**1. UML INTRODUCTION**

UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems.

* UML was created by Object Management Group (OMG) and UML
* Specification draft was proposed to the OMG in January 1997.
* OMG is continuously putting effort to make a truly industry standard.
* UML stands for Unified Modeling Language.
* UML is different from the other common programming languages like C++, Java, and COBOL etc.
* UML is a pictorial language used to make software blue prints.

**GOALS OF UML**

* A picture is worth a thousand words, this absolutely fits while discussing about UML.
* UML diagrams are not only made for developers but also for business users, common people and anybody interested to understand the system.
* The system can be a software or non software. So it must be clear that.
* UML is not a development method rather it accompanies with processes to make a successful system.

**CONCEPTUAL MODEL OF UML**

* A conceptual model can be defined as a model which is made of Concepts and their relationships.
* A conceptual model is the first step before drawing a UML diagram. It helps to understand the entities in the real world and how they interact with each other. Conceptual model of UML can be mastered by learning.

The following three major elements:

* UML building blocks.
* Rules to connect the building blocks.
* Common mechanisms of UML.

**OO Analysis and Design**

The purpose of OO analysis and design can described as:

1. Identifying the objects of a system.

2. Identify their relationships.

3. Make a design which can be converted to executables using OO languages.

OO Analysis --> OO Design --> OO implementation using OO languages.

* During object oriented analysis the most important purpose is to identify objects and describing them in a proper way. If these objects are identified efficiently then the next job of design is easy. The objects should be identified with responsibilities. Responsibilities are the functions performed by the object. Each and every object has some type of responsibilities to be performed. When these responsibilities are collaborated the purpose of the system is fulfilled.
* The second phase is object oriented design. During this phase emphasis is given upon the requirements and their fulfillment. In this stage the objects are collaborated according to their intended association. After the association is complete the design is also complete.
* The third phase is object oriented implementation. In this phasethe design is implemented using object oriented languages like Java, C++ etc.

**BUILDING BLOCKS**

The building blocks of UML can be defined as:

1. Things
2. Relationships
3. Diagrams

**(1) Things**

**Things** are the most important building blocks of UML. Things can be:

* Structural
* Behavioral
* Grouping
* Annotational

**Structural things**

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

**Class**

Class represents set of objects having similar responsibilities.



**Interface**

Interface defines a set of operations which specify the responsibility of a class.



**Collaboration**

Collaboration defines interaction between elements



**Use case**

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



**Component**

Component describes physical part of a system.



**Node**

A node can be defined as a physical element that exists at run time.



**Behavioral things**

**A behavioral thing** consists of the dynamic parts of UML models. Following are the behavioral things:

**Interaction**

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

****

**State machine**

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



**Grouping things**

**Grouping things** can be defined as a mechanism to group elements of a UML model together. There is only one grouping thing available

**Package**

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



**Annotational things**

**Annotational things** can be defined as a mechanism to capture remarks, descriptions, and comments of UML model elements. **Note** is the only one Annotational thing available

**Note:**

A note is used to render comments, constraints etc of an UML element.



**(2) Relationships**

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

There are four kinds of relationships available.

* Dependency
* Association
* Generalization.
* Realization

**Dependency**

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



**Association**

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



**Generalization**

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



**Realization**

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



# (3) Diagrams

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

[**Use Case Diagram**](http://pigseye.kennesaw.edu/~dbraun/csis4650/A&D/UML_tutorial/use_case.htm) displays the relationship among actors and use cases

[**Class Diagram**](http://pigseye.kennesaw.edu/~dbraun/csis4650/A&D/UML_tutorial/class.htm) models class structure and contents using design elements such as classes, packages and objects. It also displays relationships such as containment, inheritance, associations and others.

[**Interaction Diagrams**](http://pigseye.kennesaw.edu/~dbraun/csis4650/A&D/UML_tutorial/interaction.htm)

* **Sequence Diagram**displays the time sequence of the objects participating in the interaction. This consists of the vertical dimension (time) and horizontal dimension (different objects).
* **Collaboration Diagram**displays an interaction organized around the objects and their links to one another. Numbers are used to show the sequence of messages.

[**State Diagram**](http://pigseye.kennesaw.edu/~dbraun/csis4650/A&D/UML_tutorial/state.htm) displays the sequences of states that an object of an interaction goes through during its life in response to received stimuli, together with its responses and actions.

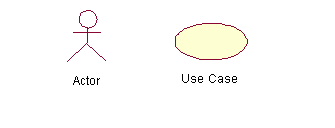
[**Activity Diagram**](http://pigseye.kennesaw.edu/~dbraun/csis4650/A&D/UML_tutorial/activity.htm) displays a special state diagram where most of the states are action states and most of the transitions are triggered by completion of the actions in the source states. This diagram focuses on flows driven by internal processing.

[**Physical Diagrams**](http://pigseye.kennesaw.edu/~dbraun/csis4650/A&D/UML_tutorial/physical.htm)

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

# Use Case Diagrams

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



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

### When to Use: Use Cases Diagrams

Use cases are used in almost every project.  These are helpful in exposing requirements and planning the project. During the initial stage of a project most use cases should be defined, but as the project continues more might become visible.

**Modeling steps for Use case Diagram**

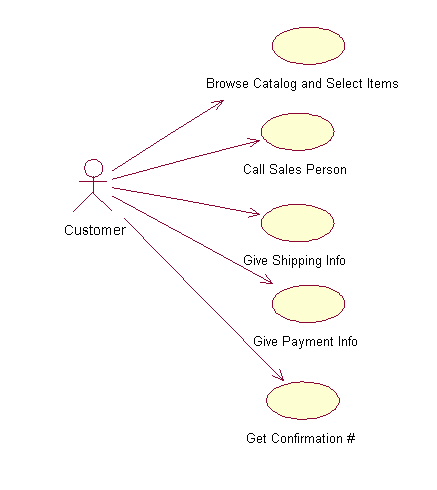
1. Draw the lines around the system and actors lie outside the system.
2. Identify the actors which are interacting with the system.
3. Separate the generalized and specialized actors.
4. Identify the functionality the way of interacting actors with system and specify the behavior of actor.
5. Functionality or behavior of actors is considered as use cases.
6. Specify the generalized and specialized use cases.
7. Se the relationship among the use cases and in between actor and use cases.
8. Adorn with constraints and notes.
9. If necessary, use collaborations to realize use cases.

### How to Draw: Use Cases Diagrams

Use cases are a relatively easy UML diagram to draw, but this is a very simplified example.  This example is only meant as an introduction to the UML and use cases.  Start by listing a sequence of steps a user might take in order to complete an action.  For example a user placing an order with a sales company might follow these steps.

1. Browse catalog and select items.
2. Call sales representative.
3. Supply shipping information.
4. Supply payment information.
5. Receive conformation number from salesperson.

These steps would generate this simple use case diagram:



This example shows the customer as a actor because the customer is using the ordering system.  The diagram takes the simple steps listed above and shows them as actions the customer might perform.  The salesperson could also be included in this use case diagram because the salesperson is also interacting with the ordering system.

From this simple diagram the requirements of the ordering system can easily be derived.  The system will need to be able to perform actions for all of the use cases listed.  As the project progresses other use cases might appear.  The customer might have a need to add an item to an order that has already been placed.  This diagram can easily be expanded until a complete description of the ordering system is derived capturing all of the requirements that the system will need to perform.

**The *<<extends>>*Relationship**

* <<Extends>> relationships represent exceptional or seldom invoked cases.
* The exceptional event flows are factored out of the main event flow for clarity.
* Use cases representing exceptional flows can extend more than one use case.
* The direction of a <<extends>> relationship is to the extended use case

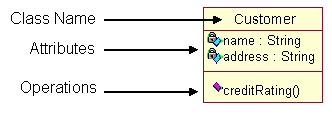
**The *<<includes>>*Relationship**

* <<Includes>> relationship represents behavior that is factored out of the use case.
* <<Includes>> behavior is factored out for reuse, not because it is an exception.
* The direction of a <<includes>> relationship is to the using use case (unlike <<extends>> relationships).

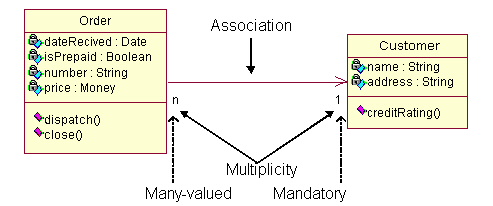
# Class Diagrams

Class diagrams are widely used to describe the types of objects in a system and their relationships.  Class diagrams model class structure and contents using design elements such as classes, packages and objects.  Class diagrams describe three different perspectives when designing a system, conceptual, specification, and implementation.   These perspectives become evident as the diagram is created and help solidify the design.  This example is only meant as an introduction to the UML and class diagrams.  Classes are composed of three things: a name, attributes, and operations.

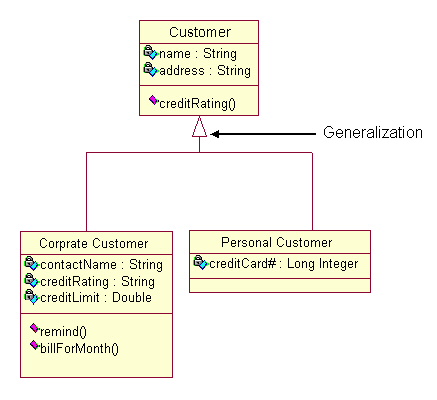
Below is an example of a class.



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



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

A generalization is used when two classes are similar, but have some differences.  

In this example the classes Corporate Customer and Personal Customer have some similarities such as name and address, but each class has some of its own attributes and operations.  The class Customer is a general form of both the Corporate Customer and Personal Customer classes.  This allows the designers to just use the Customer class for modules and do not require in-depth representation of each type of customer.

### When to Use: Class Diagrams

Class diagrams are used in nearly all Object Oriented software designs. Use them to describe the Classes of the system and their relationships to each other.

**Modeling steps for Class Diagrams**

1. Identity the things that are interacting with class diagram.
2. Set the attributes and operations.
3. Set the responsibilities.
4. Identify the generalization and specification classes.
5. Set the relationship among all the things.
6. Adorn with tagged values, constraints and notes.

### How to Draw: Class Diagrams

Class diagrams are some of the most difficult UML diagrams to draw.  To draw detailed and useful diagrams a person would have to study UML and Object Oriented principles for a long time.  Therefore, this page will give a very high level overview of the process.

Before drawing a class diagram consider the three different perspectives of the system the diagram will present; conceptual, specification, and implementation.  Try not to focus on one perspective and try see how they all work together.

When designing classes consider what attributes and operations it will have.  Then try to determine how instances of the classes will interact with each other. These are the very first steps of many in developing a class diagram.  However, using just these basic techniques one can develop a complete view of the software system.



**Interaction Diagrams**

Interaction diagrams model the behavior of use cases by describing the way groups of objects interact to complete the task.  The two kinds of interaction diagrams are **sequence** and **collaboration** diagrams. This example is only meant as an introduction to the UML and interaction diagrams.

**When to Use: Interaction Diagrams**

Interaction diagrams are used when you want to model the behavior of several objects in a use case.  They demonstrate how the objects collaborate for the behavior.  Interaction diagrams do not give a in depth representation of the behavior.  If you want to see what a specific object is doing for several use cases use a [state diagram](http://pigseye.kennesaw.edu/~dbraun/csis4650/A&D/UML_tutorial/state.htm).  To see a particular behavior over many use cases or threads use an [activity diagrams](http://pigseye.kennesaw.edu/~dbraun/csis4650/A&D/UML_tutorial/activity.htm).

**How to Draw: Interaction Diagrams**

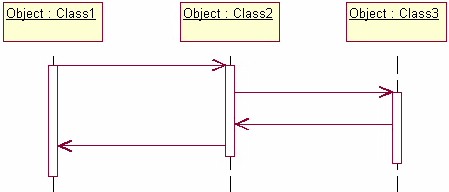
Sequence diagrams, collaboration diagrams, or both diagrams can be used to demonstrate the interaction of objects in a use case.  Sequence diagrams generally show the sequence of events that occur.  Collaboration diagrams demonstrate how objects are statically connected.  Both diagrams are relatively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

**Sequence diagrams:**

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

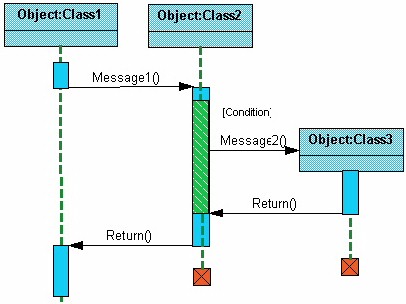
**Modeling steps for Sequence Diagrams**

1. Set the context for the interactions, system, subsystem, classes, object or use cases.
2. Set the stages for the interactions by identifying objects which are placed as actions in interaction diagrams.
3. Lay them out along the X-axis by placing the important object at the left side and others in the next subsequent.
4. Set the lifelines for each and every object by sending create and destroy messages.
5. Start the message which is initiating interactions and place all other messages in the increasing order of items.
6. Specify the time and space constraints.
7. Set the pre and post conditions.



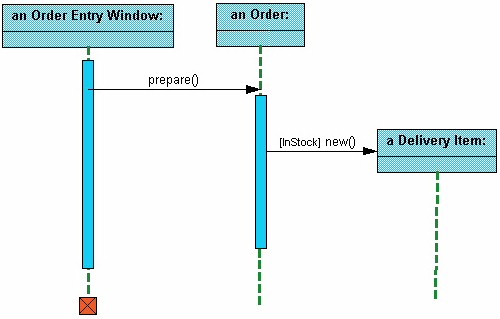
Below is a slightly more complex example.  The light blue vertical rectangles the objects activation while the green vertical dashed lines represent the life of the object.  The green vertical rectangles represent when a particular object has control.  The represents when the object is destroyed.  This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

object of class 2 can send a message() to the object of class 3.

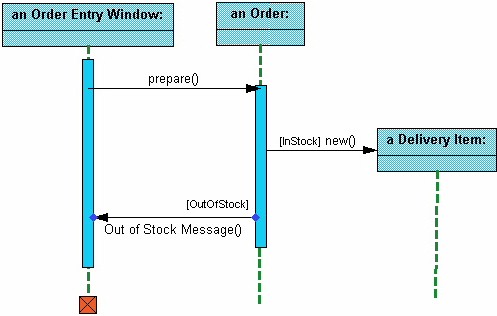


The next diagram shows the beginning of a sequence diagram for placing an order.  The object an Order Entry Window is created and sends a message to an Order object to prepare the order. Notice the the names of the objects are followed by a colon.  The names of the classes the objects belong to do not have to be listed.  However the colon is required to denote that it is the name of an object following the objectName:className naming system.

Next the Order object checks to see if the item is in stock and if the [InStock] condition is met it sends a message to create an new Delivery Item object.



The next diagrams adds another conditional message to the Order object.  If the item is [OutOfStock] it sends a message back to the Order Entry Window object stating that the object is out of stack.



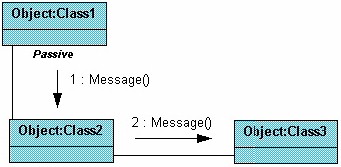
This simple diagram shows the sequence that messages are passed between objects to complete a use case for ordering an item.

**Collaboration diagrams**

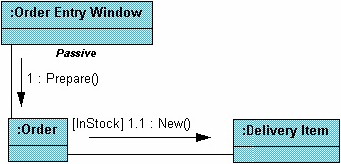
Collaboration diagrams are also relatively easy to draw.  They show the relationship between objects and the order of messages passed between them.  The objects are listed as icons and arrows indicate the messages being passed between them. The numbers next to the messages are called sequence numbers.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

**Modeling steps for Collaboration Diagrams**

1. Set the context for interaction, whether it is ystem, subsystem, operation or class or one scenario of use case or collaboration.
2. Identify the objects that play a role in the interaction. Lay them as vertices in graph, placing important objects in centre and neigboring objects to outside.
3. Set the initial properties of each of these objects. If the attributes or tagged values of an object changes in significant ways over the interaction, place a duplicate object, update with these new values and connect them by a message stereotyped as become or copy.
4. Specify the links among these objects. Lay the association links first represent structural connection lay out other links and adorn with stereotypes.
5. Starting with the message that initiates this interaction, attach each subsequent message to appropriate link, setting sequence number as appropriate.



