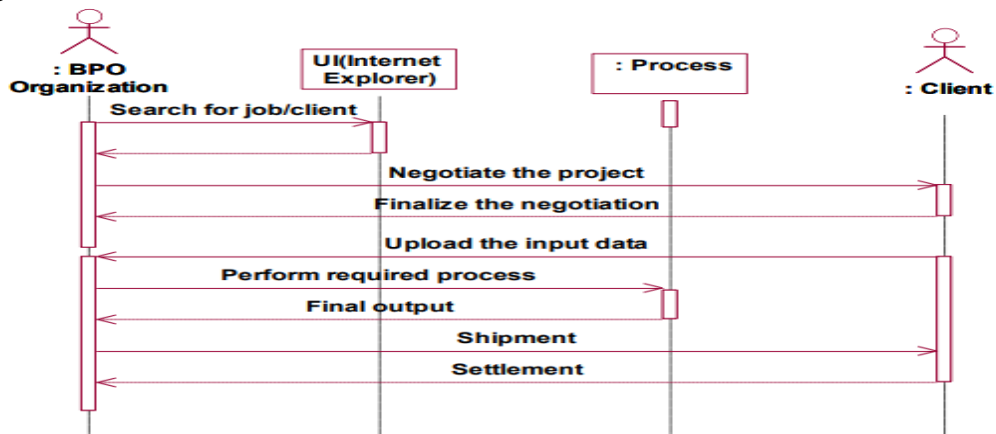
USECASE DIAGRAM:



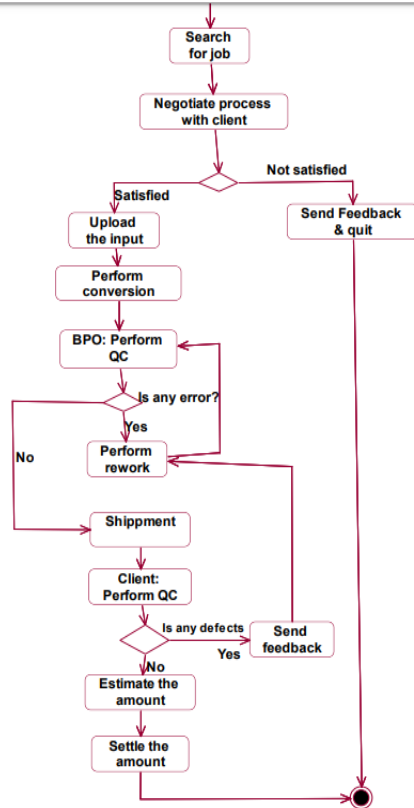
CLASS DIAGRAM:



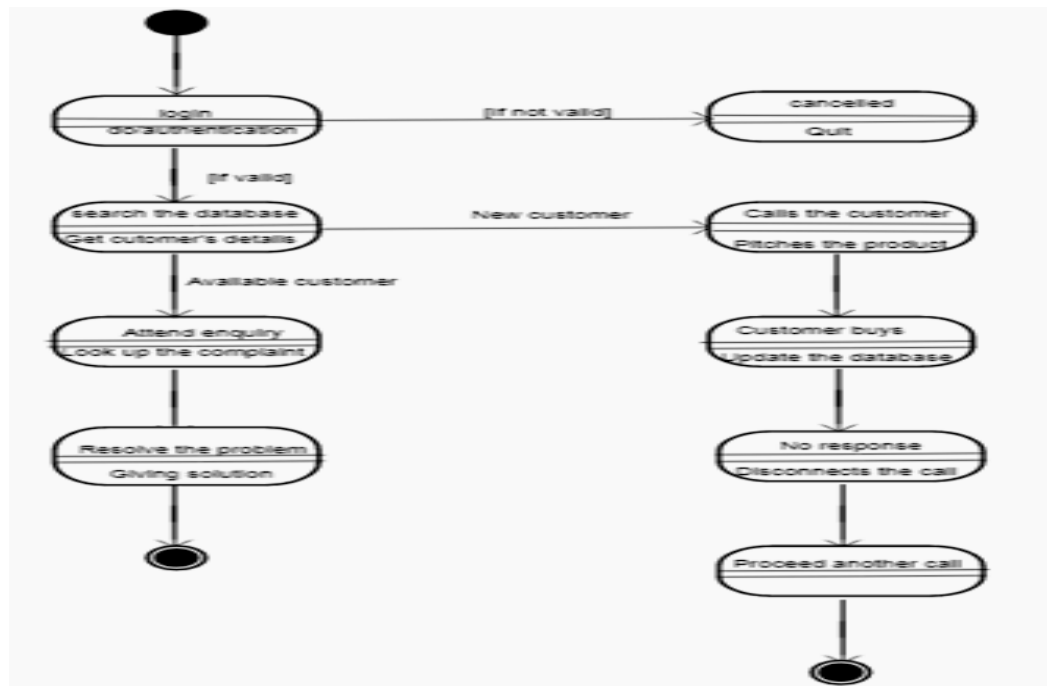
SEQUENCE DIAGRAM:



