

# PRACTICAL

# FILE



**Department:** Computer Science and Engineering

**Session:** January to June

**Subject:** Software Engineering

**Subject Code:**

**Semester:** 6<sup>th</sup>



## Syllabus

1. Study and usage of Open Project or similar software to draft a project plan
2. Study and usage of Open Project or similar software to track the progress of a project
3. Preparation of Software Requirement Specification Document, Design Documents and Testing Phase related documents for some problems
4. Preparation of Software Configuration Management and Risk Management related documents
5. Study and usage of any Design phase CASE tool
6. To perform unit testing and integration testing
7. To perform various white box and black box testing techniques
8. Testing of a web site

**List of Practical**

<b>Sr. No.</b>	<b>Topic</b>
1.	Study and usage of Open Project or similar software to draft a project plan
2.	Study and usage of Open Project or similar software to track the progress of a project
3.	Preparation of Software Requirement Specification Document, Design Documents and Testing Phase related documents for some problems
4.	Preparation of Software Configuration Management and Risk Management related documents
5.	Study and usage of any Design phase CASE tool
6.	To perform unit testing and integration testing
7.	To perform various white box and black box testing techniques
8.	Testing of a web site



**Sample File**

**HOSPITAL MANAGEMENT SYSTEM  
USING UML**

**A PROJECT REPORT**

## **TABLE OF CONTENTS**

<b>S.NO.</b>	<b>TITLE</b>	<b>PAGE NO.</b>
1.	INTRODUCTION TO PROJECT	
2.	OVERALL DESCRIPTION	
3.	EXTERNAL INTERFACE REQUIREMENTS	
4.	SYSTEM FEATURES	
5.	OTHER NON-FUNCTIONAL REQUIREMENTS	
6.	SYSTEM STUDY	
7.	FEASIBILITY STUDY	
8.	SYSTEM ANALYSIS	
9.	DATA FLOW DIAGRAMS	
10.	INTRODUCTION TO UML	
11.	INTRODUCTION TO RATIONAL ROSE	
12.	USE CASE DESCRIPTION	
13.	USE CASE DIAGRAM	
14.	CLASS DIAGRAM	
15.	SEQUENCE DIAGRAMS	
16.	COLLABORATION DIAGRAMS	
17.	STATE CHART DIAGRAMS	
18.	ACTIVITY DIAGRAMS	
19.	COMPONENT DIAGRAM	
20.	DEPLOYMENT DIAGRAM	
21.	SCOPE	
22.	CONCLUSION	
23.	BIBLIOGRAPHY	

# INTRODUCTION TO THE PROJECT

## PURPOSE

Hospital management system is the most important part for Doctors. As test is the most important for knowing cause of any diseases. Besides maintaining other records of the hospital every year or after some time, the price of conducting tests changes, then it issues a list of prices of every test. Hence, software concerning Hospital Management should be well organized for its efficient working. So, this project is an attempt to relieve the burden of manual system by providing fully automated and a secure system. The function, which this system provides, is maintaining the records of the patient pertaining to his name, address, tests and tests results. The password system check allows only the authorized person to use the system in order to maintain the security of the system and if the user forgets his password then he can use his security question and password to enter the system.

# **OVERALL DESCRIPTION**

## **PRODUCT PERSPECTIVE**

In the proposed system, all the work done is computerized. All the Tests of the Patients are recorded in database through the computers. Because of maintenance of records we can easily identify each user. This System also provides the facilities of maintaining records of the Lab reports & Calculating Bills. The software will be general software which can be used on any computer system satisfying certain requirements.

## **PRODUCT FUNCTIONS**

- Handling the management of different types of Users.
- Handling the management of Administrator such as Change of Password, Login, and Logout etc.
- Handling the full management of Patient such as Enquiry, Test, Bill Generation, and Menu etc.
- Handling the full management of Staff such as updating Staff Masters, Calculating Staff Salary, and Enquiry about Staff etc.
- Handling the different types of report generation such as Pending Reports, Bill, and Menu List etc.
- Handling the working of some facilities for the Pathologic Lab such as Calculator, Notepad, and MS-Office XP.
- General and Standardized Health Packages for the OPD and IPD Patients

# **EXTERNAL INTERFACE REQUIREMENTS**

## **HARDWARE INTERFACES**

An important factor in the proposed system is hardware. In hardware we describe the peripheral devices that are attached to the system and from these devices we insert values as input to attain output.

Mentioned below are the hardware used in this project:-

- Intel Pentium IV.
- 128 RAM
- Hard disk of 40 GB or more.
- CD drive.
- Mouse.
- Colored monitor.
- Keyboard.
- Inkjet or laser printer.

## **SOFTWARE INTERFACES**

- Unified Modeling Language: Rational Rose
- Operating System: Windows XP



## **SYSTEM FEATURES**

- One Integrated View to Patients for Billing, Collection, Discharge Detail, Patient Medical History etc.
- Package supports Adaptability & Scalability of software making it more robust.
- General and Standardized Health Packages for the OPD and IPD Patients
- Authentication and verification of entries through Audit Trail Facility
- Easy Query Handling for instant decision of Bed Allocation for Patients, and request for the Bed Transfers
- Effective Search facility to search any type of information related to Patient history
- Graphical Presentation of the Data for Top **Management** for analysis.
- Comprehensive Performance Reports.
- Built in Work Flow **Management** for all functional areas
- Multiple Store Accounting
- Interface facility with the Smart Card Technology
- Interface with Bar Code

## OTHER NON FUNCTIONAL REQUIREMENTS

### QUALITY ATTRIBUTES:

- **Graphical User Interface:** This System is totally graphical user interface. This mean all work is done by Pointing Devices or we can say that there is no need of remembering the commands by the users.
- **Understandable by novice users:** This System is easily understandable by the Novice Users. Novice users are those that are used the system first time.
- **Flexible:** This system is flexible as new modules can be added easily and faster into the system.
- **Time saving:** - Records maintaining manually takes a lots of Paper work. This work was very hectic and consumes a lot of labor work. In this all Data is stored Computerized. Data is also being stored directly & time & labor work is saved.
- **Easy To Change:** - With the help of proposed system it will be easier to change data regarding the candidates and to store at the same time.
- **Easy To Understand:** - This system is easily understandable by the user. Novice users also used this system easily.
- **Chance of Errors:** - Since database is maintained automatically there is no chance of error inconsistencies there. If an error occurs in System in any case can be maintained easily.
- **User Friendly:** - It will provide a user friendly interface and will be easily accessible.
- **Fast Process:** Speed of Processing is high.
- **Security:** - Degree of Security is high in this system because only valid user can access the system.

# **SYSTEM STUDY**

System study is a comprehensive management study for investigation of overall problem, which is to be solved by which there will be increase in the efficiency and effectiveness of the system. When a new system is developed, we need to study the old system and its relationship with an outside of the system.

## **EXISTING SYSTEM STUDY**

A Hospital is a place where Patients come up for general diseases. Hospitals provide facilities like:-

- Consultation by Doctors on Diseases.
- Diagnosis for diseases.
- Providing treatment facility.
- Facility for admitting Patients (providing beds, nursing, medicines etc.)
- Immunization for Patients/Children.