The example below shows a simple collaboration diagram for the placing an order use case.  This time the names of the objects appear after the colon, such as :Order Entry Window following the objectName: className naming convention. This time the class name is shown to demonstrate that all of objects of that class will behave the same way.



# State chart diagram

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

**Modeling steps for Statechart Diagram**

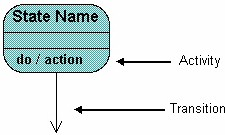
1. Choose the context for state machine, whether it is a class ,a use case, or the system as a whole.
2. Choose the initial & final states of the objects.
3. Decide on the stable states of the object by considering the conditions in which the object may exist for some identifiable period of time. Start with the high level states of the objects & only then consider its possible substates.
4. Decide on the meaningful partial ordering of stable states over the lifetime of the object.
5. Decide on the events that may trigger a transition from state to state. Model these events as triggers to transitions that move from one legal ordering of states to another.
6. Attach actions to these transitions and/or to these states.
7. Consider ways to simplify your machine by using substates, branches, forks, joins and history states.
8. Check that all states are reachable under some combination of events.
9. Check that no state is a dead from which no combination of events will transition the object out of that state.
10. Trace through the state machine, either manually or by using tools, to check it against expected sequence of events & their responses.

### When to Use: State Diagrams

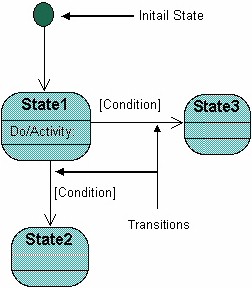
Use state diagrams to demonstrate the behavior of an object through many use cases of the system.  Only use state diagrams for classes where it is necessary to understand the behavior of the object through the entire system.  Not all classes will require a state diagram and state diagrams are not useful for describing the collaboration of all objects in a use case.  State diagrams are other combined with other diagrams such as [interaction diagrams](http://pigseye.kennesaw.edu/~dbraun/csis4650/A&D/UML_tutorial/interaction.htm) and [activity diagrams](http://pigseye.kennesaw.edu/~dbraun/csis4650/A&D/UML_tutorial/activity.htm).

**How to Draw: State Diagrams**

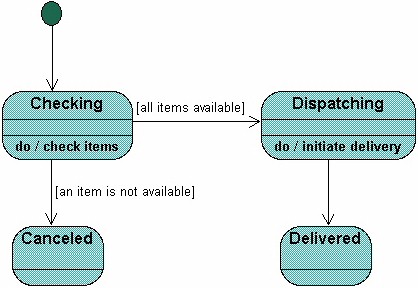
State diagrams have very few elements.  The basic elements are rounded boxes representing the state of the object and arrows indicting the transition to the next state.  The activity section of the state symbol depicts what activities the object will be doing while it is in that state.



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

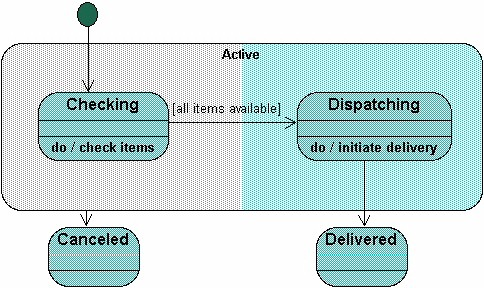


Below is an example of a state diagram might look like for an Order object.  When the object enters the Checking state it performs the activity "check items."  After the activity is completed the object transitions to the next state based on the conditions [all items available] or [an item is not available].  If an item is not available the order is canceled.  If all items are available then the order is dispatched.  When the object transitions to the Dispatching state the activity "initiate delivery" is performed.  After this activity is complete the object transitions again to the Delivered state.



State diagrams can also show a super-state for the object. A super-state is used when many transitions lead to the certain state.  Instead of showing all of the transitions from each state to the redundant state a super-state can be used to show that all of the states inside of the super-state can transition to the redundant state.  This helps make the state diagram easier to read.

The diagram below shows a super-state.  Both the Checking and Dispatching states can transition into the Canceled state, so a transition is shown  from a super-state named Active to the state Cancel.  By contrast, the state Dispatching can only transition to the Delivered state, so we show an arrow only from the Dispatching state to the Delivered state.



# Activity Diagram

Activity diagrams describe the workflow behavior of a system.  Activity diagrams are similar to [state diagrams](http://pigseye.kennesaw.edu/~dbraun/csis4650/A&D/UML_tutorial/state.htm) because activities are the state of doing something.  The diagrams describe the state of activities by showing the sequence of activities performed.  Activity diagrams can show activities that are conditional or parallel.

**Modeling steps for Activity Diagrams**

1. Select the object that have high level responsibilities.
2. These objects may be real or abstract. In either case, create a swimlane for each important object.
3. Identify the precondition of initial state and post conditions of final state.
4. Beginning at initial state, specify the activities and actions and render them as activity states or action states.
5. For complicated actions, or for a set of actions that appear multiple times, collapse these states and provide separate activity diagram.
6. Render the transitions that connect these activities and action states.
7. Start with sequential flows, consider branching, fork and joining.
8. Adorn with notes tagged values and so on.

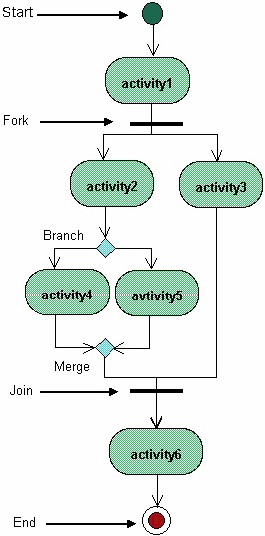
### When to Use: Activity Diagrams

Activity diagrams should be used in conjunction with other modeling techniques such as [interaction diagrams](http://pigseye.kennesaw.edu/~dbraun/csis4650/A&D/UML_tutorial/interaction.htm) and [state diagrams](http://pigseye.kennesaw.edu/~dbraun/csis4650/A&D/UML_tutorial/state.htm).  The main reason to use activity diagrams is to model the workflow behind the system being designed.  Activity Diagrams are also useful for analyzing a use case by describing what actions need to take place and when they should occur;  describing a complicated sequential algorithm;  and modeling applications with parallel processes.

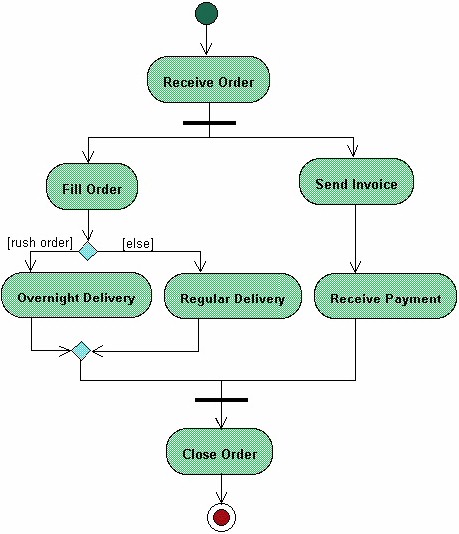
However, activity diagrams should not take the place of  [interaction diagrams](http://pigseye.kennesaw.edu/~dbraun/csis4650/A&D/UML_tutorial/interaction.htm) and [state diagrams](http://pigseye.kennesaw.edu/~dbraun/csis4650/A&D/UML_tutorial/state.htm).  Activity diagrams do not give detail about how objects behave or how objects collaborate.

### How to Draw: Activity Diagrams

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



Below is a possible activity diagram for processing an order.  The diagram shows the flow of actions in the system's workflow.  Once the order is received the activities split into two parallel sets of activities.  One side fills and sends the order while the other handles the billing.  On the Fill Order side, the method of delivery is decided conditionally.  Depending on the condition either the Overnight Delivery activity or the Regular Delivery activity is performed.  Finally the parallel activities combine to close the order.



# Physical Diagrams

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

**Modeling steps for Component Diagrams**

1. Identify the component libraries and executable files which are interacting with the system.
2. Represent this executables and libraries as components.
3. Show the relationships among all the components.
4. Identify the files, tables, documents which are interacting with the system.
5. Represent files, tables, documents as components.
6. Show the existing relationships among them generally dependency.
7. Identify the seams in the model.
8. Identify the interfaces which are interacting with the system.
9. Set attributes and operation signatures for interfaces.
10. Use either import or export relationship in b/w interfaces & components.
11. Identify the source code which is interacting with the system.
12. Set the version of the source code as a constraint to each source code.
13. Represent source code as components.
14. Show the relationships among components.

**Modeling steps for Deployment Diagram**

1. Identify the processors which represent client & server.
2. Provide the visual cue via stereotype classes.
3. Group all the similar clients into one package.
4. Provide the links among clients & servers.
5. Provide the attributes & operations.
6. Specify the components which are living on nodes.
7. Adorn with nodes & constraints & draw the deployment diagram.

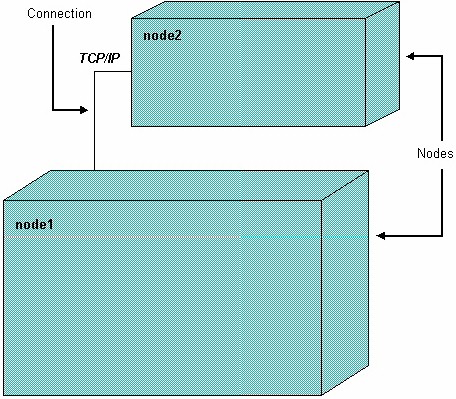
### When to Use: Physical Diagrams

Physical diagrams are used when development of the system is complete.  Physical diagrams are used to give descriptions of the physical information about a system.

### How to Draw: Physical Diagrams

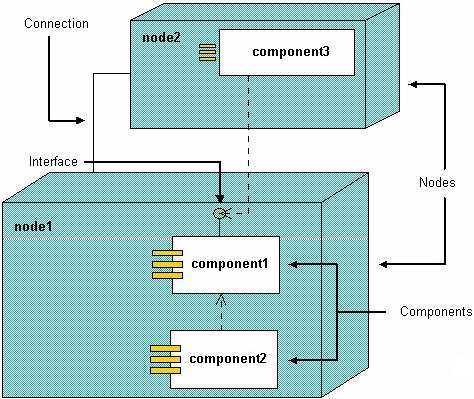
Many times the deployment and component diagrams are combined into one physical diagram.  A combined deployment and component diagram combines the features of both diagrams into one diagram.

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



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

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



**2.SOFTWARE REQUIREMENT SPECIFICATION**

**A** **Software requirements specification** (SRS), a requirements specification for a [software system](http://en.wikipedia.org/wiki/Software_system), is a complete description of the behavior of a system to be developed and may include a set of [use cases](http://en.wikipedia.org/wiki/Use_case) that describe interactions the users will have with the software

**Software Requirements** is a field within [Software Engineering](http://en.wikipedia.org/wiki/Software_Engineering) that deals with establishing the needs of stakeholders that are to be solved by software. The IEEE Standard Glossary of Software Engineering Technology defines a software requirement as:

1. A condition or capability needed by a user to solve a problem or achieve an objective.
2. A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document.
3. A documented representation of a condition or capability as in 1 or 2.

**IEEE Standard SRS Template**

**1. Introduction**

1.1. Purpose

1.2. Scope

1.3. Definitions, acronyms & abbreviations

1.4. References

1.5. Overview

**2. Overall description**

2.1.Product perspective

2.1.1. System interfaces

2.1.2. User interfaces

2.1.3. Hardware interfaces

2.1.4. Software interfaces

2.1.5. Communications interfaces

2.1.6. Memory constraints

2.1.7. Operations

2.1.8. Site adaptation requirements

2.2. Product functions

2.3. User characteristics

2.4. Constraints

2.5. Assumptions and dependencies

2.6. Apportioning of requirements

**3. Specific Requirements**

3.1 External interface requirements

3.1.1 User interfaces

3.1.2 Hardware interfaces

3.1.3 Software interfaces

3.1.4 Communication interfaces

3.2 Specific requirements

3.2.1 Sequence diagrams

3.2.2 Classes for classification of specific requirements

3.3 Performance requirements

3.4 Design constraints

3.5 Software system attributes

3.5.1 Reliability

3.5.2 Availability

3.5.3 Security

3.5.4 Maintainability

3.6 Other requirements

**4. Supporting information**

4.1 Table of contents and index

4.2 Appendices

**1.  Introduction**

The following subsections of the SRS should provide an overview of the entire SRS.

**1.1 Purpose**

Identify the purpose of this SRS and its intended audience.

**1.2 Scope.**

(1)  Identify the software product(s) to be produced by name

(2)  Explain what the software product(s) will, and, if necessary, will not do

(3)  Describe the application of the software being specified.  As a portion of this, it should:

        (a)  Describe the relevant benefits, objectives, and goals as precisely as possible

        (b)  Be consistent with similar statements in higher-level specifications if they exist.

**1.3 Definitions, Acronyms, and Abbreviations**

Provide the definitions of all terms, acronyms, and abbreviations required to properly interpret the SRS.  This information may be provided by reference to an appendix or other document(s).

**1.4 References**

In this subsection:

(1)  Provide a complete list of all documents referenced elsewhere in the SRS.

(2)  Identify each document by title, report number  (if applicable), date, and publishing organization.

(3)  Specify the sources from which the references can be obtained.

**1.5 Overview**

Describe the rest of the SRS and how it is organized.

**2.  THE GENERAL DESCRIPTION**

Describe the general factors that affect the product and its requirements.  This section usually consists of the five subsections that follow.  This section does not state specific requirements; each of its subsections makes those requirements easier to understand; they do not specify design or express specific requirements.

**2.1 Product Perspective**

This subsection of the SRS relates the product to other products or projects.

(1)  If the product is independent and totally self-contained, it should be stated here.

(2)  If the SRS defines a product that is a component of a larger system or project:

        (a)  Describe the functions of each component of the larger system or project, and identify interfaces

        (b)  Identify the principal external interfaces of this software product (not a detailed description)

        (c)  Describe the computer hardware and peripheral equipment to be used (overview only)

A block diagram showing the major components of the larger system or project, interconnections, and external interfaces can be very helpful.

**2.2 Product Functions**

Provide a summary of the functions that the software will perform. Sometimes the function summary that is necessary for this part can be taken directly from the section of the higher-level specification (if one exists) that allocates particular functions to the software product.  The functions should be organized in a way that makes the list of functions understandable to the customer or to anyone else reading the document for the first time.  Block diagrams showing the different functions and their relationships can be helpful.  Such a diagram is not a requirement on the design of a product itself; it is simply an effective explanatory tool.