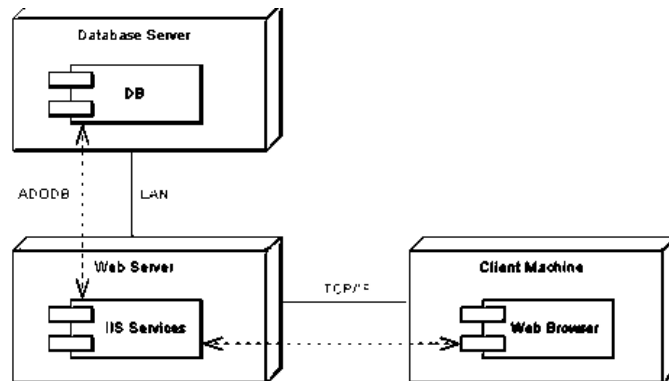
ACTIVITY DIAGRAM:



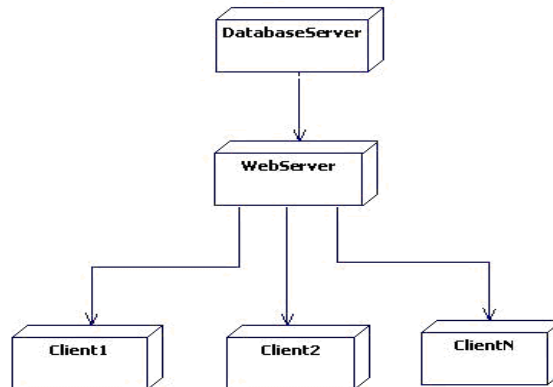
STATE CHART DIAGRAM:



COMPONENT DIAGRAM:



DEPLOYMENT DIAGRAM:



IMPLEMENTATION & TESTING

```
import java.util.Scanner;
import java.util.Timer;
import java.util.TimerTask;
```

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
```

```
/**
 *
 * @author GOMATHI
 */
```

```
//global static int empid=0,projectid=0;
```

```

public class bposystem {

    String bponame,address,email,state,contact_no;
    int noofemp;
    Scanner sc= new Scanner(System.in);
    public void addbpodetails()
    {
        System.out.println("Enter the BPO Name");
        bponame=sc.nextLine();
        System.out.println("Enter the address");
        address=sc.nextLine();
        System.out.println("Enter your statename");
        state=sc.nextLine();
        System.out.println("Enter your Contact No");
        contact_no=sc.nextLine();
        System.out.println("Enter your Number of Employees");
        noofemp=sc.nextInt();
    }
    public void update()
    {
        int choice;
        System.out.println("Which details you want to update");
        System.out.println("1. Name of Organization 2. Address 3. State 4. Contact No ");
        System.out.println("Press the Menu Number to update");
        choice= sc.nextInt();
        switch(choice)
        {
            case 1:
                System.out.println("ENTER YOUR NEW NAME OF ORGANIZATION");
                bponame=sc.nextLine();
                break;
            case 2:
                System.out.println("ENTER YOUR NEW ADDRESS");
                address=sc.nextLine();
                break;
            case 3:
                System.out.println("ENTER THE NEW STATE");
                state=sc.nextLine();
                break;
            case 4:
                System.out.println("ENTER YOUR NEW CONTACT NUMBER");
                contact_no=sc.nextLine();
                break;
            default:
                System.out.println("All changes get updated ");
                break;
        }
    }
}

```