Various operational works that are done in a Hospital are:-

- Recording information about the Patients that come.
- Generating bills.
- Recording information related to diagnosis given to Patients.
- Keeping record of the Immunization provided to children/patients.
- Keeping information about various diseases and medicines available to cure them.

These are the various jobs that need to be done in a Hospital by the operational staff and Doctors. All these works are done on papers.

The work is done as follows:-

- Information about Patients is done by just writing the Patients name, age and gender. Whenever the Patient comes up his information is stored freshly.
- Bills are generated by recording price for each facility provided to Patient on a separated sheet and at last they all are summed up.

- Diagnosis information to patients is generally recorded on the document, which contains Patient information. It is destroyed after some time period to decrease the paper load in the office.
- Immunization records of children are maintained in pre-formatted sheets, which are kept in a file.
- Information about various diseases is not kept as any document. Doctors themselves do this job by remembering various medicines.

All this work is done manually by the receptionist and other operational staff and lot of papers are needed to be handled and taken care of. Doctors have to remember various medicines available for diagnosis and sometimes miss better alternatives as they can't remember them at that time.

## **PROBLEMS WITH CONVENTIONAL SYSTEM**

1. **Lack of immediate retrievals:** - The information is very difficult to retrieve and to find particular information like- E.g. - To find out about the patient's history, the user has to go through various registers. This results in inconvenience and wastage of time.
2. **Lack of immediate information storage:** - The information generated by various transactions takes time and efforts to be stored at right place.
3. **Lack of prompt updating:** - Various changes to information like patient details or immunization details of child are difficult to make as paper work is involved.
4. **Error prone manual calculation:** - Manual calculations are error prone and take a lot of time this may result in incorrect information. For example, calculation of patient's bill based on various treatments.
5. **Preparation of accurate and prompt reports:** - This becomes a difficult task as information is difficult to collect from various registers.

## **GOALS OF PROPOSED SYSTEM**

1. **Planned approach towards working:** - The working in the organization will be well planned and organized. The data will be stored properly in data stores, which will help in retrieval of information as well as its storage.

2. **Accuracy:** - The level of accuracy in the proposed system will be higher. All operation would be done correctly and it ensures that whatever information is coming from the center is accurate.

3. **Reliability:** - The reliability of the proposed system will be high due to the above stated reasons. The reason for the increased reliability of the system is that now there would be proper storage of information.

4. **No Redundancy:** - In the proposed system utmost care would be that no information is repeated anywhere, in storage or otherwise. This would assure economic use of storage space and consistency in the data stored.

5. **Immediate retrieval of information:** - The main objective of proposed system is to provide for a quick and efficient retrieval of information. Any type of information would be available whenever the user requires.

6. **Immediate storage of information:** - In manual system there are many problems to store the largest amount of information.

7. **Easy to Operate:** - The system should be easy to operate and should be such that it can be developed within a short period of time and fit in the limited budget of the user.

## **FEASIBILITY STUDY**

Depending on the results of the initial investigation the survey is now expanded to a more detailed feasibility study. "**FEASIBILITY STUDY**" is a test of system proposal according to its workability, impact of the organization, ability to meet needs and effective use of the resources. It focuses on these major questions:

1. What are the user's demonstrable needs and how does a candidate system meet them?
2. What resources are available for given candidate system?
3. What are the likely impacts of the candidate system on the organization?
4. Whether it is worth to solve the problem?

During feasibility analysis for this project, following primary areas of interest are to be considered. Investigation and generating ideas about a new system does this.

### **STEPS IN FEASIBILITY ANALYSIS**

- Eight steps involved in the feasibility analysis are:
- Form a project team and appoint a project leader.
- Prepare system flowcharts.
- Enumerate potential proposed system.
- Define and identify characteristics of proposed system.
- Determine and evaluate performance and cost effective of each proposed system.
- Weight system performance and cost data.
- Select the best-proposed system.
- Prepare and report final project directive to management.

### **TECHNICAL FEASIBILITY**

A study of resource availability that may affect the ability to achieve an acceptable system. This evaluation determines whether the technology needed for the proposed system is available or not.

- Can the work for the project be done with current equipment existing software technology & available personal?
- Can the system be upgraded if developed?
- If new technology is needed then what can be developed?

This is concerned with specifying equipment and software that will successfully satisfy the user requirement. The technical needs of the system may include:

### **Front-end and back-end selection**

An important issue for the development of a project is the selection of suitable front-end and back-end. When we decided to develop the project we went through an extensive study to determine the most suitable platform that suits the needs of the organization as

well as helps in development of the project.

The aspects of our study included the following factors.

#### **Front-end selection:**

1. It must have a graphical user interface that assists employees that are not from IT background.
2. Scalability and extensibility.
3. Flexibility.
4. Robustness.
5. According to the organization requirement and the culture.
6. Must provide excellent reporting features with good printing support.
7. Platform independent.
8. Easy to debug and maintain.
9. Event driven programming facility.
10. Front end must support some popular back end like Ms Access.

#### **Back-end Selection:**

1. Multiple user support.
2. Efficient data handling.
3. Provide inherent features for security.
4. Efficient data retrieval and maintenance.

5. Stored procedures.
6. Popularity.
7. Operating System compatible.
8. Easy to install.
9. Various drivers must be available.
10. Easy to implant with the Front-end.

The technical feasibility is frequently the most difficult area encountered at this stage. It is essential that the process of analysis and definition be conducted in parallel with an assessment to technical feasibility. It centers on the existing computer system (hardware, software etc.) and to what extent it can support the proposed system. The system is technically viable because can run on PC with the following hardware and software configuration:-

Processor : Pentium IV

Memory : 128MB RAM

Hard disk capacity : 40GB

Operating System : Windows XP service

Front end Tool : Visual Basic 6.0

Back End Tool : MS access

There is nothing that is not feasible technically.

## **ECONOMICAL FEASIBILITY**

Economic justification is generally the “Bottom Line” consideration for most systems. Economic justification includes a broad range of concerns that includes cost benefit analysis. In this we weight the cost and the benefits associated with the candidate system and if it suits the basic purpose of the organization i.e. profit making, the project is making to the analysis and design phase. The financial and the economic questions during the preliminary investigation are verified to estimate the following:

- The cost to conduct a full system investigation.
- The cost of hardware and software for the class of application being considered.



- The benefits in the form of reduced cost.
- The proposed system will give the minute information, as a result the performance is improved which in turn may be expected to provide increased profits.
- This feasibility checks whether the system can be developed with the available funds. The **Hospital Management System** does not require enormous amount of money to be developed. This can be done economically if planned judiciously, so it is economically feasible. The cost of project depends upon the number of man hours required.

### **OPERATIONAL FEASIBILITY**

It is mainly related to human organizations and political aspects. The points to be considered are:

What changes will be brought with the system?

What organization structures are disturbed?

What new skills will be required? Do the existing staff members have these skills? If not, can they be trained in due course of time?

The system is operationally feasible as it very easy for the End users to operate it. It only needs basic information about Windows platform.

## **SYSTEM ANALYSIS**

System analysis is a detailed study of the various operations performed by a system and their relationships within and outside the system. The goal of system analysis is to produce the system requirements of the proposed information system. There is doubt that presently existing manual system can be improved tremendously by changing certain procedures and implying manpower. But from the study of present manual system it is found that there is a large amount of redundancy i.e. some of the information has to be recorded and checked repeatedly which lead to wastage of time and increase the possibility of committing errors. Considering all the aspects, the introduction of computers is done to overcome the difficulties and to provide complete justification to computerize the existing system.