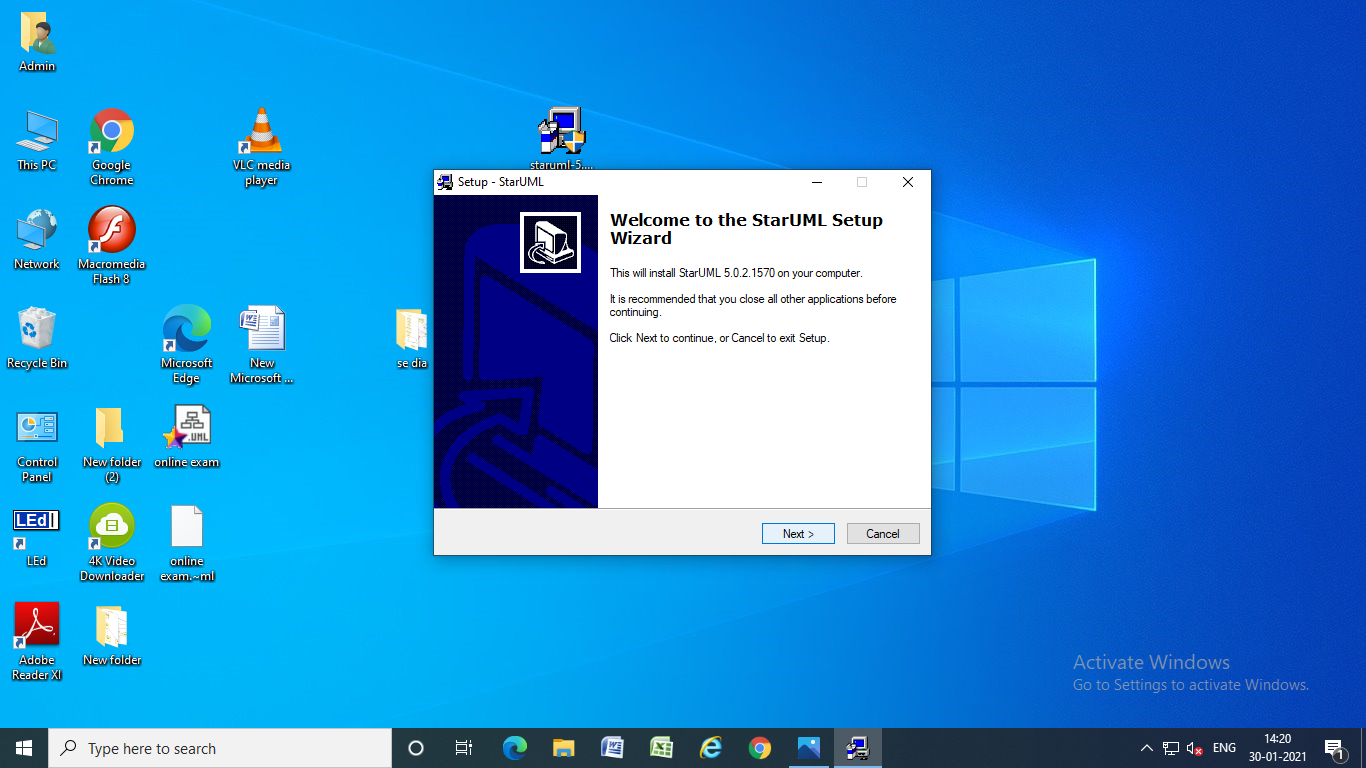
**2.3 User Characteristics**

Describe those general characteristics of the eventual users of the product that will affect the specific requirements.

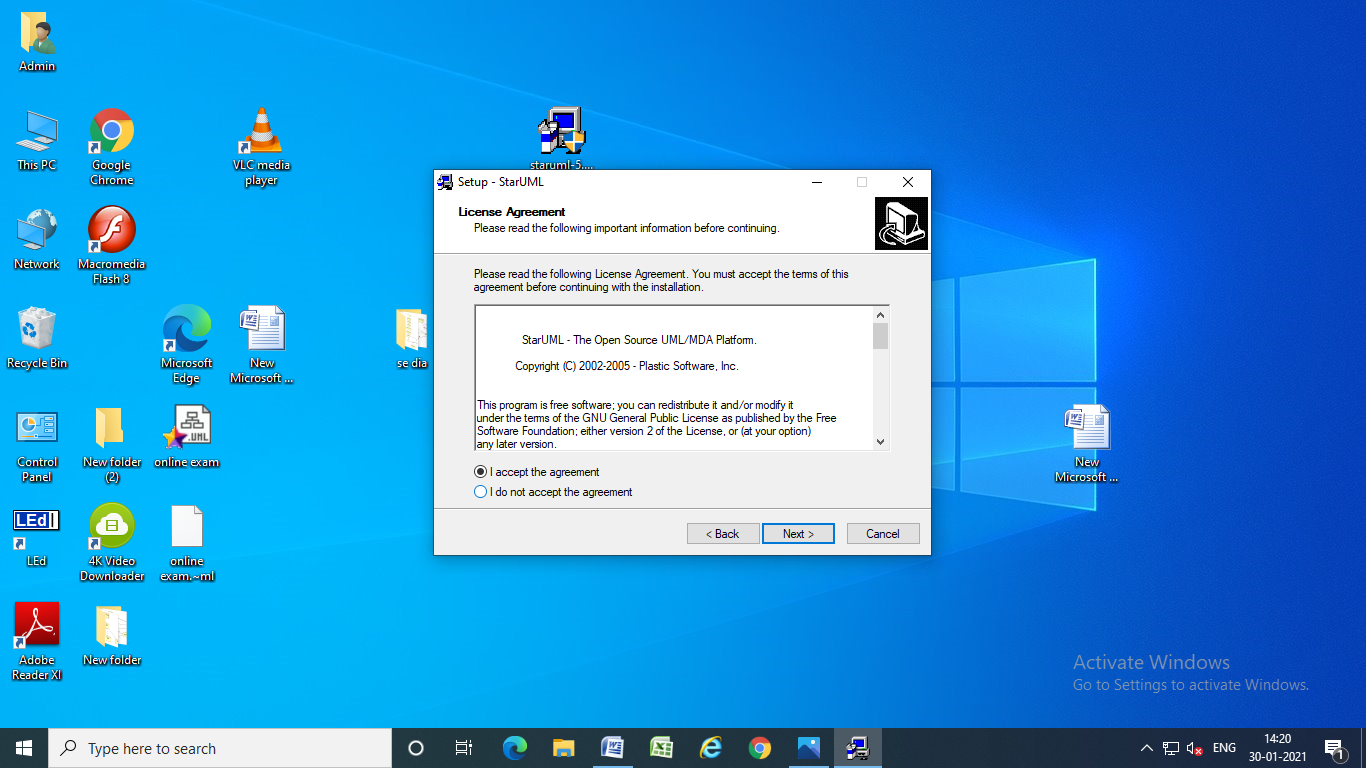
Many people interact with a system during the operation and maintenance phase of the software life cycle.  Some of these people are users, operators, and maintenance and systems personnel.  Certain characteristics of these people, such as educational level, experience, and technical expertise impose important constraints on the system's operating environment.

**3.Instalation process of starUml 5.0 software:**

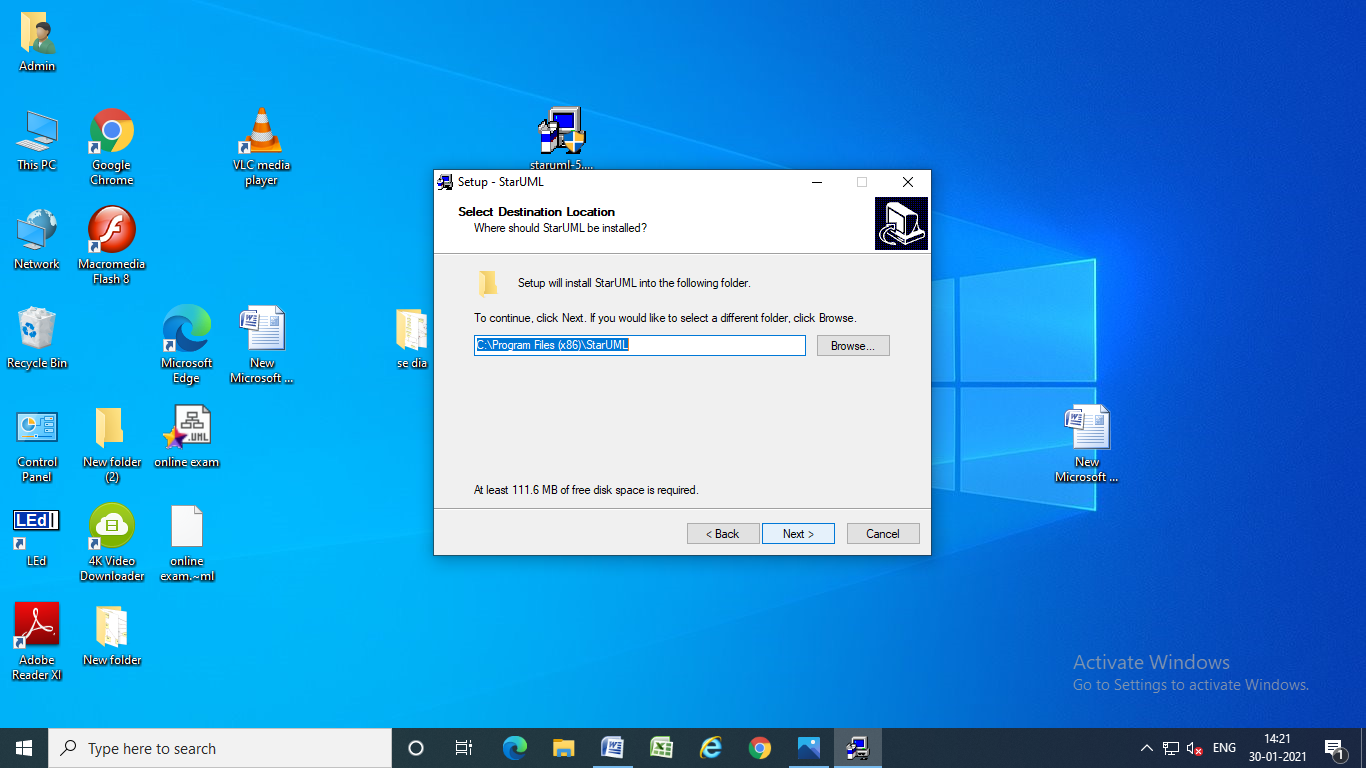
**Step:1** Run the software ,open staruml setup click on next

****

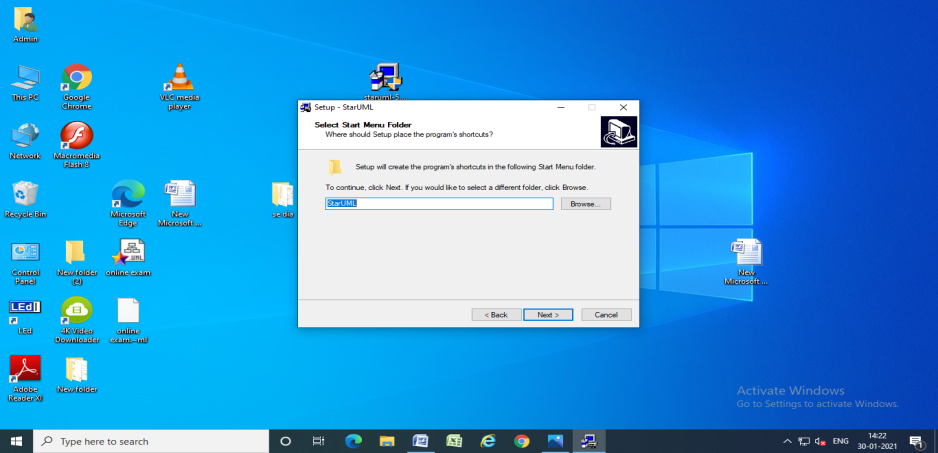
**Step 2:** open licensed Agreement, click on accept the agreement and click on next button

****

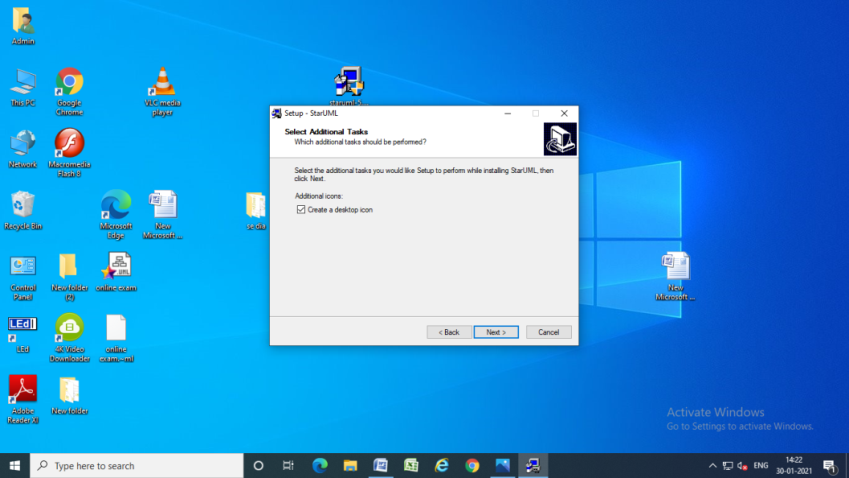
**Step 3:** software saved in programme files click on next button

****

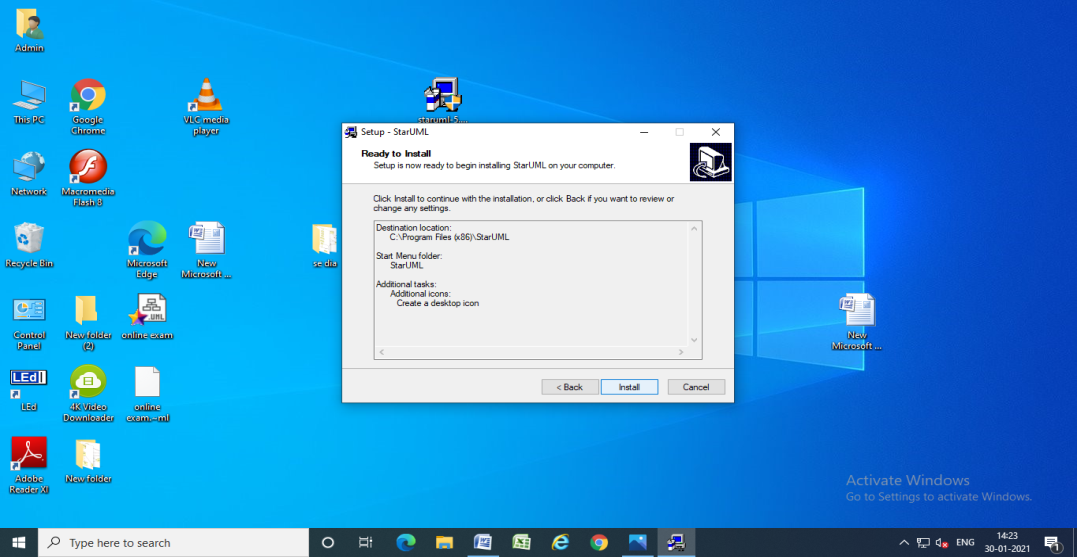
**Step 4:** click on next button

****

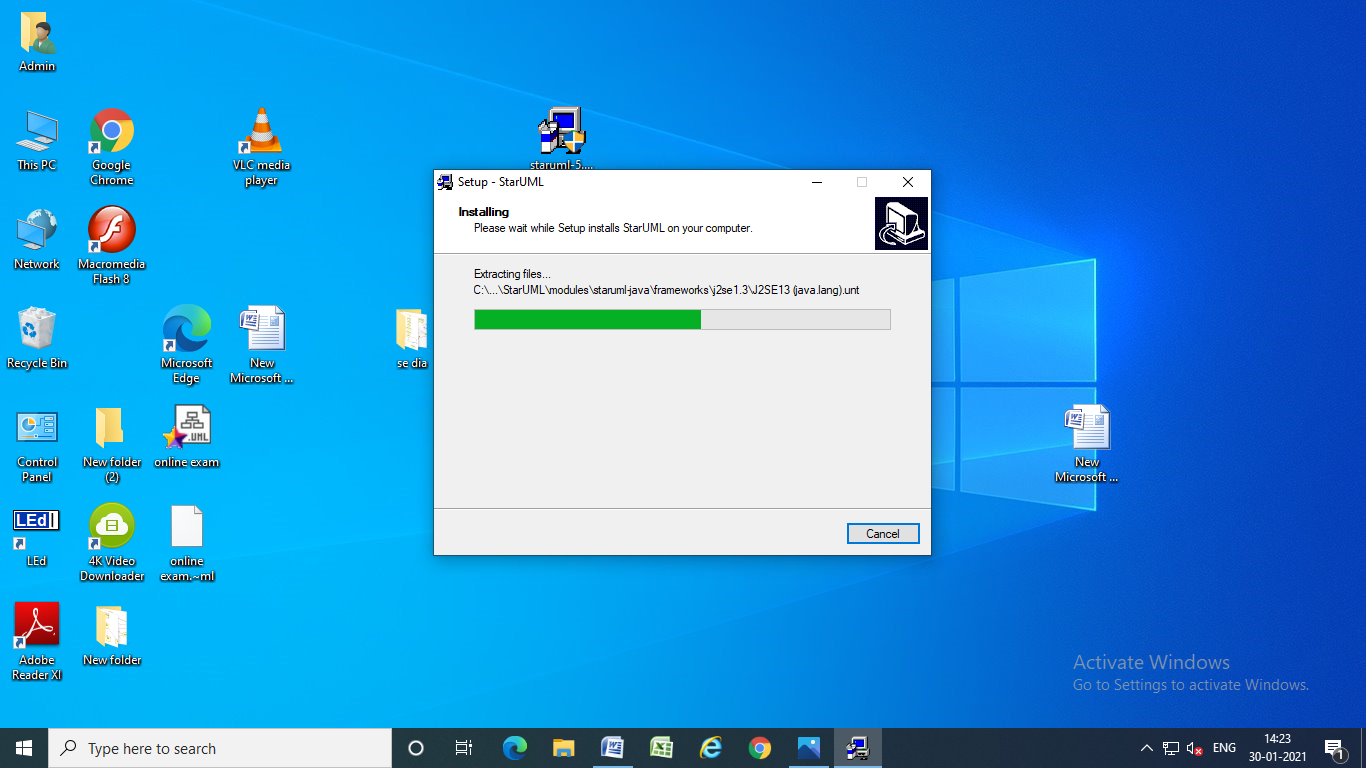
**Step 5:** select create a desktop icon, then click on next button