```

}

class employeedetails
{
    Scanner sc= new Scanner(System.in);
    String empname, empadd,empphoneno,dept,shift;
    int salary;
    public static int empid=0,projectid=0;
    public void addemployeedetails()
    {
        empid+=1;
        System.out.println("Enter employeenname");
        empname=sc.nextLine();
        System.out.println("Enter employee address");
        empadd=sc.nextLine();
        System.out.println("Enter your employee Phone Number");
        empphoneno=sc.nextLine();
        System.out.println("Ente the department");
        dept=sc.nextLine();
        System.out.println("Enter the shift");
        shift=sc.nextLine();
        System.out.println("Enter the salary");
        salary=sc.nextInt();
        System.out.println("Enter your project id");
        projectid=projectid+1;
    }
    public void salcal()
    {
        salary=salary+100*projectid;
        System.out.println("The salary of employee is"+salary);
    }
}

class project
{
    Scanner sc= new Scanner(System.in);
    String projectname,projectdes,projectduration,status,startdate,shipdate;
    public static int projectid=0;
    int amount, clientid;
    public void addproject()
    {
        System.out.println("Enter your Project Name");
        projectname=sc.nextLine();
        projectid+=1;
        System.out.println("Enter your Project Description");
        projectdes=sc.nextLine();
        System.out.println("Enter your Project Duration");
        projectduration=sc.nextLine();
        System.out.println("Enter your Project Startdate. Enter in this format DD/MM/YY");
    }
}

```

```

        startdate=sc.nextLine();
        System.out.println("Enter your Project Shipdate. Enter in this format DD/MM/YY");
        shipdate=sc.nextLine();
        System.out.println("Enter the amount");
        amount=sc.nextInt();
        System.out.println("Enter the client id");
        clientid=sc.nextInt();
        //System.out.println("Enter the status");
        //status=sc.nextLine();

    }
}
class client
{
    Scanner sc= new Scanner(System.in);
    String clientname, address, state, pincode,contactno,mailid;
    public static int clientid=0;
    public void addclient()
    {
        System.out.println("Enter the client name");
        clientname=sc.nextLine();
        //System.out.println("Enter the client id");
        clientid+=1;
        System.out.println("Enter the client address");
        address=sc.nextLine();
        System.out.println("Enter the client State");
        state=sc.nextLine();
        System.out.println("Enter the client pincode");
        pincode=sc.nextLine();
        System.out.println("Enter the client contactno");
        contactno=sc.nextLine();
        System.out.println("Enter the client mailid");
        mailid=sc.nextLine();
    }
}
class process {
    Scanner sc=new Scanner(System.in);
    public void queries(String projectna, String projectdes, String projectdur)
    {
        System.out.println("*****Queries Panel*****");
        System.out.println("What is the Project Name?");
        System.out.println("Project Name is: "+projectna);
        System.out.println("What is the Project Name?");
        System.out.println("Project Description is: "+projectdes);
        System.out.println("What is the Project Duration?");
        System.out.println("Project Duration is: "+projectdur);
    }
    public void conversation()
    {
        System.out.println("Speak with the client...");
    }
}

```



```

Timer timer=new Timer();
timer.schedule(new TimerTask(){
@Override
public void run()
{
System.out.println("Conversation going on...");
}
},4500);
}
public void shipment(String ship)
{
String reason;
int ch,ch1;
System.out.println("Shipment date"+ship);
System.out.println("IF PROJECT HAS BEEN COMPLETED PRESS 1 OR ELSE PRESS 0");
ch=sc.nextInt();
if(ch==1)
{
System.out.println("Project has been successfully completed");
System.out.println("Press 1 to export the project to client");
ch1=sc.nextInt();
if(ch1==1)
{
System.out.println("Project has been successfully exported.");
System.out.println("Payment will be done within 12 hours to your account");
}
else
{
}
}
else
{
System.out.println("Project has been delayed");
System.out.println("Enter the reason for delay");
reason=sc.nextLine();
System.out.println("Penalty will be imposed for each day of delay. 1 day Rs.1000 will be deduced
from the amount ");
}
}
public void calpayment(int amt)
{
int daysdelay,payment,bonus=100;
System.out.println("Enter the total delay in time");
daysdelay=sc.nextInt();
System.out.println("amount is "+amt);
if(daysdelay==0)
{
payment=amt*bonus;
}
else
{

```

```

        payment=amt-daysdelay*1000;
    }
    System.out.println("Final amount payed"+payment);
}
}
class start extends project
{
public static void main(String args[])
{

bposystem bpo=new bposystem();
employeeedetails emp=new employeeedetails();
project prj =new project();
client clt=new client();
process pr=new process();
bpo.addbpodetails();
emp.addemployeeedetails();
emp.salcal();

prj.addproject();
int am= prj.amount;
String ship=prj.shipdate;
String prjn=prj.projectname;
String prjnd=prj.projectdes;
String prjdur=prj.projectduration;
clt.addclient();
pr.queries(prjn, prjnd, prjdur);
pr.conversation();
pr.shipment(ship);
pr.calpayment(am);

}
}

```