### **PROBLEM UNDERSTANDING**

There is not yet much use of the computers for any work. This has resulted in low efficiency, a lot of time wastage and data redundancy. Another major flaw in manual system is insecurity. There are no backups and data recovery solutions available.

Therefore main motive behind the new system was to minimize the problem faced due to old system. All the problems regarding old system were thoroughly seen. All the minus points were determined so that no problem could persist.

### **Statement of the Problem**

The development environment ensures that it has the portability and connectivity to run on virtually all standard hardware platforms, with stringent data security and easy recovery in case of a system failure.

It provides the benefits of streamlined operations, enhanced administration and control, improved response to patient care, cost control, and increased profitability.

We believe that every hospital is unique in terms of its requirements and priorities. Hence, flexibility has been built to allow easy customization.

## **REQUIRED SPECIFICATION**

SRS is a document that completely describes what the proposed system should do without describing how the system will do it. SRS helps the developer in knowing what the client expects from the computerized system.

### **Method used for data collection**

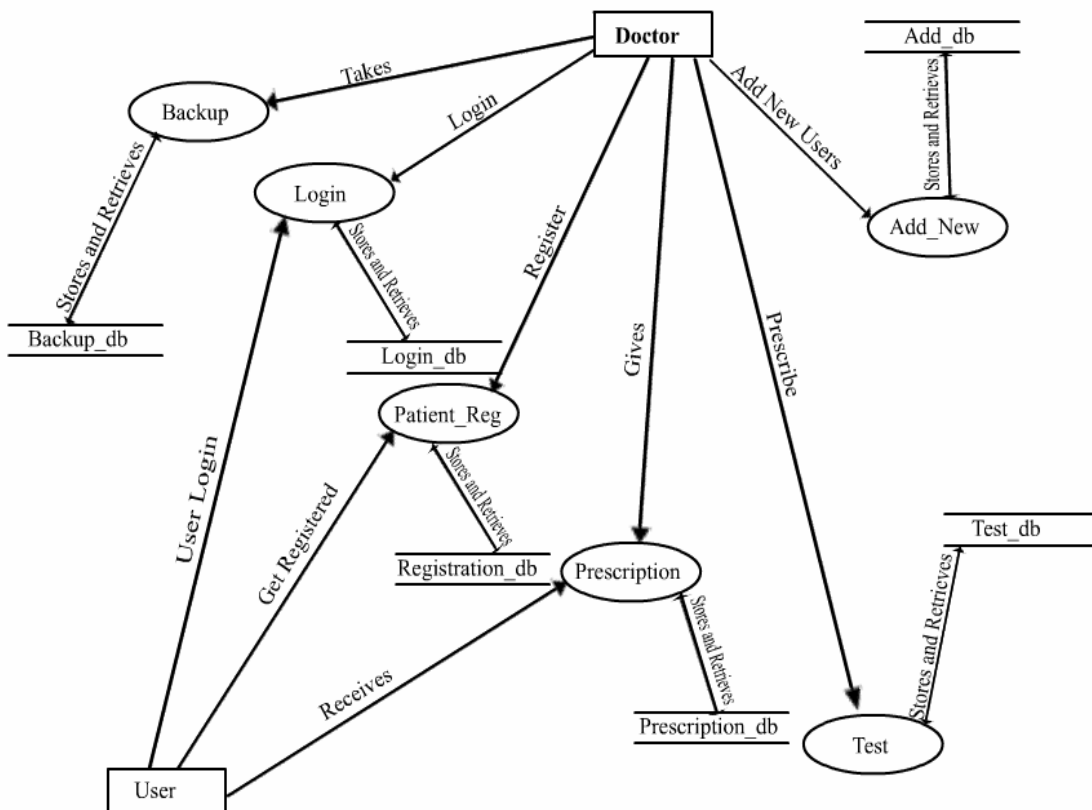
Key part of the system analysis is gathering information about the present system. A system not well defined in the analysis stage can never be correct. The main sources of information that provide accurate data about the existing system are as follows:

- ✓ User of the system
- ✓ Forms and documents used in the system.
- ✓ Procedure manuals and computer programs of the existing system.

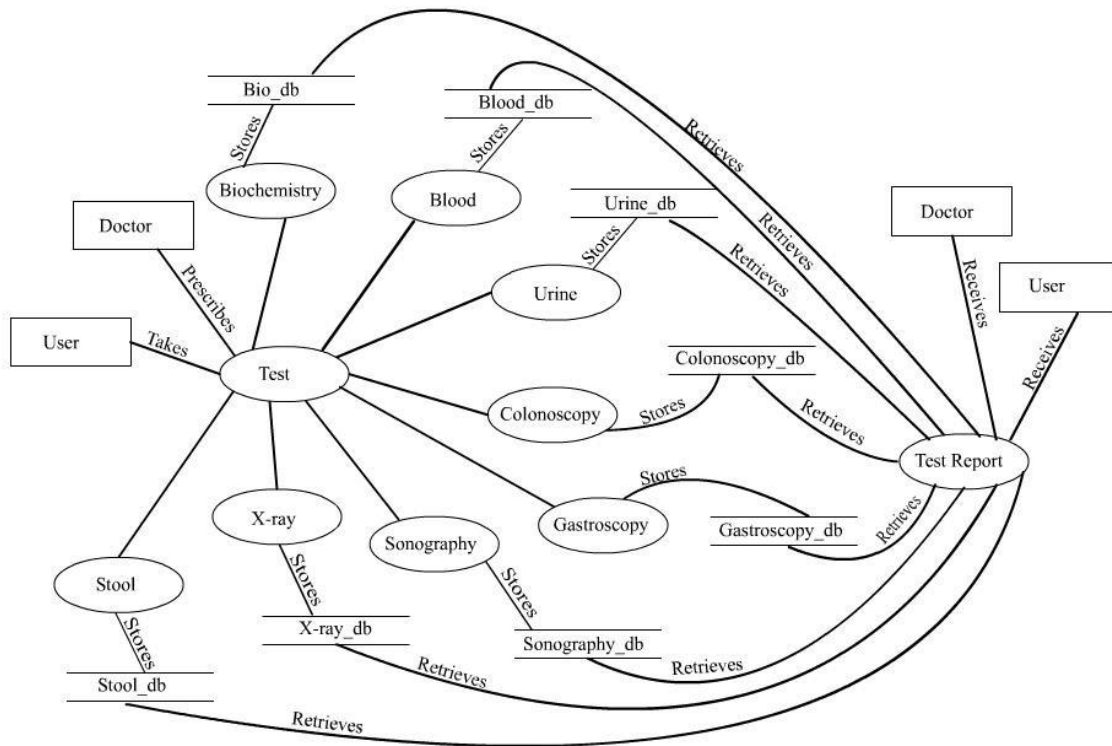
After examining the present system we found that the present system is slow as well as not very accurate. Our system will be very efficient in dealing with the day-to-day transactions Whole of the work will be computerized. Hence our system will make the work very easier.