****

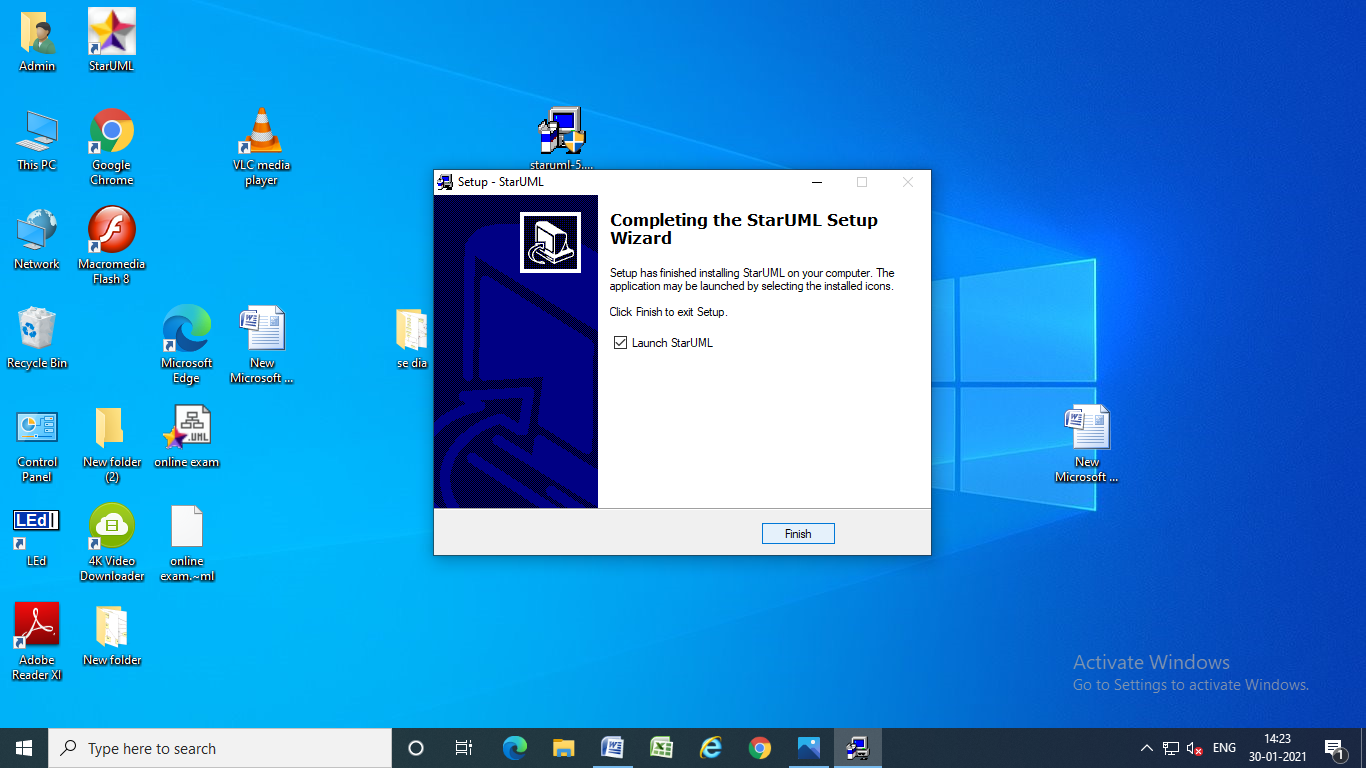
**Step 6:** Install the software



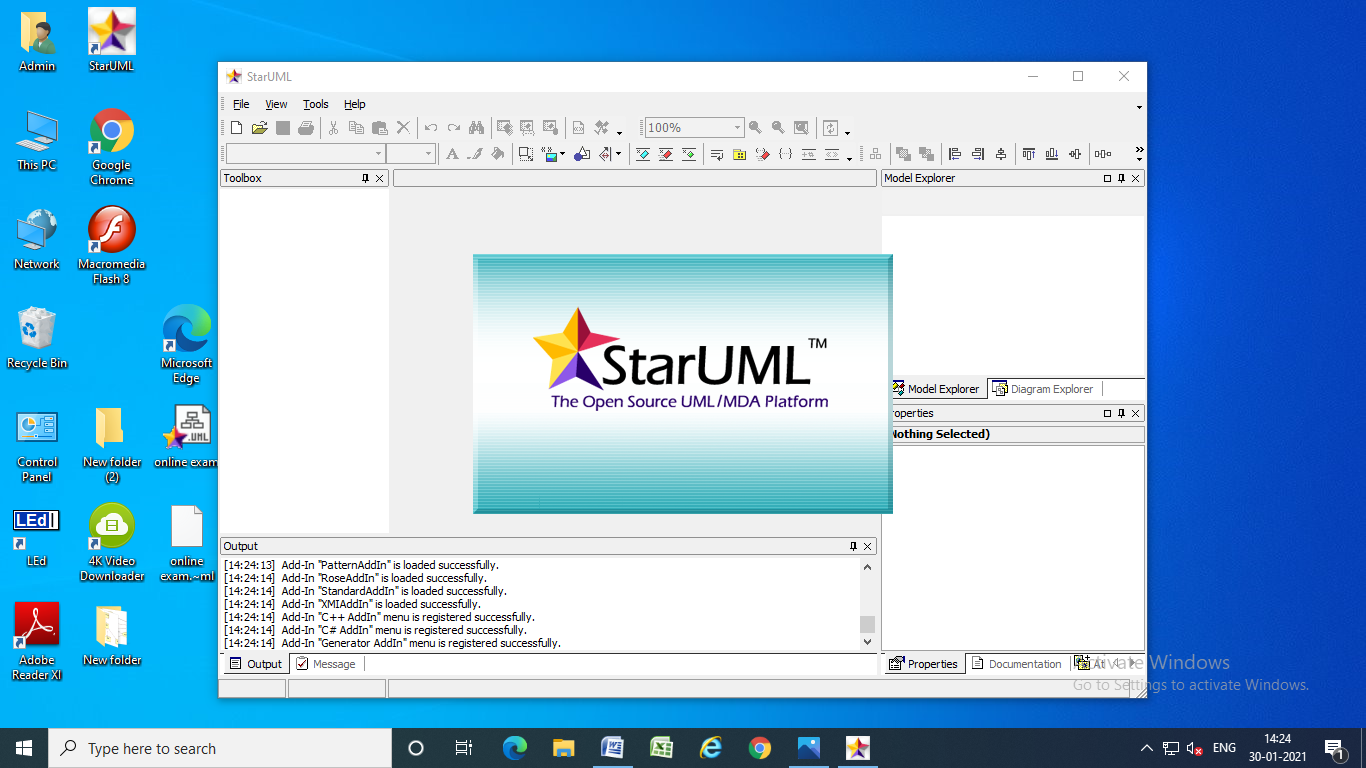
**Step:7** Running installation process



**Step:8** finish the installation, click on finish button

****

**Step:9** automatically open the staruml window.

****

**PROJECTS**

**1.Online Exam Registration**

**AIM: To create a system to perform the Exam Registration system**

**(1) PROBLEM STATEMENT:**

Exam Registration system. is used in the effective dispatch of registration form to all of the students. This system adopts a comprehensive approach to minimize the manual work and schedule resources, time in a cogent manner. The core of the system is to get the online registration form (with details such as name, reg.no etc.,) filled by the student whose testament is verified for its genuineness by the Exam Registration System with respect to the already existing information in the database. This forms the first and foremost step in the processing of exam application. After the first round of verification done by the system, the information is in turn forwarded to the Exam Controller. The application is then processed manually based on the report given by the system. The system also provides the student the list of exam dates. The controller will be provided with fees details to display the current status of application to the student, which they can view in their online interface. After all the necessary criteria has been met, the original information is added to the database and the hall ticket is sent to the student.

**(2) SOFTWARE REQUIREMENT SPECIFICATION:**

**1.0 INTRODUCTION**

Exam Registration System is an interface between the Student and the Exam Controller responsible for the Issue of Hall Ticket. It aims at improving the efficiency in the Issue of Hall ticket and reduces the complexities involved in it to the maximum possible extent.

**1.1 PURPOSE**

If the entire process of 'Issue of Hall ticket' is done in a manual manner then it would takes several days for the hall ticket to reach the student? Considering the fact that the number of students for hall ticket is increasing every year, an Automated System becomes essential to meet the demand. So this system uses several programming and database techniques to elucidate the work involved in this process. As this is a matter of National Security, the system has been carefully verified and validated in order to satisfy it.

**1.2 SCOPE**

• The System provides an online interface to the user where they can fill in their personal details and submit the necessary documents (may be by scanning).

• The controller concerned with the issue of hall ticket can use this system to reduce his workload and process the application in a speedy manner.

• Provide a communication platform between the student and the controller.

• Students will come to know their status of application and the date in which they must subject themselves for manual document verification.

**1.3 DEFINITIONS, ACRONYMS AND THE ABBREVIATIONS**

• Exam Controller - Refers to the super user who is the Central Authority who has been vested with the privilege to manage the entire system.

• Student - One who wishes to obtain the Hall Ticket.

• ERS - Refers to this Examination Registration System.

• HTML - Markup Language used for creating web pages.

• J2EE – Java 2 Enterprise Edition is a programming platform java platform for developing and running distributed java applications.

• HTTP - Hyper Text Transfer Protocol.

• TCP/IP – Transmission Control Protocol/Internet Protocol is the communication protocol used to connect hosts on the Internet.

**1.4 REFERENCES**

IEEE Software Requirement Specification format.

**1.5 TECHNOLOGIES TO BE USED**

• HTML

• JSP

• Javascript

• Java

**1.6 TOOLS TO BE USED**

• Eclipse IDE (Integrated Development Environment)

• Rational Rose tool (for developing UML Patterns)

**1.7 OVERVIEW**

SRS includes two sections overall description and specific requirements - Overall Description will describe major role of the system components and inter-connections. Specific Requirements will describe roles & functions of the actors.

**2.0. OVERALL DESCRIPTION**

**2.1 PRODUCT PERSPECTIVE**

The ERS acts as an interface between the 'student' and the 'exam controller'. This system tries to make the interface as simple as possible and at the same time not risking the security of data stored in. This minimizes the time duration in which the user receives the hall ticket.

**2.2 SOFTWARE INTERFACE**

* Front End Client - The exporter online interface is built using JSP and HTML.
* Web Server – Apache Tomcat Server (Oracle Corporation)
* Back End - Oracle 11g database

**2.3 HARDWARE INTERFACE**

The server is directly connected to the client systems. The client systems have access to the database in the server.

**2.4 SYSTEM FUNCTIONS**

• Secure Registration of information by the Students.

• SMS and Mail updates to the students by the controller.

• Controller can generate reports from the information and is the only authorized personnel to add the eligible application information to the database.

**2.5 USER CHARACTERISTICS**

• Student - They are the people who desire to obtain the hall ticket and submit the information to the database.

• Exam controller - He has the certain privileges to add the registration status and to approve the issue of hall ticket. He may contain a group of persons under him to verify the documents and give suggestion whether or not to approve the dispatch of hall ticket.

**2.6 CONSTRAINTS**

• The applicants require a computer to submit their information.

• Although the security is given high importance, there is always a chance of intrusion in the web world which requires constant monitoring.

• The user has to be careful while submitting the information. Much care is required.

**2.7 ASSUMPTIONS AND DEPENDENCIES**

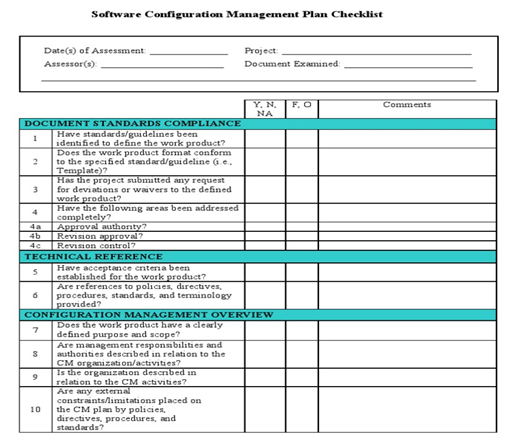
• The Students and Exam Controller must have basic knowledge of computers and English Language.

• The student may be required to scan the documents and send.

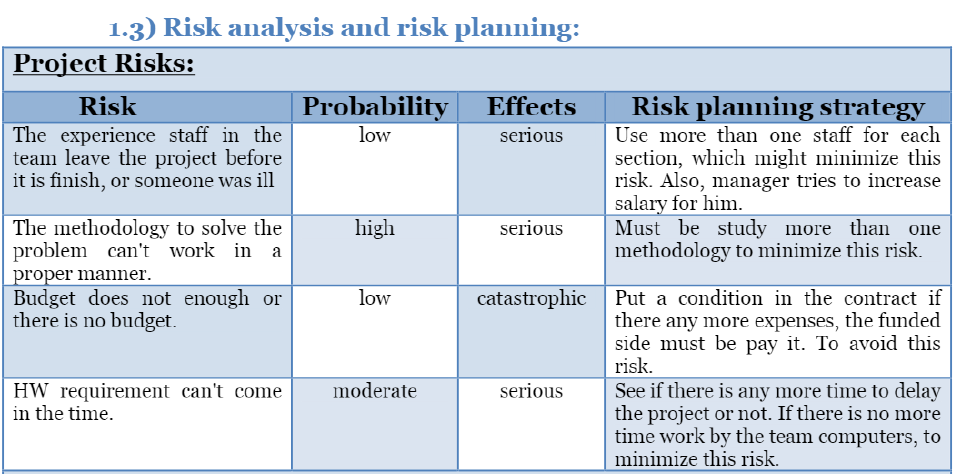
**3. SOFTWARE CONFIGURATION MANAGEMENT**

The purpose of Software Configuration Management is to establish and maintain the integrity of the products of the software project throughout the project's software life cycle.

Software Configuration Management involves identifying the configuration of the software (i.e., selected software work products and their descriptions) at given points in time, systematically controlling changes to the configuration, and maintaining the integrity and traceability of the configuration throughout the software life cycle. The work products placed under software configuration management include the software products that are delivered to the customer (e.g., the software requirements document and the code) and the items that are identified with or required to create these software products

****

**Risk Management related documents**

****

**TEST SCENARIOS FOR SETTING EXAM FUNCTIONALITY**

1. Verify that the application has a portal to add subject wise questions and their options.
2. Verify that the examiner can set the examination details like ‘Exam Name’, ‘Subject’, ‘Exam Code’ etc.
3. Verify that the examiner can set the total number of questions and based on the number of questions, the examiner is presented with the window to add question details.
4. Verify that the examiner can set details for each question – Question, Options, Marks etc.
5. Verify that the examiner can set or leave the option of negative marking.
6. Verify that the examiner can set the passing marks for clearing the exam.
7. Verify that the examiner can set time duration for the whole exam or for individual questions if required.

**Test scenarios for Student’s section**

1. Verify the student can choose the examination based on the exam name or code.
2. Verify that the student should see the options to fill the required details like name, roll number, etc before starting the exam.
3. Verify that after filling the required details user should see the option to begin the exam along with instructions.
4. Verify that once the examination begins a timer gets started based on the test duration.
5. Verify that for each question user is presented with an option for multiple-choice questions(MCQ) type questions.
6. Verify that the user can chose single(radio button) or multiple(checkbox) option based on the type of questions.
7. Verify that on question’s window user is presented with options to move to the previous or next question.
8. Verify that once all the questions are answered or passed user can end the test.
9. Verify that if the time duration for the test gets reached the test automatically ends.
10. Verify that once the test is submitted, the test evaluation is performed considering the positive and negative marking.
11. Verify that on evaluation user is presented with Pass/Fail status along with Marks secured, questions attempted etc.