## DATA FLOW DIAGRAMS

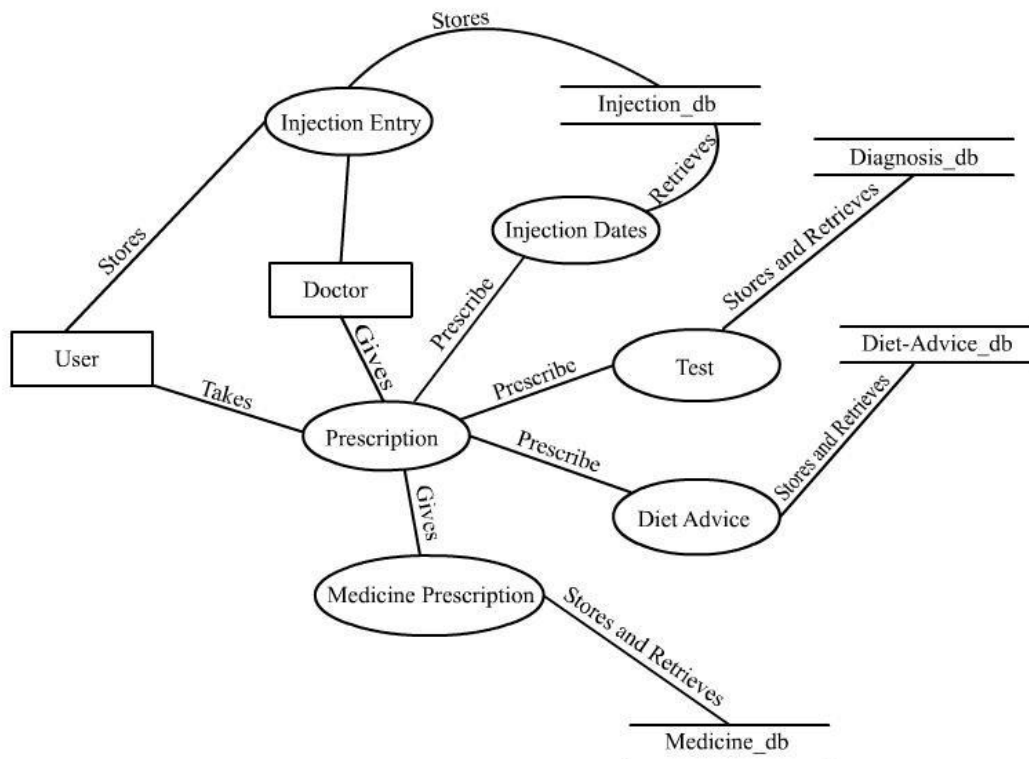
The DFDs given below illustrates and describes data flow, processing steps, source, sinks and data stores used to describe the overall functionality of the project. The following one describes the processing transactions that take place.



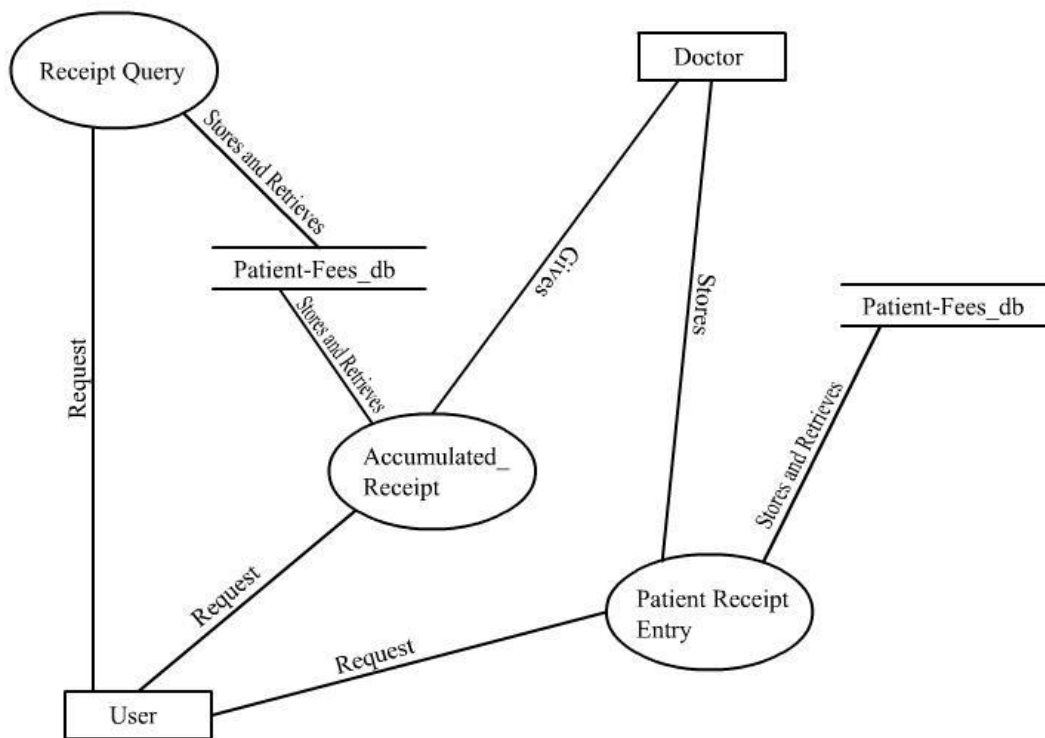
DFD LEVEL 0



DFD LEVEL 1



DFD LEVEL 2



DFD LEVEL 3

# **INTRODUCTION TO UML**

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

## **Goals of UML**

The primary goals in the design of the UML were:

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

## **Why Use UML?**

As the strategic value of software increases for many companies, the industry looks for techniques to automate the production of software and to improve quality and reduce cost and time-to-market. These techniques include component technology, visual programming, patterns and frameworks. Businesses also seek techniques to manage the complexity of systems as they increase in scope and scale. In particular, they recognize the need to solve recurring architectural problems, such as physical distribution, concurrency, replication, security, load balancing and fault tolerance. Additionally, the

development for the World Wide Web, while making some things simpler, has exacerbated these architectural problems. The Unified Modeling Language (UML) was designed to respond to these needs.

### **UML building blocks:-**

1. Things
2. Relationships
3. Diagrams

#### **1. Things:-**

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

- Structural Things
- Behavioral Things
- Grouping Things
- Annotational Things

- **Structural Things:**

The Structural things define the static part of the model. They represent physical and conceptual elements.

Following are the list of structural things:-

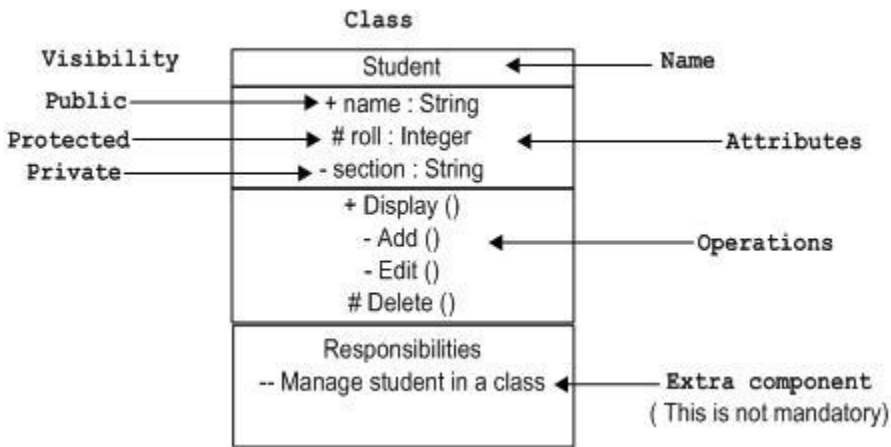
1. Class
2. Interface
3. Collaboration
4. Use case
5. Active classes
6. Components
7. Nodes

#### ***Class Notation:***

UML *class* is represented by the diagram shown below. The diagram is divided into four parts.

1. The top section is used to name the class.
2. The second one is used to show the attributes of the class.
3. The third section is used to describe the operations performed by the class.
4. The fourth section is optional to show any additional components.

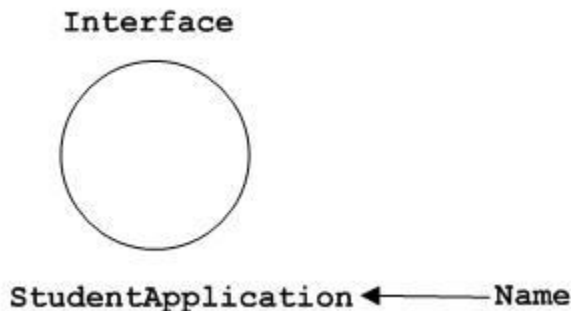




Classes are used to represent objects. Objects can be anything having properties and responsibility.

### ***Interface Notation:***

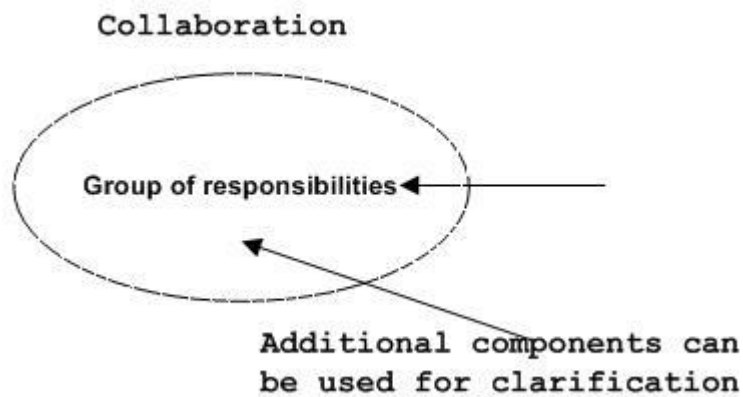
Interface is represented by a circle as shown below. It has a name which is generally written below the circle.



Interface is used to describe functionality without implementation. Interface is the just like a template where you define different functions not the implementation. When a class implements the interface it also implements the functionality as per the requirement.

### ***Collaboration Notation:***

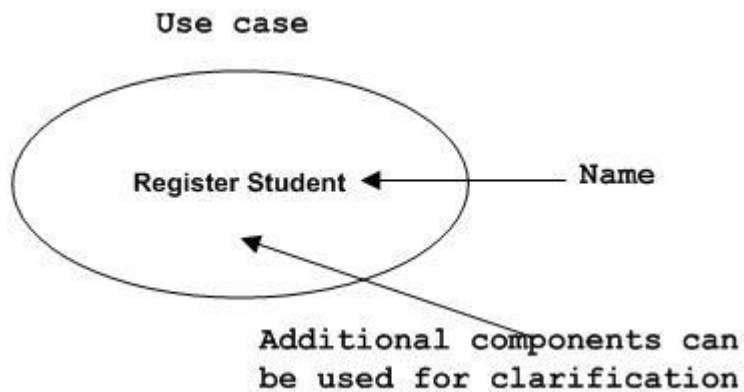
Collaboration is represented by a dotted ellipse as shown below. It has a name written inside the ellipse.



Collaboration represents responsibilities. Generally responsibilities are in a group.

***Use case Notation:***

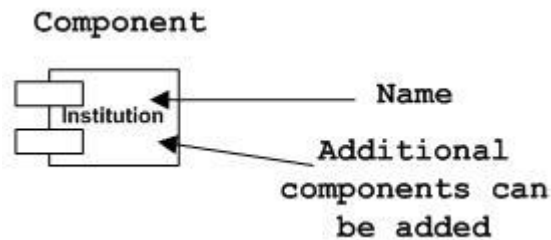
Use case is represented as an eclipse with a name inside it. It may contain additional responsibilities.



Use case is used to capture high level functionalities of a system.

***Component Notation:***

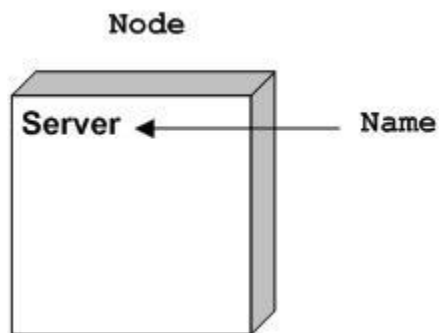
A component in UML is shown as below with a name inside. Additional elements can be added wherever required.



Component is used to represent any part of a system for which UML diagrams are made.

### ***Node Notation:***

A node in UML is represented by a square box as shown below with a name. A node represents a physical component of the system.



Node is used to represent physical part of a system like server, network etc.

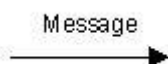
### • **Behavioral things:**

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

1. Interaction
2. State machine

### ***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.

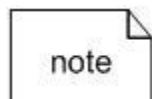


- **Annotational things:**

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.



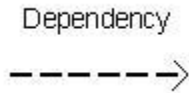
## 2. Relationships

A model is not complete unless the relationships between elements are described properly. The *Relationship* gives a proper meaning to an UML model. Following are the different types of relationships available in UML.

1. Dependency
2. Association
3. Generalization
4. Realization

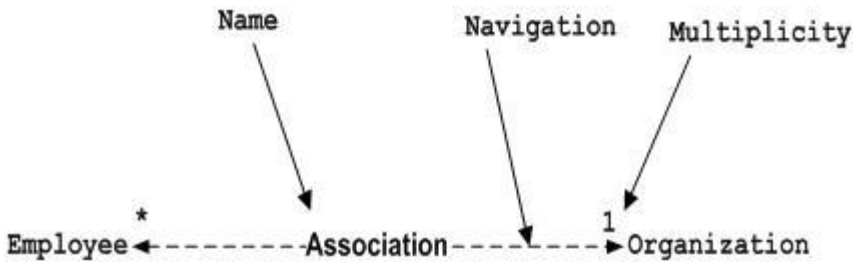
**Dependency:**

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



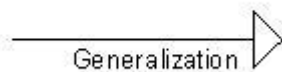
### **Association:**

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.



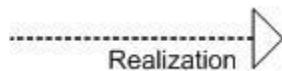
### **Generalization:**

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. UML Diagrams:**

UML diagrams are the ultimate output of the entire discussion. All the elements, relationships are used to make a complete UML diagram and the diagram represents a system.

The visual effect of the UML diagram is the most important part of the entire process. All the other elements are used to make it a complete one.