**USECASE DIAGRAM:**

The Exam Registration use cases in our system are:

1. Login

2. View exam details

3. Register

4. Acknowledgement

5. Fee Processing

**ACTORS INVOLVED:**

1. Student

2. System DB

**USE-CASE NAME: LOGIN**

The student enters his username and password to login and retrieve the information’.

**USE-CASE NAME: VIEW EXAM DETAILS**

The student view the details about the exam schedule which contains Date,time,etc...

**USE-CASE NAME: REGISTER**

The student should notify the fee details that only the student can pay the correct amount

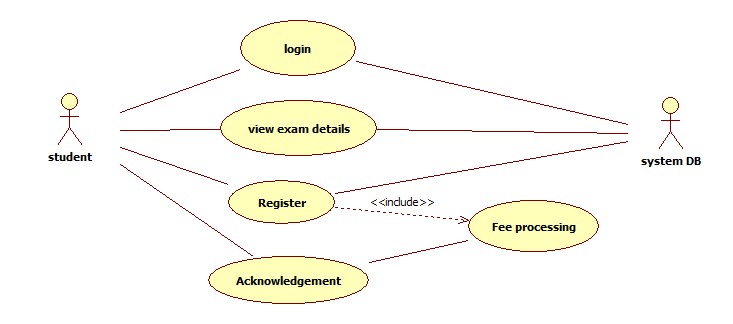
**USE-CASE NAME: ACKNOWLEDGEMENT**

The exam fees should be paid by the student to get the hall ticket from the exam controller.

**USE-CASE NAME: FEE PROCESSING**

All the details should be viewed by both the student and the controller to verify whether all the entered details are correct.

**USE-CASE DIAGRAM:**

****

**CLASS DIAGRAM:**

The class diagram, also referred to as object modeling is the main static analysis diagram. The main task of object modeling is to graphically show what each object will do in the problem domain. The problem domain describes the structure and the relationships among objects.

The Exam Registration System class diagram consists of four two classes of registration system.

1. Student\_details

2. Exam\_details

3. Register

**1) STUDENT\_DETAILS**

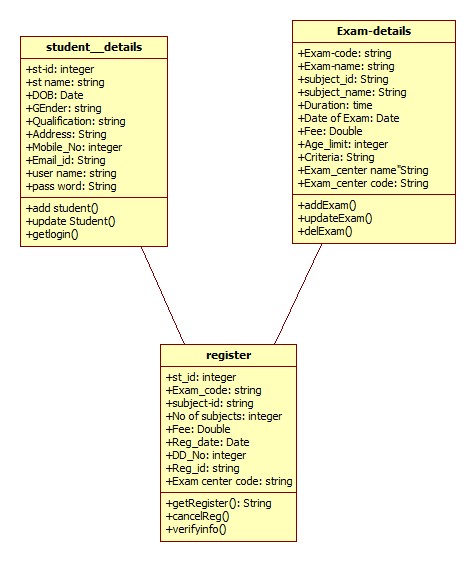
It consists of six attributes and six operations. The attributes id, password, name, age, sex, course. The operations of this class are login(), logout(), conformation(), register(), newfeesdetails().

**2) EXAM\_DETAILS**

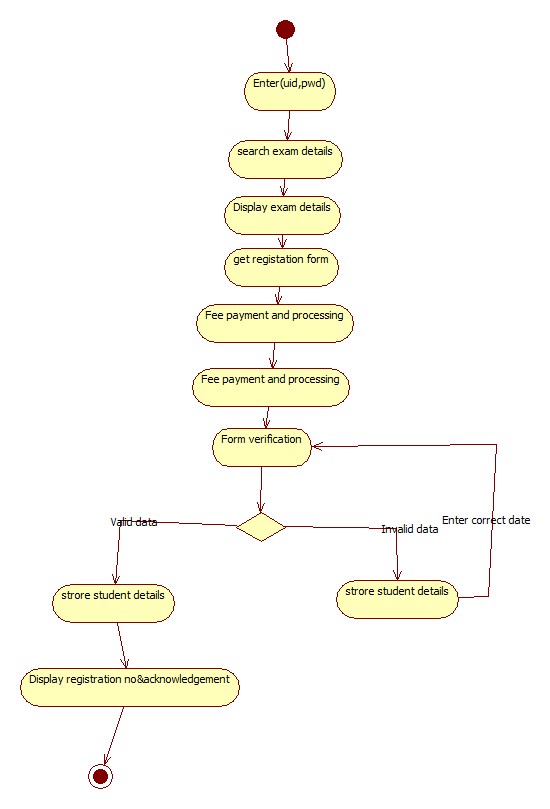
It consists of four attributes and six methods. The attributes are userid, password, examfees, fees due. The methods are login(),logout(), feesdetails(), displayfees(), conformation(), examcontroller().

**3) REGISTER**

This class is used to maintain the registered student information such as, subject registered, date of registration and etc,.

****

**ACTIVITY DIAGRAM:**

****

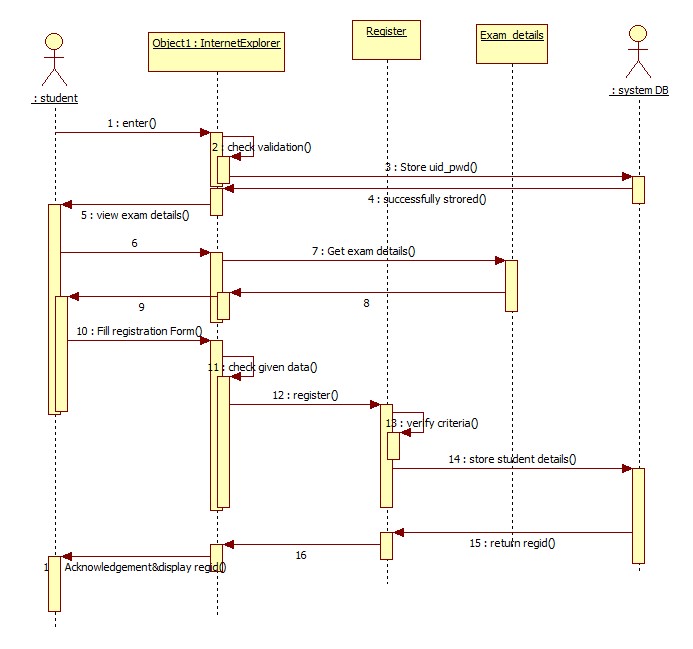
**INTERACTION DIAGRAM**:

A sequence diagram represents the sequence and interactions of a given USE-CASE or scenario. Sequence diagrams can capture most of the information about the system. Most object to object interactions and operations are considered events and events include signals, inputs, decisions, interrupts, transitions and actions to or from users or external devices.

An event also is considered to be any action by an object that sends information. The event line represents a message sent from one object to another, in which the “form” object is requesting an operation be performed by the “to” object. The “to” object performs the operation using a method that the class contains.

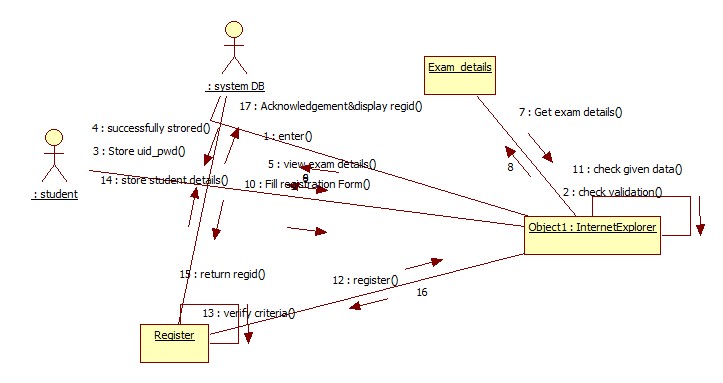
It is also represented by the order in which things occur and how the objects in the system send message to one another.

**SEQUENCE DIAGRAM:**

****

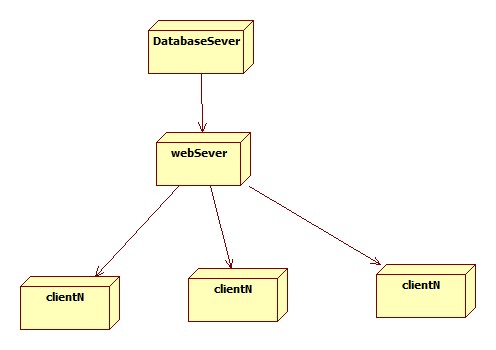
The sequence and collaboration diagram represents that the student enter the information to get the hall ticket and the exam controller issues the hall ticket after verifying the necessary items and this data are stored in the database.

**COLLABORATION DIAGRAM:**

****

**DEPLOYMENT DIAGRAM:**

Deployment diagrams are used to visualize the topology of the physical components of a system where the software components are deployed.

****

**2.LIBRARY MANAGEMENT SYSTEM**

**(1) PROBLEM STATEMENT:**

• Libraries are used to store books, but require a system to navigate to a specific book or specific content within a book. A library database system is an infrastructure that allows users to search books and book content, add/remove, and download selected books.

• The problem faced is that library users require an efficient method to find a specific book or keyword(s) within a book given a continuously expanding library. Efficiency requires that the processing time should stay relatively the same even as the library contents increases.

**2.SOFTWARE REQUIREMENT SPECIFICATION**

**1. INTRODUCTION**

With the increase in the number of readers, better management of libraries system is required. The Library management system focuses on improving the management of libraries in a city or town. “What If you can check whether a book is available in the library through your phone?” or “what if instead of having different library cards for different libraries you can just have one ?” or “you can reserve a book or issue a book from your phone sitting at your home!”. The Integrated Library Management system provides you the ease of issuing, renewing, or reserving a book from an library within your town through your phone. The Integrated Library Management system is developed on the android platform which basically focuses on issuing, renewing and reserving a book.

**1.1 PURPOSE**

The purpose of the project is to maintain the details of books and library members of different libraries. The main purpose of this project is to maintain a easy circulation system between clients and the libraries, to issue books using single library card, also to search and reserve any book from different available libraries and to maintain details about the user (fine, address, phone number).Moreover, the user can check all these features from their home.

**1.2 SCOPE**

* Manually updating the library system into an android based application so that the user can know the details of the books available and maximum limit on borrowing from their computer and also through their phones.
* The ILM System provides information's like details of the books, insertion of new books, deletion of lost books, limitation on issuing books, fine on keeping a book more than one month from the issued date.
* Also user can provide feedback for adding some new books to the library.

**1.3 Definition, Acronyms, Abbreviation:**

 JAVA -> platform independence

 SQL -> Structured query Language

 DFD -> Data Flow Diagram

 CFD -> Context Flow Diagram

 ER -> Entity Relationship

 IDE -> Integrated Development Environment

 SRS -> Software Requirement Specification

**2. OVERALL DESCRIPTION**

**2.1 PRODUCT PRESPECTIVE**

The proposed Library Management System will take care of the current book detail at any point of time. The book issue, book return will update the current book details automatically so that user will get the update current book details.

**Product function:**

* The key goal of this initiative is to minimize the work done manually.
* Book Problems, Refunds, and Fine Calculation/Management can be handled by this program. Generating separate
* Record-Keeping Records according to end-user criteria

**User characteristics:**

We have 2 levels of users

* + User Module: The user can verify the availability of the books in the user module.  
    Book return
* Administration module: The sub-modules in the administration module are as follows.
  + Register user
  + Entry book details
  + Book issue

**General Constraints:**

To have updated & accurate values, any change about the library book must be registered.

**Assumption and dependencies:**

All the details entered will be reliable and up to date. This software package is built with the assistance of the sun micro framework using java as the front end. The back end of Microsoft SQL Server 2005, which is supported by Windows 7.

**Specific Requirement:**

**External Interface Requirement:**

It should be simple and easy for consumers to understand and use. It should also be an immersive interface. For the user and administrator, the device should prompt the login to the program and correct feedback parameters.

**User Interface:**

The program offers a decent graphical interface for the user that can be run on the device by an administrator, performing the necessary tasks such as designing, reviewing, displaying the book information.

* Allows users to access quick reports in between real hours, such as Book Issues/Returned, etc.
* Based on various requirements, stock verification, and search service.

**Hardware interface:**

* Operating system: window
* Hard disk:40 GB
* RAM: 256 MB
* Processor: Pentium(R)Dual-core CPU

**Software interface :**

* Java language
* Net beans IDE 7.0.1
* MS SQL server 2005

**Communication interface:**

Window

**2.4.1 FUNCTIONAL REQUIREMENT**

 **R.1:Register**

 Description : First the user will have to register/sign up. There are two different type of users.

 The library manager/head : The manager have to provide details about the name of library ,address, phone number, email id.

 Regular person/student : The user have to provide details about his/her name of address, phone number, email id.

 **R.1.1: Sign up**

 Input: Detail about the user as mentioned in the description.

 Output: Confirmation of registration status and a membership number and password will be generated and mailed to the user.

 Processing: All details will be checked and if any error are found then an error message is displayed else a membership number and password will be generated.

 **R.1.2 : Login**

 Input: Enter the membership number and password provided.

 Output : User will be able to use the features of software.

 **R.2 : Manage books by user.**

 **R.2.1 : Books issued.**

 Description : List of books will be displaced along with data of return.

**R.2.2 : Search**

 Input : Enter the name of author's name of the books to be issued.

 Output : List of books related to the keyword.

 **R.2.3 : Issues book**

 State : Searched the book user wants to issues.

 Input : click the book user wants.

 Output : conformation for book issue and apology for failure in issue.

 Processing : if selected book is available then book will be issued else error will be displayed.

 **R.2.4 : Renew book**

 State : Book is issued and is about to reach the date of return.

 Input : Select the book to be renewed.

 Output : conformation message.

 Processing : If the issued book is already reserved by another user then error message will be send and if not then conformation message will be displayed.

 **R.2.5 : Return**

 Input ; Return the book to the library.

 Output : The issued list will be updated and the returned book will be listed out.

 **R.2.6 ; Reserve book**

 Input ; Enter the details of the book.

 Output : Book successfully reserved.

 Description : If a book is issued by someone then the user can reserve it ,so that later the user can issue it.

 **R.2.6 Fine**

 Input : check for the fines.

 Output : Details about fines on different books issued by the user.

 Processing : The fine will be calculated, if it crossed the date of return and the user did not renewed if then fine will be applied by Rs 10 per day.

 **R.3 Manage book by librarian**

 **R.3.1 Update details of books**

 **R.3.1.1 Add books**

 Input : Enter the details of the books such as names ,author ,edition, quantity.

 Output : confirmation of addition.

 **R.3.1.2 Remove books**

 Input : Enter the name of the book and quantity of books.

 Output : Update the list of the books available.

* + 1. **Non Functional Requirements**

• **Usability Requirement**

The system shall allow the users to access the system from the phone using android application. The system uses a android application as an interface. Since all users are familiar with the general usage of mobile app, no special training is required. The system is user friendly which makes the system easy.

• **Availability Requirement**

The system is available 100% for the user and is used 24 hrs a day and 365 days a year. The system shall be operational 24 hours a day and 7 days a week.

• **Efficiency Requirement**

Mean Time to Repair (MTTR) - Even if the system fails, the system will be recovered back up within an hour or less.

• **Accuracy**

The system should accurately provide real time information taking into consideration various concurrency issues. The system shall provide 100% access reliability.

• **Performance Requirement**

The information is refreshed depending upon whether some updates have occurred or not in the application. The system shall respond to the member in not less than two seconds from the time of the request submittal. The system shall be allowed to take more time when doing large processing jobs. Responses to view information shall take no longer than 5 seconds to appear on the screen.

• **Reliability Requirement**

The system has to be 100% reliable due to the importance of data and the damages that can be caused by incorrect or incomplete data. The system will run 7 days a week, 24 hours a day.

* 1. **USER CHARACTERSTICS**

We have 3 levels of users :

 User module: In the user module, user will check the availability of the books.