There are two broad categories of diagrams and then are again divided into sub-categories:

1. Structural Diagrams
2. Behavioral Diagrams

## 1. Structural Diagrams:

The *structural diagrams* represent the static aspect of the system. These static aspects represent those parts of a diagram which forms the main structure and therefore stable.

These static parts are represented by classes, interfaces, objects, components and nodes. The four structural diagrams are:

1. Class diagram
2. Object diagram
3. Component diagram
4. Deployment diagram

### ***Class Diagram:***

Class diagrams are the most common diagrams used in UML. Class diagram consists of classes, interfaces, associations and collaboration.

Class diagrams basically represent the object oriented view of a system which is static in nature.

Active class is used in a class diagram to represent the concurrency of the system.

Class diagram represents the object orientation of a system. So it is generally used for development purpose. This is the most widely used diagram at the time of system construction.

### ***Object Diagram:***

Object diagrams can be described as an instance of class diagram. So these diagrams are more close to real life scenarios where we implement a system.

Object diagrams are a set of objects and their relationships just like class diagrams and also represent the static view of the system.

The usage of object diagrams is similar to class diagrams but they are used to build prototype of a system from practical perspective.

### ***Component Diagram:***

Component diagrams represent a set of components and their relationships. These components consist of classes, interfaces or collaborations.

So Component diagrams represent the implementation view of a system.

During design phase software artifacts (classes, interfaces etc) of a system are arranged in different groups depending upon their relationship. Now these groups are known as components.

Finally, component diagrams are used to visualize the implementation.

### ***Deployment Diagram:***

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

Deployment diagrams are used for visualizing deployment view of a system. This is generally used by the deployment team.

## **2. Behavioral Diagrams:**

Any system can have two aspects, static and dynamic. So a model is considered as complete when both the aspects are covered fully.

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

UML has the following five types of behavioral diagrams:

1. Use case diagram
2. Sequence diagram
3. Collaboration diagram
4. Statechart diagram
5. Activity diagram

### ***Use case Diagram:***

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

A use case represents a particular functionality of a system.

So use case diagram is used to describe the relationships among the functionalities and their internal/external controllers. These controllers are known as actors.

### ***Sequence Diagram:***

A sequence diagram is an interaction diagram. From the name it is clear that the diagram deals with some sequences, which are the sequence of messages flowing from one object to another.

Interaction among the components of a system is very important from implementation and execution perspective.

So Sequence diagram is used to visualize the sequence of calls in a system to perform a specific functionality.

### ***Collaboration Diagram:***

Collaboration diagram is another form of interaction diagram. It represents the structural organization of a system and the messages sent/received. Structural organization consists of objects and links.

The purpose of collaboration diagram is similar to sequence diagram. But the specific purpose of collaboration diagram is to visualize the organization of objects and their interaction.

### ***Statechart Diagram:***

Any real time system is expected to be reacted by some kind of internal/external events. These events are responsible for state change of the system.

Statechart diagram is used to represent the event driven state change of a system. It basically describes the state change of a class, interface etc.

State chart diagram is used to visualize the reaction of a system by internal/external factors.

### ***Activity Diagram:***

Activity diagram describes the flow of control in a system. So it consists of activities and links. The flow can be sequential, concurrent or branched.

Activities are nothing but the functions of a system. Numbers of activity diagrams are prepared to capture the entire flow in a system.

Activity diagrams are used to visualize the flow of controls in a system. This is prepared to have an idea of how the system will work when executed.

### **Rules to connect the building blocks:-**

Specify what a well-formed model should look like.

The UML has semantic rules for

1. Names
2. Scope
3. Visibility
4. Integrity
5. Execution

### **Common Mechanisms:-**

Mechanisms/elements that apply consistently throughout the language:



1. Specifications
2. Adornments
3. Common Divisions
4. Extensibility Mechanisms

### **Specifications:-**

The UML is more than just a graphical language. Rather, behind every part of its graphical notation there is a specification that provides a textual statement of the syntax & semantics of that building block.

### **Adornments:-**

“Adorn” the model – i.e., enhance the model. Adds to the meaning and/or semantics of the element to which it pertains.

“Notes” are the mechanism provided by UML for adorning a model

### **Common Division:-**

In modeling object-oriented systems, the world often gets divided in at least a couple of ways.

There is the division of class & object. A class is an abstraction. An object is one concrete manifestation of that abstraction.

### **Extensibility Mechanisms:-**

The UML extensibility mechanisms include:-

1. Stereotypes
2. Tagged values
3. Constraints

### ***Stereotypes:***

1. Used to create new building blocks from existing blocks.
2. New building blocks are domain-specific.
3. A particular abstraction is marked as a “stereotype” and this stereotype is then used at other places in the model to denote the associated abstraction.

### ***Tagged Values:***

1. Used to add to the information of the element (not of its instances).

2. Stereotypes help create new building blocks; tagged values help create new attributes.
3. Commonly used to specify information relevant to code generation, configuration management, etc.

### **Constraints:**

A Constraint extends the semantics of a UML building block, allowing you to add new rules or modify existing ones.

### **UML Architecture**

Any real world system is used by different users. The users can be developers, testers, business people, analysts and many more. So before designing a system the architecture is made with different perspectives in mind. The most important part is to visualize the system from different viewer's perspective. The better we understand the better we make the system.

UML plays an important role in defining different perspectives of a system. These perspectives are:

1. Design
2. Implementation
3. Process
4. Deployment

And the centre is the **Use Case** view which connects all these four. A **Use case** represents the functionality of the system. So the other perspectives are connected with use case.

**Design** of a system consists of classes, interfaces and collaboration. UML provides class diagram, object diagram to support this.

**Implementation** defines the components assembled together to make a complete physical system. UML component diagram is used to support implementation perspective.

**Process** defines the flow of the system. So the same elements as used in *Design* are also used to support this perspective.

**Deployment** represents the physical nodes of the system that forms the hardware. UML deployment diagram is used to support this perspective.

### **Views of UML Diagrams:-**

- **Design View:** The design view of a system is the structural view of the system. This gives an idea of what a given system is made up of. Class diagrams and object diagrams form the design view of the system.

- **Process View:** The dynamic behavior of a system can be seen using the process view. The different diagrams such as the state diagram, activity diagram, sequence diagram, and collaboration diagram are used in this view.
- **Implementation View:** Next, you have the component view that shows the grouped modules of a given system modeled using the component diagram.
- **Deployment View:** The deployment diagram of UML is used to identify the deployment modules for a given system. This is the deployment view of the
- **Use case View:** Finally, we have the use case view. Use case diagrams of UML are used to view a system from this perspective as a set of discrete activities or transactions.