Issue book

Reserve book

Return book

Fine details

 Library module:

Add new book

Remove books

Update details of book

 Administration module:

The following are the sub module in the administration module :

Register user

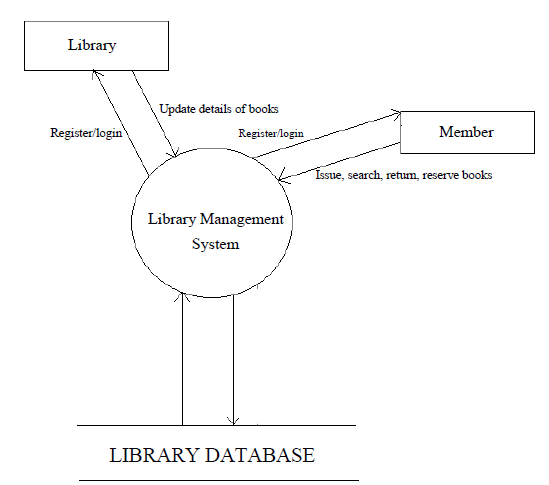
Entry book details

Book issue

**2.6 CONSTRAINTS**

Any update regarding the book from the library is to be recorded to have update & correct values, and any fine on a member should be notified as soon as possible and should be correctly calculated.

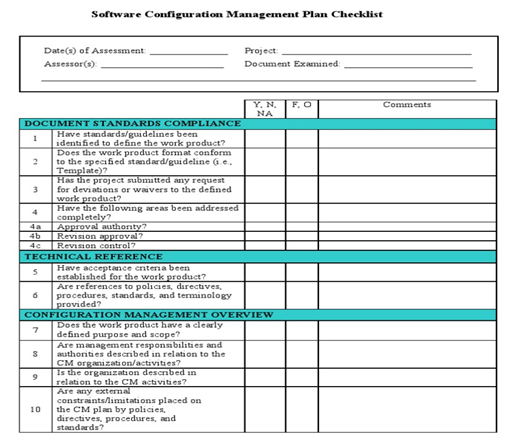
**2.7 FLOW DIAGRAM**



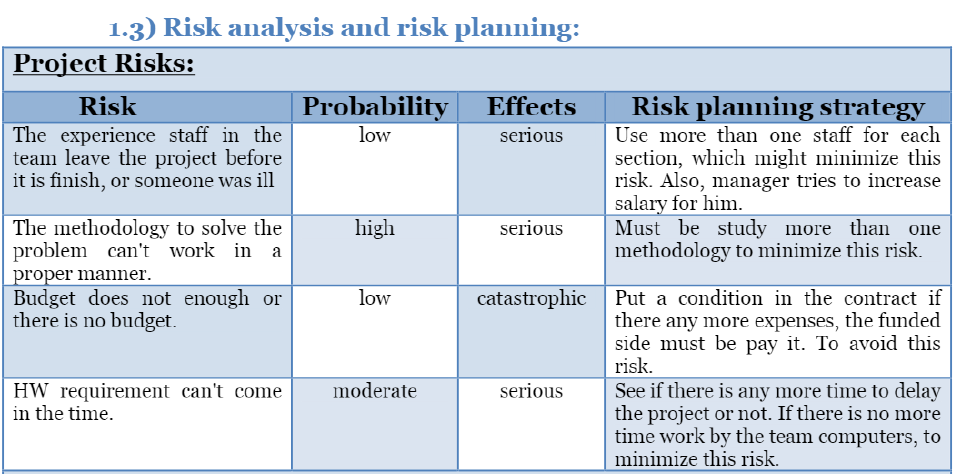
**3)Software Configuration Management**

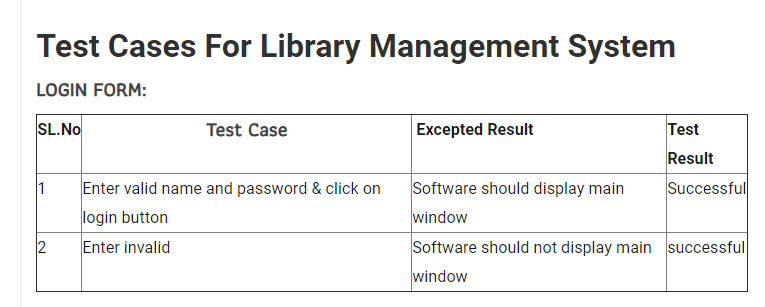
The purpose of Software Configuration Management is to establish and maintain the integrity of the products of the software project throughout the project's software life cycle.

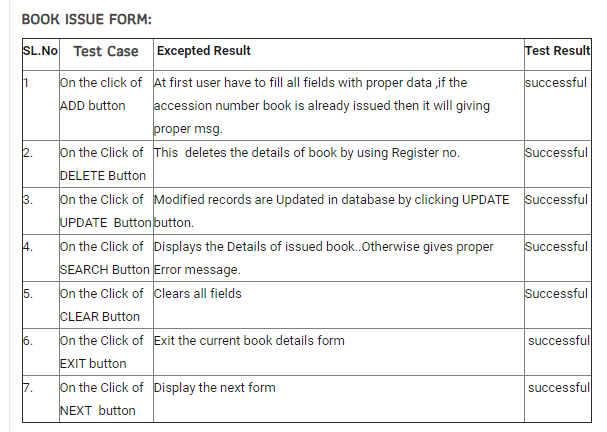
Software Configuration Management involves identifying the configuration of the software (i.e., selected software work products and their descriptions) at given points in time, systematically controlling changes to the configuration, and maintaining the integrity and traceability of the configuration throughout the software life cycle. The work products placed under software configuration management include the software products that are delivered to the customer (e.g., the software requirements document and the code) and the items that are identified with or required to create these software products



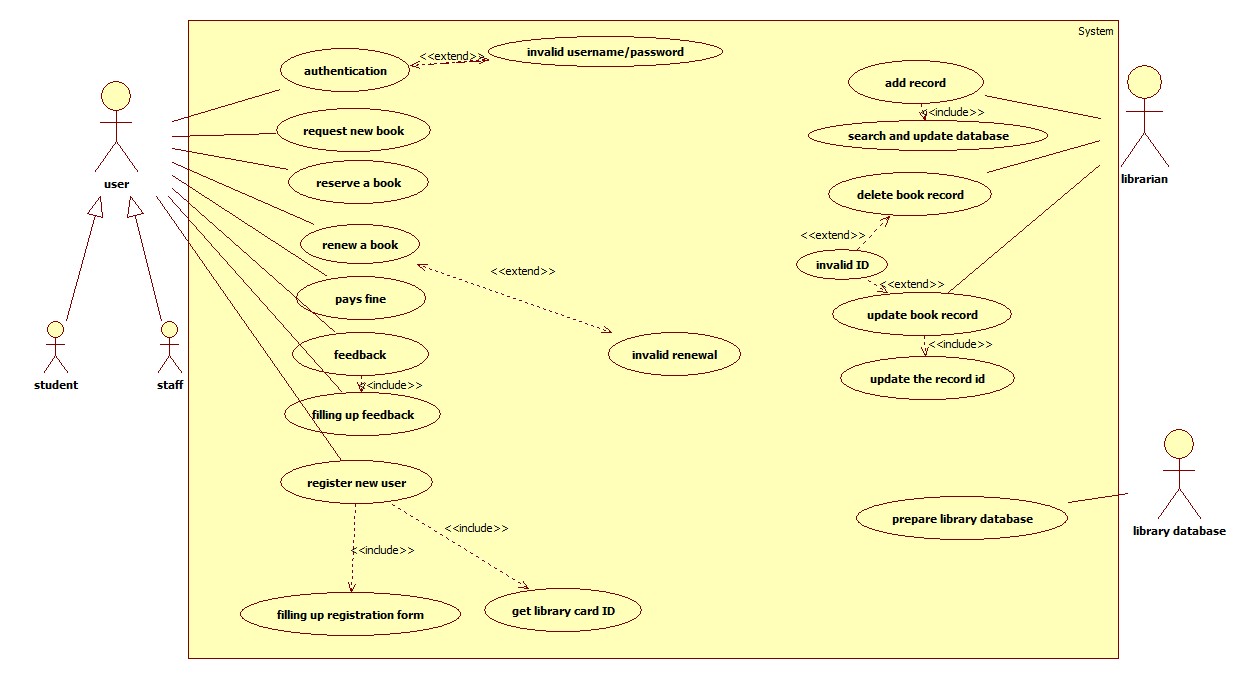
**Risk Management related documents**

****

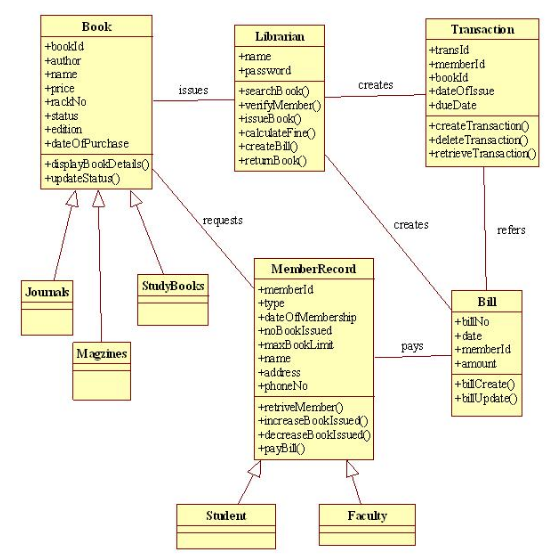
****

****

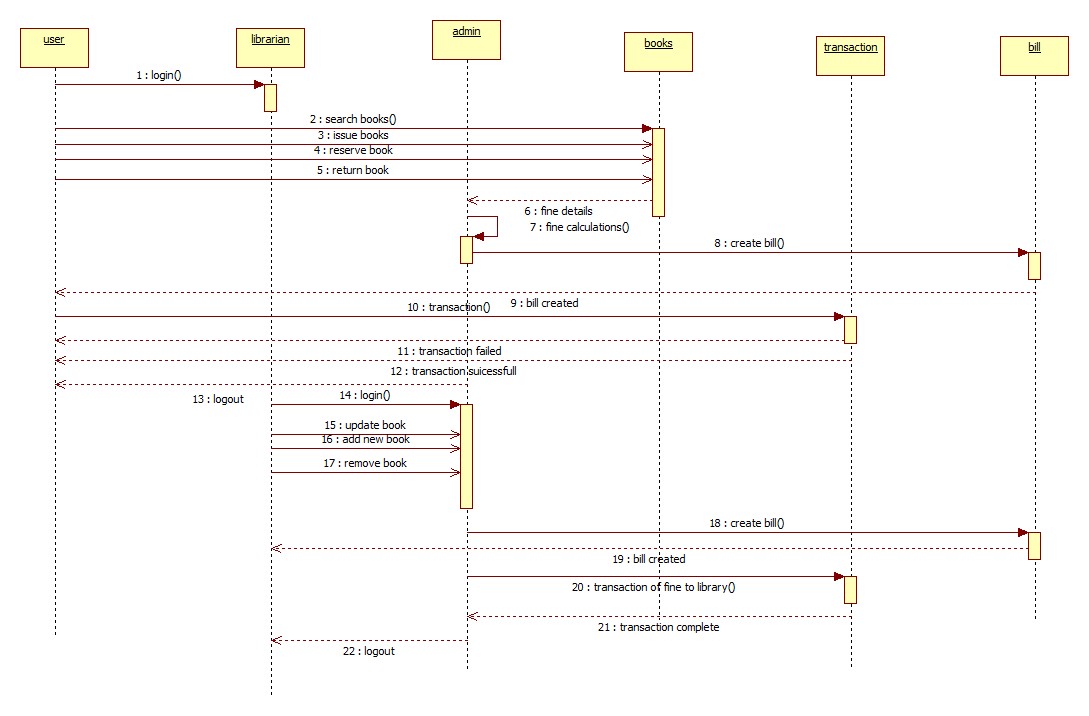
**USE CASE DIAGRAM**

****

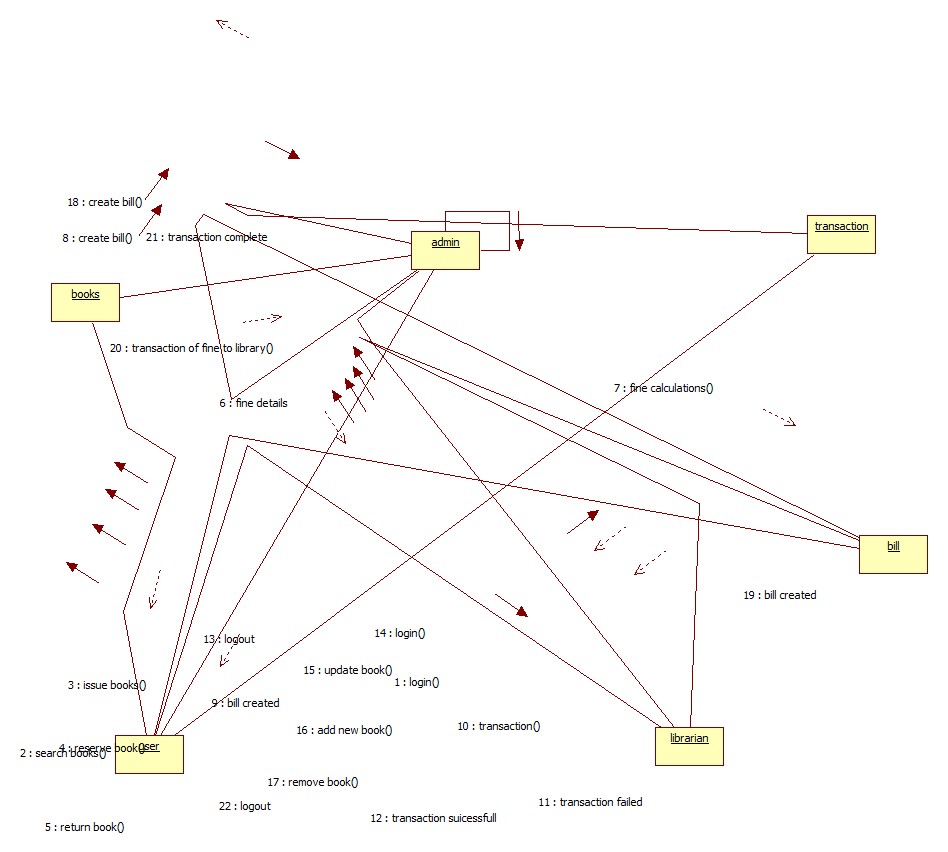
**CLASS DIAGRAM**

****

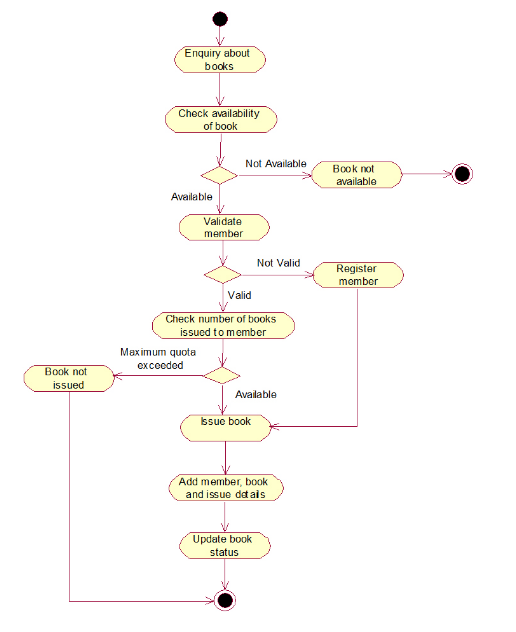
**SEQUENTIAL DIAGRAM**

****

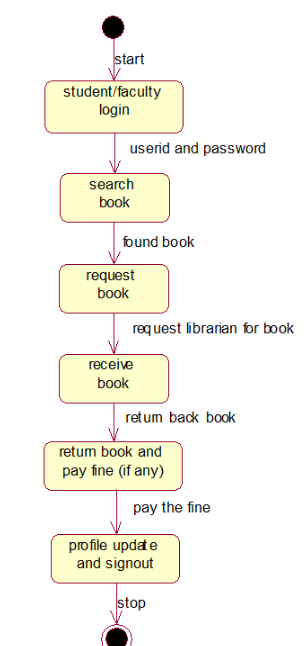
**COLABARATION DIAGRAM**

****

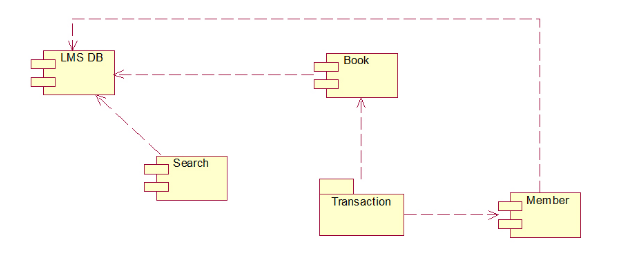
**ACTIVTY DIAGRAM**

****

**STATECHART DIAGRAM**

****

**COMPONENT DIAGRAM**

****

**ADDITIONAL PROJECTS**

**1. ATM (ATOMATED TELLER MACHINE) SYSTEM**

**(I) PROBLEM STATEMENT**

The project entitled ATM system has a drastic change to that of the older version of banking system, customer feel inconvenient with the transaction method as it was in the hands of the bank employees. In our ATM system, the above problem is overcome here, the transactions are done in person by the customer thus makes the customers feel safe and secure. Thus the application of our system helps the customer in checking the balance and transaction of the amount by validating the pin number therefore ATM system is more user friendly Software

**Software Requirement Specification (SRS) for ATM**

**1. Introduction**

An automated teller machine (ATM) is computerized telecommunications device that provides a financial institution's customers a secure method of performing financial transactions, in a public space without the need bank. Through ATM, customers interact with a user-friendly interface that enables them to access their bank accounts and perform various transactions.

**1.1 Purpose**

This SRS defines External Interface, Performance and Software System Attributes requirements of ATM. This document is intended for the following group of people:-

* Developers:- for the purpose of maintenance and new releases of the software.
* Management of the bank.
* Documentation writers.
* Testers.

**1.2 Scope**

This software facilitates the user to perform various transactions in his account without going to bank. This software offers benefits such cash withdrawals, balance transfers, deposits, inquiries, credit card advances and other banking related operations for customers.

The software takes as input the login Id and the bank account number of the user for login purposes. The outputs then comprise of an interactive display that lets the user select the desirable function that he wants to perform.

**1.3 Definitions, Acronyms, and Abbreviations.**

AC- Alternate Current

AIMS- ATM Information Management System.

ATM- An unattended electronic machine in a public place, connected to a data system and related equipment and activated by a bank customer to obtain cash withdrawals and other banking services.

Internet- An interconnected system of networks that connects computers around the world via the TCP/IP protocol.

TCP/IP: Transmission Control Protocol/Internet Protocol.

VGA Video Graphics Adaptor is a display standard.

**1.4 References**

The references for the above software are as follows:-

i. www.google.co.in

ii. www.wikipedia.com

iii. IEEE. Software Requirements Specification Std. 830-1993.

**1.5 Overview**

Section 1.0 discusses the purpose and scope of the software.

Section 2.0 describes the overall functionalities and constraints of the software and user characteristics.

Section 3.0 details all the requirements needed to design the software.

**2. The Overall Description**

**2.1 Product Perspective**

* The ATM is a single functional unit consisting of various subcomponents.
* This software also allows performing various other functions apart from just accessing his bank account such as mobile bill clearings etc.
* Some of its hardware components are cassettes, memory, drives, dispensers i.e. for receipts and cash, a card reader, printer, switches, a console, a telephone dialer port, a networking port and disks.

**2.2 Product Functions**

The major functions that ATM performs are described as follows:-

* Language Selection:- After the user has logged in, the display provides him with a list of languages from which he can select any one in order to interact with the machine throughout that session.
* Account Maintenance:- The various functions that a user can perform with his account are as follows:-
  + Account Type:-The user has the freedom to select his account type to which all the transactions are made, i.e. he can select whether the account is current account or savings account etc.
  + Withdrawal/Deposit: The software allows the user to select the kind of operation to be performed i.e. whether he wants to withdraw or deposit the money.
  + Amount:- The amount to be withdrawn or deposited is then mentioned by the user.
  + Denominations: - The user is also provided with the facility to mention the required denominations. Once he enters his requirements the machine goes through its calculations on the basis of current resources to check whether it is possible or not. If yes, the amount is given to the user otherwise other possible alternatives are displayed.
  + Money Deposition: - Money deposition shall be done with an envelope. After typing the amount to be deposited and verification of the same, the customer must insert the envelope in the depositary.
  + Balance Transfer: - Balance transfer shall be facilitated between any two accounts linked to the card for example saving and checking account.
  + Balance Enquiry: - Balance enquiry for any account linked to the card shall be facilitated. .
* Billing: - Any transaction shall be recorded in the form of a receipt and the same would be dispensed to the customer. The billing procedures are handled by the billing module that enable user to choose whether he wants the printed statement of the transaction or just the updating in his account.
* Cancelling: - The customer shall abort a transaction with the press of a Cancel key. For example on entering a wrong depositing amount. In addition the user can also cancel the entire session by pressing the abort key and can start a fresh session all over again.
* Map locating other machines: - The machine also has a facility of displaying the map that marks the locations of other ATM machines of the same bank in the entire city.
* Mobile Bills Clearings:- The machine also allows the user to clear off his pending mobile bills there only, if the name of his operator is mentioned there in the list. The machine displays the list of the companies supported by that bank to the user.
  1. **User Characteristics**

There are different kinds of users that will be interacting with the system. The intended users of the software are as follows:-

* User A: A novice ATM customer. This user has little or no experience with electronic means of account management and is not a frequent user of the product. User A will find the product easy to use due to simple explanatory screens for each ATM function. He is also assisted by an interactive teaching mechanism at every step of the transaction, both with the help of visual and audio help sessions.
* User B: An experienced customer. This user has used an ATM on several occasions before and does most of his account management through the ATM. There is only a little help session that too at the beginning of the session thus making the transaction procedure faster.
* Maintenance Personnel: A bank employee. This user is familiar with the functioning of the ATM. This user is in charge of storing cash into the ATM vault and repairing the ATM in case of malfunction. This user is presented with a different display when he logs in with the administrator’s password and is provided with options different from that of normal user. He has the authority to change or restrict various features provided by the software in situations of repairing.
  1. **Constraints**

The major constraints that the project has are as follows: -

* The ATM must service at most one person at a time.
* The number of invalid pin entries attempted must not exceed three. After three unsuccessful login attempts, the card is seized/blocked and need to be unlocked by the bank.
* The simultaneous access to an account through both, the ATM and the bank is not supported.
* The minimum amount of money a user can withdraw is Rs 100/- and the maximum amount of money a user can withdraw in a session is Rs.10,000/- and the maximum amount he can withdraw in a day is Rs 20,000/-
* Before the transaction is carried out, a check is performed by the machine to ensure that a minimum amount of Rs 1000/- is left in the user’s account after the withdrawal failing which the withdrawal is denied.
* The minimum amount a user can deposit is Rs 100/- and the maximum amount he can deposit is Rs 10,000/-.
* A user can select only that cellular operator for mobile bill clearings that is supported by the bank.
* The software requires a minimum memory of 20GB the database used should be Oracle7.0.
* There shall be a printer installed with the machine to provide the user with the printed statement of the transaction.
* For voice interactions, speakers should also be there to accompany the machine.

1. **External Interface Requirements**

3.1.1 User Interface Requirements

The interface provided to the user should be a very user-friendly one and it should provide an optional interactive help for each of the service listed. The interface provided is a menu driven one and the following screens will be provided:-

1. A login screen is provided in the beginning for entering the required username/pin no. and account number. 2. An unsuccessful login leads to a reattempt (maximum three) screen for again entering the same information. The successful login leads to a screen displaying a list of supported languages from which a user can select any one. 3. In case of administrator, a screen will be shown having options to reboot system, shut down system, block system, disable any service. 4. In case of reboot/ shut down, a screen is displayed to confirm the user’s will to reboot and also allow the user to take any backup if needed. 5. In case of blocking system, a screen is provided asking for the card no. By entering the card no of a particular user, system access can be blocked for him. 6. Administrator is also provided with a screen that enables him to block any service provided to the user by entering the name of the service or by selecting it from the list displayed. 7. After the login, a screen with a number of options is then shown to the user. It contains all the options along with their brief description to enable the user to understand their functioning and select the proper option. 8. A screen will be provided for user to check his account balance.

9. A screen will be provided that displays the location of all other ATMs of same bank elsewhere in the city. 10. A screen will be provided for the user to perform various transactions in his account.

The following reports will be generated after each session dealt with in the machine:-

1. The login time and logout time along with the user’s pin no and account number is registered in the bank’s database. 2. The ATM’s branch ID through which the session is established is also noted down in the bank’s database. 3. Various changes in the user’s account after the transactions, if any, are reported in the database. 4. A printed statement is generated for the user displaying all the transactions he performed.

Other various user interface requirements that need to be fulfilled are as follows:-

The display screen shall be of 10" VGA color type. The display screen shall have 256 color resolution. The display screen shall also support touch screen facility. The speakers shall support Yamaha codecs. The keypad shall consist of 16 tactile keys. There shall be 8 tactile function keys. The keyboard will be weather resistant. The transaction receipt shall be 3.1" × 6". The statement receipt shall be 4.2" × 12". The deposit envelopes shall be 9" long and 4" wide.

**3.1.2 Hardware Interface Requirements**

There are various hardware components with which the machine is required to interact. Various hardware interface requirements that need to be fulfilled for successful functioning of the software are as follows:- The ATM power supply shall have a 10/220 V AC manual switch. The ATM card should have the following physical dimensions:- o Width - o Height - o Thickness - The card reader shall be a magnetic stripe reader The card reader shall have Smart card option. The slot for a card in the card reader may include an extra indentation for the embossed area of the card. In effect it acts as a polarization key and may be used to aid the correct insertion orientation of the card. This is an additional characteristic to the magnetic field sensor which operates off the magnetic stripe and is used to open a mechanical gate on devices such as ATMs. There shall be a 40 column dot matrix receipt printer. There shall be a 40 column dot matrix statement printer. The receipt dispenser shall be a maximum of 4" width and 0.5" thickness. The statement dispenser shall be a maximum of 5" width and 0.5" thickness. The envelope depository shall be a maximum of 4.5" width, 10" length and 0.5" thickness. Screen resolution of at least 800X600-required for proper and complete viewing of screens. Higher resolution would not be a problem.

3.1.3 Software Interface Requirements

In order to perform various different functions, this software needs to interact with various other software’s. So there are certain software interface requirements that need to be fulfilled which are listed as follows:-

The transaction management software used to manage the transaction and keep track of resources shall be BMS version 2.0. The card management software used to verify pin no and login shall be CMS version 3.0. Yamaha codecs 367/98 for active speakers. The database used to keep record of user accounts shall be Oracle version7.0.

**Functional Requirements:**

|  |  |
| --- | --- |
| ATM-FR-X | ATM functional requirement |
| ATM-FR-1 | The card reader determines the account number from entered card |
| ATM-FR-2 | ATM is in the idle state when there is no operation |
| ATM-FR-3 | The user is prompted to enter a PIN after a card is entered. |
| ATM-FR-4 | A menu is displayed to the user with following options: Withdraw, Deposit, Pay bill, Account update and Exit. |
| ATM-FR-5 | The cash Dispenser has the ability to dispense cash. |
| ATM-FR-6 | A transaction record can be printed upon demand. |
| ATM-FR-7 | The card is ejected when the session is completed. |
| ATM-FR-8 | The envelope feeder accepts envelopes |
| ATM-FR-9 | The envelope feeder is aware when an envelope has been inserted. |
| ATM-FR-10 | The printer determines whether there is sufficient paper and ink. |
| ATM-FR-11 | The ATM default to idle state. |
| ATM-FR-12 | The cash dispenser is aware of the cash amount available. |
| ATM-FR-13 | The user is prompted to make between 1 and 3 bill payments. |
| ATM-FR-14 | The keypad accepts input from the user. |
| ATM-FR-15 | The ATM is aware of state of the envelope drawer. (Which can be either empty or full). |
| ATM-FR-16 | The ATM can withhold a bank card |
| ATM-FR-17 | Transactions can be canceled at any prompt by the user pressing the CANCEL button. |

**Non-Functional Requirements:**

|  |  |
| --- | --- |
| ATM-NFR-X | ATM non-functional requirements |
| ATM-NFR-1 | The ATM unit consists of a display, a card reader, a cash dispenser, an envelope drawer, an envelope slot, a keypad and a printer. |
| ATM-NFR-2 | The keypad is a set of buttons that includes the following: 10 buttons labeled with numbers 0 to 9, an OKAY button, a CLEAR button, CANCEL button, and dynamic buttons that perform different actions |
| ATM-NFR-3 | A PIN must be entered within 20 seconds. |
| ATM-NFR-4 | The user must enter PIN correctly within three attempts. |
| ATM-NFR-5 | ATM suspends further access using a particular card if the associated PIN is entered incorrectly 3 times in succession. |
| ATM-NFR-6 | ATM must be secure |
| ATM-NFR-7 | ATM can be shut down and restarted. |
| ATM-NFR-8 | The envelope drawer can be open and refilled wit envelopes |
| ATM-NFR-9 | The envelope feeder can be opened so any envelopes that have been deposited can be removed |
| ATM-NFR-10 | The cash dispenser can be opened and refilled with cash |
| ATM-NFR-11 | The printer can be opened and refilled with paper. |

**3.1 Design Documentation**

1. Login

1.1 Brief description: This use case describes how the user logs into the System.

1.2 Flow of events:

1.2.1 Basic flow: This use case starts with the actor wishes to log in to the ATM System.

1. The system requests the user to enter the name and PIN.

2. The actor enters the name and PIN.

3. The system validates the name and the PIN and logs the user into the system.

1.2.2 Alternative flow:

1. If the user enters the wrong name and the PIN then the system displays an error message.

2. The actor can either return to the basic flow or cancel login at which point use case ends.

1.3 Pre conditions: None

1.4 Post conditions: User will perform corresponding transaction.

2. Transaction

2.1 Brief description: This describes the transaction that the user is doing.

2.2 Flow of events:

2.2.1 Basic flow: This use case starts after the user has logged on to the system.

1. The system requests the user to enter the type of transaction of either withdrawal or deposit and asks for customer information.

2. The actor enters the type of transaction and the customer information.

3. The system displays the corresponding transaction screen.

2.2.2 Alternative flow: If the customer enters any wrong information then the system displays an error message.

2.3 Pre Condition: The user logs on to the system.

2.4 Post Condition: Based on the transaction he gets the transaction screen.

3. Maintain Information about Customer

3.1 Brief description: This describes how administrator takes care of customer information.

**3.2 Flow of events:**

3.2.1 Basic flow: This use case starts after the administrator has logged into the system.

1. The system asks the administrator whether he wants to add or delete customer information.

2. The administrator then enters the type of maintenance.

3.2.2 Alternative flow: None

3.3 Pre Condition: The administrator logs on to the system before this use case begin.

3.4 Post Condition: Administrator gets the corresponding maintenance screen according to his choice.

3.2.1.1 Adding Customer

3.2.1.1.1 Basic flow:

1. This use case starts when the administrator has chosen to add customer’s information.

2. The system asks the administrator to enter customer information.

3. The administrator enters the customer information.

4. The system displays the updated information.

3.2.1.1.2 Alternative flow: If the administrator enters any wrong information the system displays an error message.

3.2.1.2 Deleting Customer

3.2.1.2.1 Basic flow:

1. This use case starts when the administrator has chosen to delete an existing customer from the system.

2. The system asks the administrator to enter the customer information.

3. Administrator enters the corresponding user information.

4. The system then displays updated results.

3.2.1.2.2 Alternative flow: If the administrator has entered any wrong information then the system displays administrator error message.

3.2.1.3 Updating an existing Customer account

3.2.1.3.1 Basic flow:

1. This use case starts when the administrator has chosen to update the customer’s information. 2. The system asks the administrator to enter the customer information.

3. The administrator enters the customer information.

4. The system displays the updated information.

3.2.1.3.2 Alternative flow: If the administrator has entered any wrong information then the system displays administrator error message.

**Testing Phase related documents.**

**ATM Machine UI Testing**

* Check in the screen all labels button, links & images are appearing correctly.
* Check whatever written on the screen are visible.
* Check the application UIO is responsive.
* Check the ATM Machine is a full touch screen, or it also supports Keyboard and Touch screen both the functionality.

**ATM Machine Positive Test Cases**

* Check the ATM card is as per the specification document.
* After entering the Debit/Credit card in the card, reader users should be able to select language and operation like withdrawal, language change, mini statement, and other options.
* When an ATM card is entered in the card reader, it should verify the card.
* Check during any transaction the ATM Machine accepts card and Pin details.
* Check after successfully enter the pin and complete the process, the user should be able to take out the money.
* After taking out the money, the money receipt should also print.
* After successfully withdraw the amount, the user should be log out from the sessions.
* Check if the user wants to print the money receipt (mini statement), it should be done by following the menu options.
* If the user enters more amounts then the account balance, then the user should get an error message.
* The ATM should have a waiting period between user session log out and active another account.
* In the ATM Machine, the user should be able to Use a Master card, Visa card, and Rupay Card.
* Verify after the transaction, the printed slip has the correct information or not.
* Verify the entered pin is encrypted or not.
* Verify the touch functionality is working correctly or not.
* Check whether the ATM is providing all types of accounts to do operations like Savings and the correct account.

**Negative Test Cases**

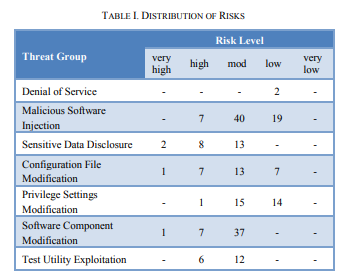
* Check the ATM Machine can accept the cards and pin or not.
* If a user enters an invalid pin, then an error message should be displayed.
* If allowed bank ATM card is entered, then only the user able to do the operations.
* Check is the ATM Machine can find the wrong pin or not.
* Check if the card is entered in the wrong way, the machine should find that and display an error message.
* Check if the user three times the wrong password, then the account should be locked.
* If the user has a lack of money, then he should receive a warning message.
* If the user inserted an expired card, then the user should not be able to perform any action.
* If the user inserted less then 100 amount, then the amount should not out from the ATM Machine.
* If the ATM Machine has dispatched the money, then the money should not again enter into the ATM.
* The machine does not accept either Visa or [MasterCard](https://www.mastercard.us/) or both debit/credit cards.
* If the user enters the wrong denominations, then a warning message should display.
* If the user has entered more then the daily limit amount, then the transaction should be canceled, and a warning message should be displayed on the screen.
* The transaction should be canceled if the user clicks on the Cancel button.
* Check an error message is a display or not when the ATM does have the currency on it.
* Check whether an error message is displayed or not when there is some network issue.
* Check after the money release it is asking or not for the user confirmation to print the transaction receipt.
* Check during the transaction if there is some power failure or network issue comes then the transaction should be marked as nulled and no amount should be dispatched.

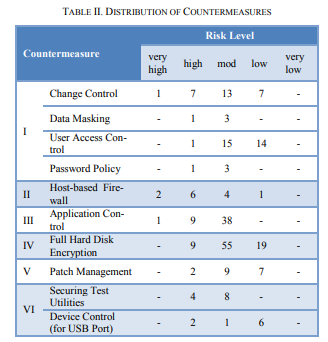
1. **Software Configuration Management**

The purpose of Software Configuration Management is to establish and maintain the integrity of the products of the software project throughout the project's software life cycle.

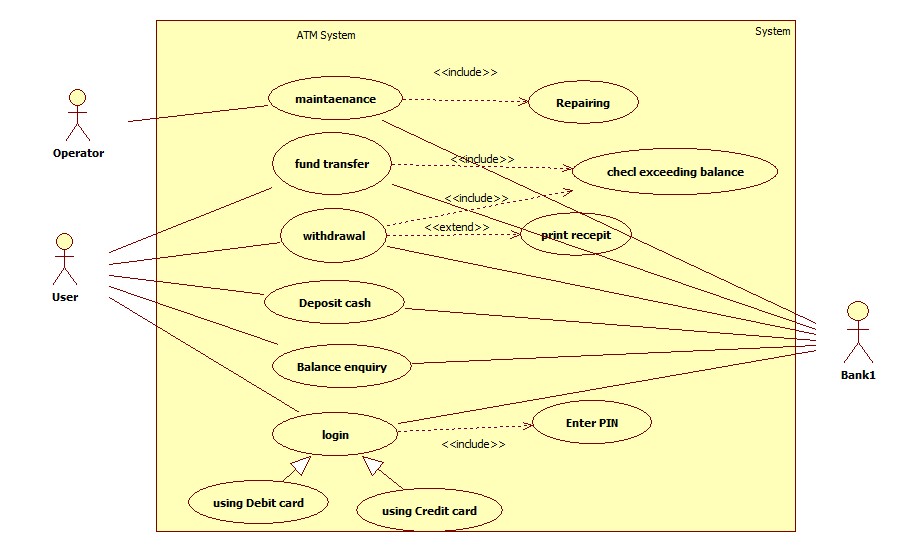
Software Configuration Management involves identifying the configuration of the software (i.e., selected software work products and their descriptions) at given points in time, systematically controlling changes to the configuration, and maintaining the integrity and traceability of the configuration throughout the software life cycle. The work products placed under software configuration management include the software products that are delivered to the customer (e.g., the software requirements document and the code) and the items that are identified with or required to create these software products

**Risk Management related documents**

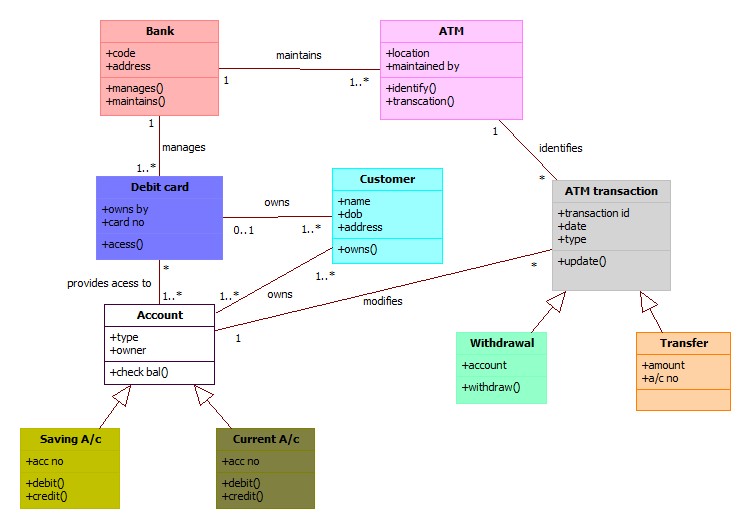




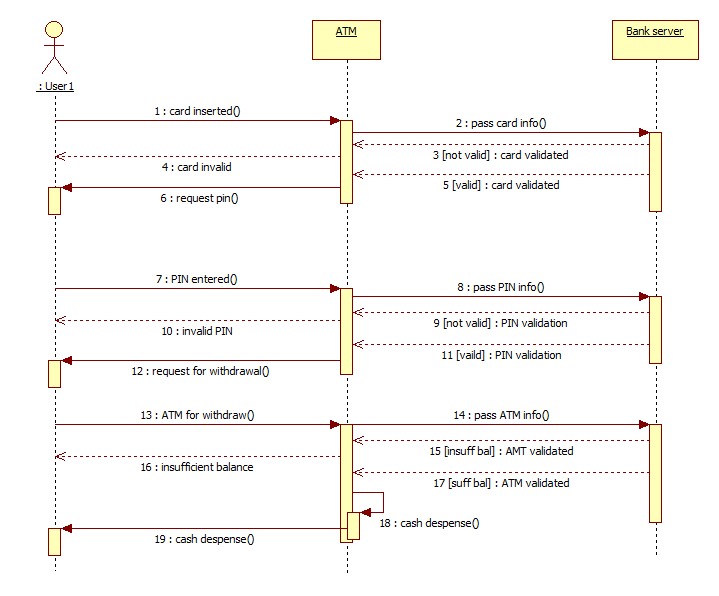
**USECASE DIAGRAM**

****

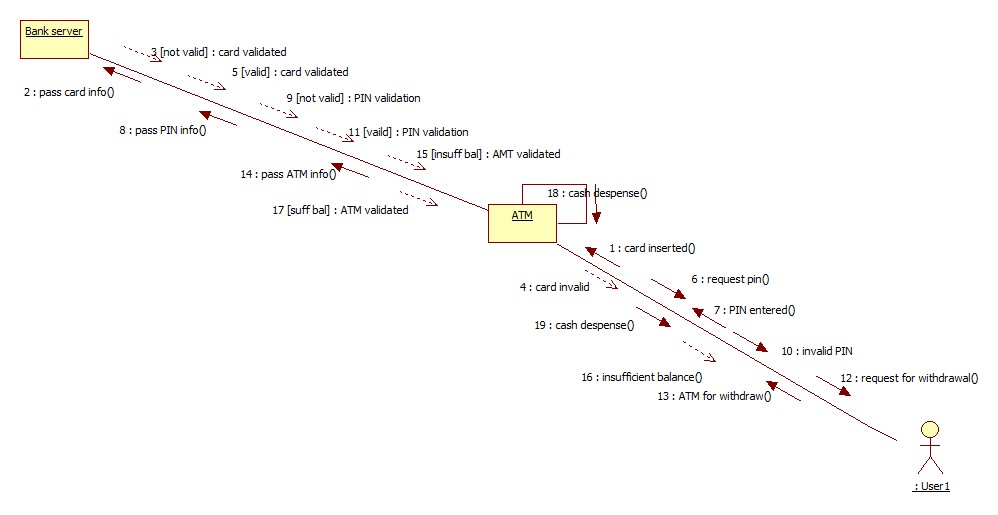
**CLASS DIAGRAM**

****

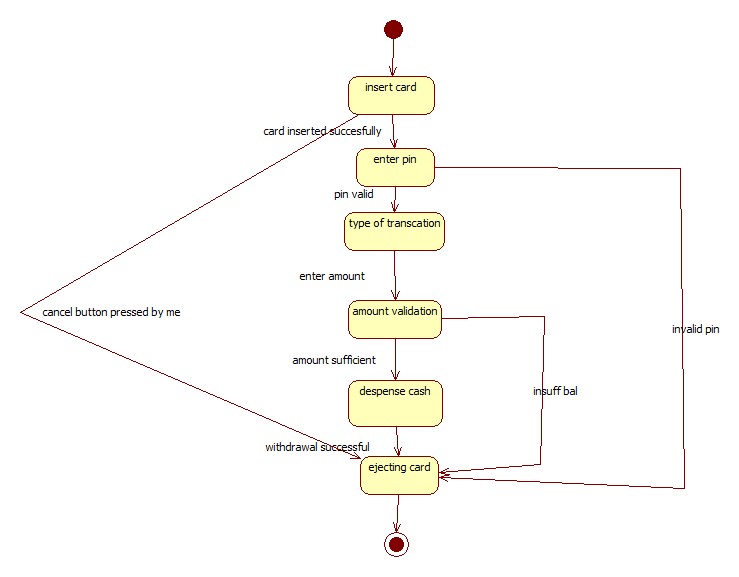
**SEQUENCE DIAGRAM**

****

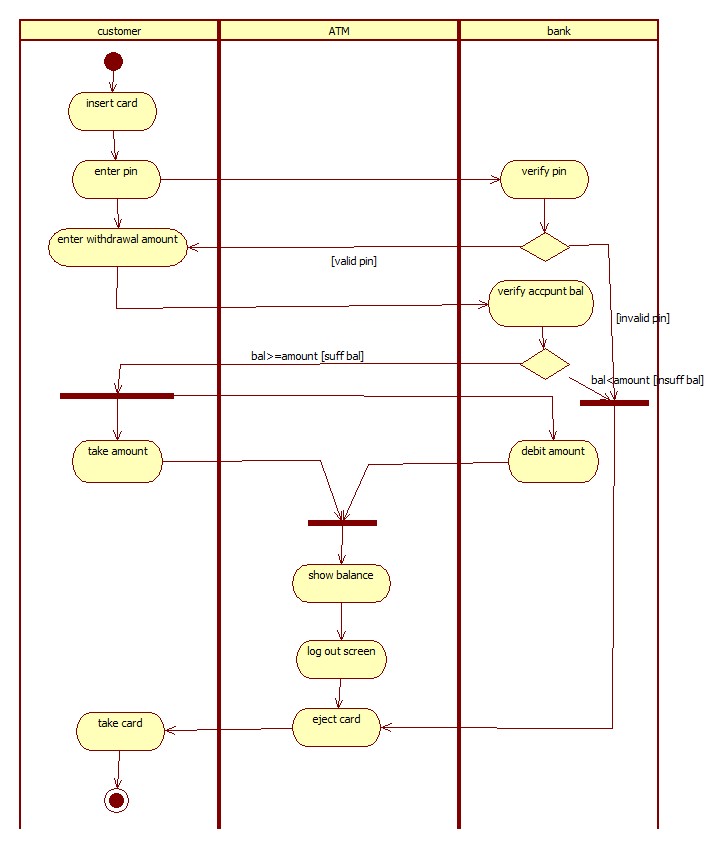
**COLLABARATION DIAGRAM**



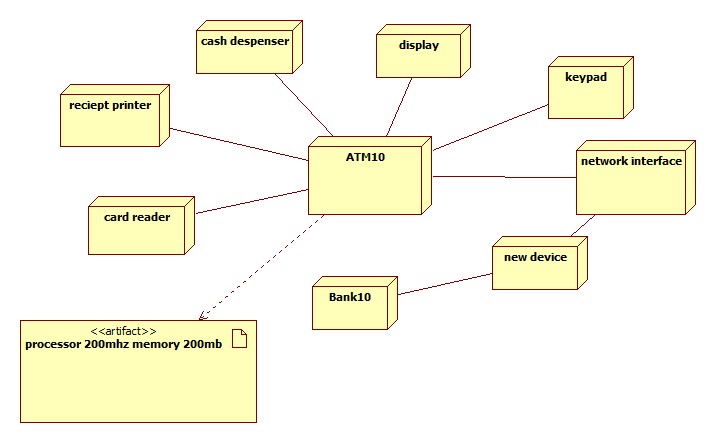
**STATECHART DIAGRAM**



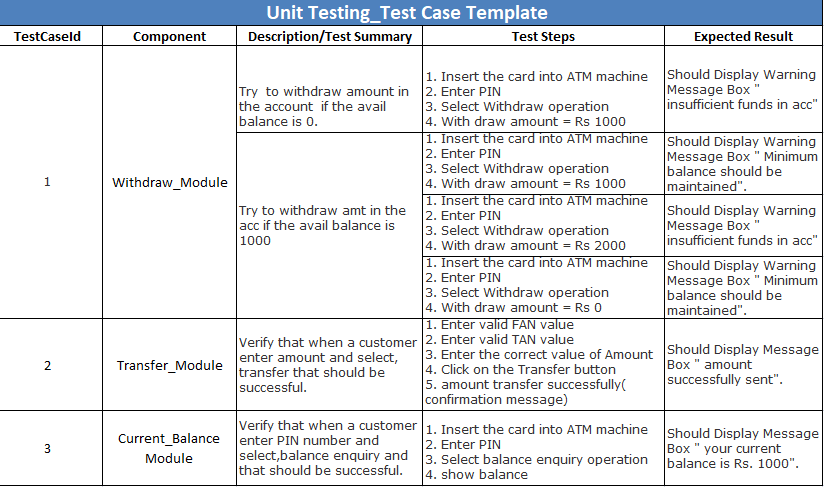
**ACTIVITY DIAGRAM**



**DEPLOYMENT DIAGRAM**



**Develop test cases for unit testing and integration testing**

****

|  |  |  |  |
| --- | --- | --- | --- |
| **Integration Testing\_Test Case Template** | | | |
| **Test Case ID** | **Test Case Objective** | **Test Case Description** | **Expected Result** |
| 1 | Check the interface link between the Login and Current\_Balance module | Enter login credentials and click on the Next button | To be directed to the Menu |
| 2 | Check the interface link between the Current\_Balance and Transfer Module | From Menu select the transfer button, enter details and click a Conform button | Entered amount should deposit in the appropriate customer account |