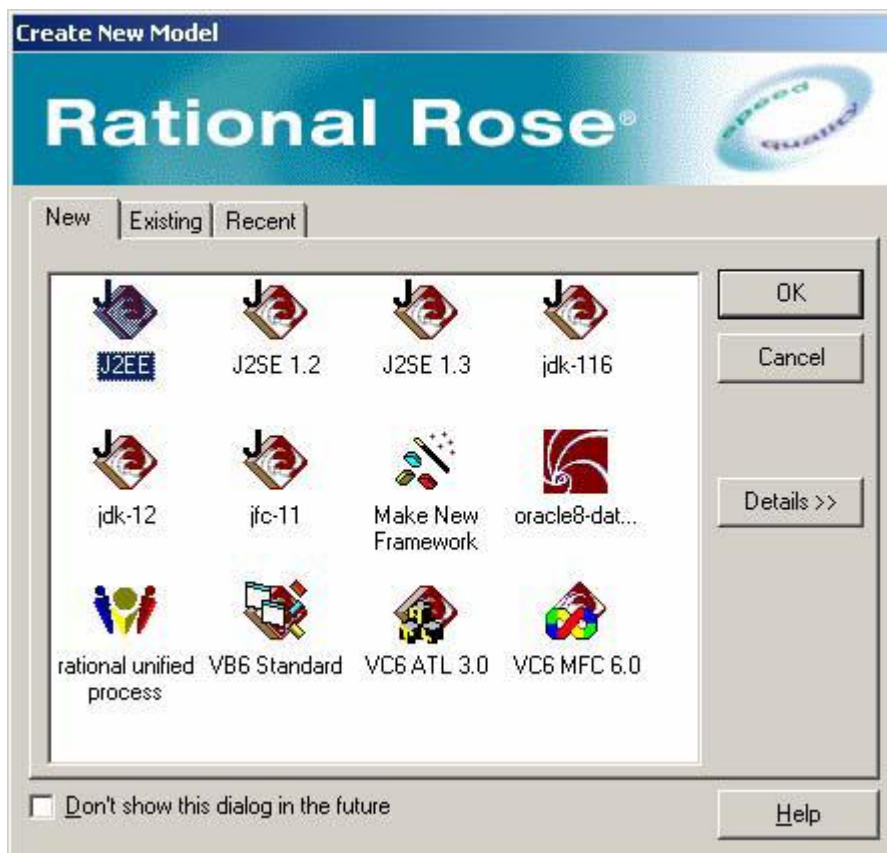
## INTRODUCTION TO RATIONAL ROSE

Rational Rose is a powerful visual modeling tool to aid in the analysis and design of object-oriented software systems. It is used to model your system before you write any code, so you can be sure that the system is architecturally sound from the beginning. It helps with system analysis by enabling you to design use cases and Use Case diagrams to show the system functionalities. It will let you design Interaction diagram to show how the objects work together to provide the needed functions.

Class diagrams can be created to show the classes in a system and how they relate to each other. Component diagrams can be developed to illustrate how the classes map to implementation components. Finally, Deployment diagrams can be produced to show the network design for the system.

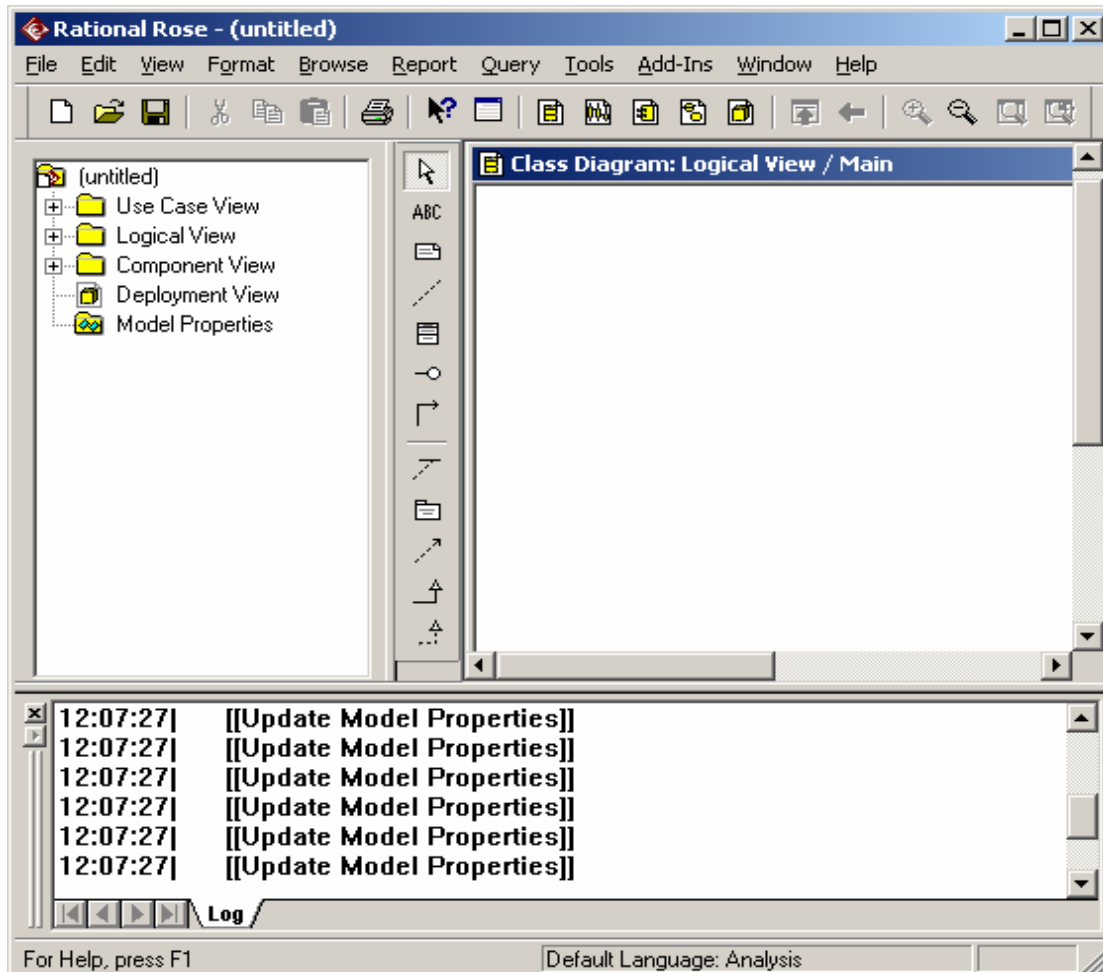
### Starting Rational Rose

When Rational Rose starts up, the following screen is displayed.



**Creating a project Model in Rational Rose :**

1. Start up Rational Rose Enterprise Edition.
2. Create a new model using the Rational Unified Process icon.
3. The window you will see will look something like this:



## Rational ROSE INTERFACE

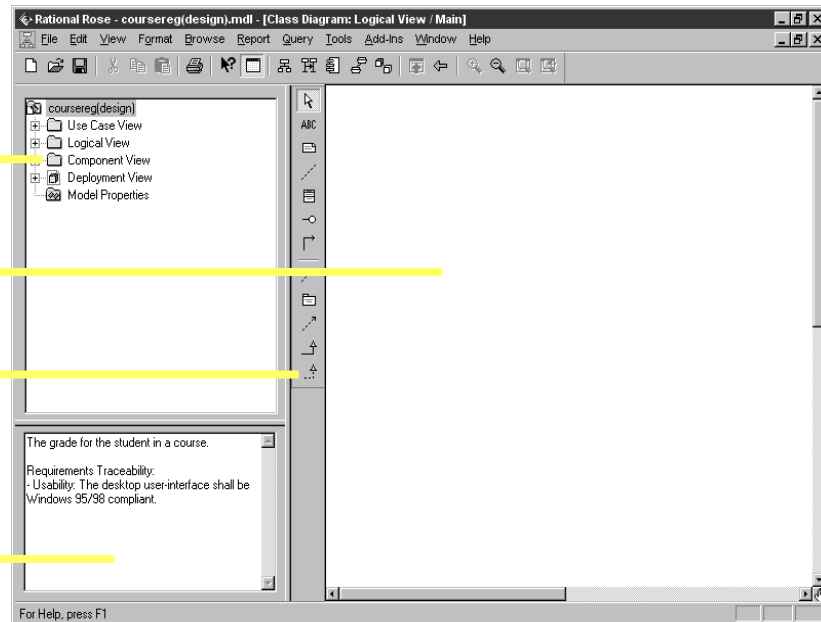
Browse

Diagram  
window

Diagram

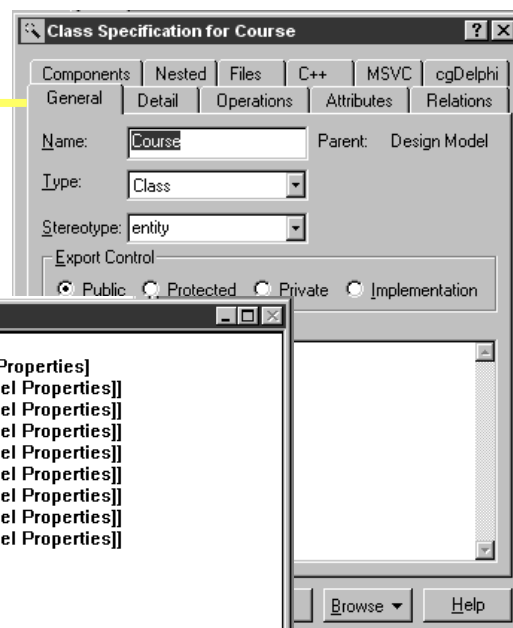
Documentation

Window



Specification

Log



**There are four views in Browser:**

- Use Case view
- Logical view
- Component view
- Deployment view

## Use Case view

The Use Case view is an implementation-independent look at the system. It focuses on a high-level picture of what the system will do, without worrying about the details of how the system will do.

Use Case view includes:

Actors

Use cases

Associations

Use case documentation

Use Case diagrams










Sequence diagrams

Collaboration diagrams

Packages

### To create a new Use Case diagram:











1. Right-click a package in the Use Case view.
2. Select New Use Case Diagram.

	Text Box	Adds a text box to the diagram.
	Note	Adds a note to the diagram.
	Anchor Note to Item	Connects a note to a use case or actor on the diagram.
	Package	Adds a new package to the diagram.
	Use Case	Adds a new use case to the diagram.
	Actor	Adds a new actor to the diagram.
	Unidirectional Association	Draws a relationship between an actor and a use case.
	Dependency or Instantiates	Draws a dependency between items on the diagram.
	Generalization	Draws a includes or an extends relationship between use cases, or draws an inheritance relationship between actors.

### To create a new Sequence diagram:













1. Right-click a use case in the Use Case view.
2. Select New -> Sequence Diagram.

### Tool Box of sequence diagram

	Text Box	Adds a text box to the diagram.
	Note	Adds a note to the diagram.
	Anchor Note to Item	Connects a note to an item in the diagram.
	Object	Adds a new object to the diagram.
	Object Message	Draws a message between two objects.
	Message to Self	Draws a reflexive message.
	Return Message	Shows a return from a procedure call.
	Destruction Marker	Shows when an object is destroyed.
	Procedure Call	Draws a procedure call between two objects.
	Asynchronous Message	Draws an asynchronous message between two objects.

### To create a new Collaboration diagram:

1. Right-click a use case in the Use Case view.
2. Select New Collaboration Diagram.

Icon	Button	Purpose
	Selects or Deselects an Item	Returns the cursor to an arrow to select an item.
	Text Box	Adds a text box to the diagram.
	Note	Adds a note to the diagram.
	Anchor Note to Item	Connects a note to an item on the diagram.
	Object	Adds a new object to the diagram.
	Class Instance	Adds a new class instance to the diagram.
	Object Link	Creates a path for communication between two objects.
	Link to Self	Shows that an object can call its own operations.
	Link Message	Adds a message between two objects or from an object to itself.
	Reverse Link Message	Adds a message in the opposite direction between two objects or from an object to itself.
	Data Token	Shows information flow between two objects.
	Reverse Data Token	Shows information flow in the opposite direction between two objects.

### State Chart Diagrams

State chart (or state) diagrams describe the states and responses of a class. State chart diagrams describe the behavior of a class in response to external stimuli. These diagrams contain the following elements:

- States, which represent the situations during the life of an object in which it satisfies some condition, performs some activity, or waits for some occurrence.
- Transitions, which represent relationships between the different states of an object



## Activity Diagrams

Activity diagrams describe the activities of a class. These diagrams are similar to state chart diagrams and use similar conventions, but activity diagrams describe the behavior of a class in response to internal processing rather than external events as in state chart diagram.

- Swimlanes, which represent responsibilities of one or more objects for actions within an overall activity; that is, they divide the activity states into groups and assign these groups to objects that must perform the activities.
- Action States, which represent atomic, or noninterruptible, actions of entities or steps in the execution of an algorithm.
- Action flows, which represent relationships between the different action states of an entity
- Object flows, which represent the utilization of objects by action states and the influence of action states on objects.

### Logical view:

The Logical view focuses on how the system will implement the behavior in the use cases.

It provides a detailed picture of the pieces of the system, and describes how the pieces interrelate. With these detailed elements, developers can construct a detailed design for the system.

### Logical view includes:

- Classes
- Class diagrams
- Associations
- Interfaces
- Sequence diagrams
- Collaboration diagrams
- State chart diagrams
- Packages

### To create a new Class diagram:

1. Right-click a package in the Logical view.
2. Select New Class Diagram.

Icon	Button	Purpose
	Selects or Deselects an Item	Returns the cursor to an arrow to select an item.
	Text Box	Adds a text box to the diagram.
	Note	Adds a note to the diagram.
	Anchor Note to Item	Connects a note to an item on the diagram.
	Class	Adds a new class to the diagram.
	Interface	Adds a new interface class to the diagram.
	Association	Draws an association relationship.
	Aggregation	Draws an aggregation relationship.
	Association Class	Links an association class to an association relationship.
	Package	Adds a new package to the diagram.
	Dependency or Instantiates	Draws a dependency relationship.
	Generalization	Draws a generalization relationship.
	Realize	Draws a realizes relationship.
	Parameterized Class	Adds a new parameterized class to the diagram.
	Class Utility	Adds a new class utility to the diagram.
	Parameterized Class Utility	Adds a new parameterized class utility to the diagram.
	Instantiated Class	Adds a new instantiated class to the diagram.
	Instantiated Class Utility	Adds a new instantiated class utility to the diagram.
	Domain	Adds a new domain to the diagram.
	Domain Package	Adds a new domain package to the diagram.
	Server Page	Adds a new server page to the diagram.
	Client Page	Adds a new client page to the diagram.
	Form	Adds a new HTML form to the diagram.
	COM Object	Adds a new COM object to the diagram.
	Applet	Adds a new applet to the diagram.

## Component view

The Component view contains information about the code libraries, executable files, run-time libraries, and other components in your model.

The Component view of the system allows you to see the relationships between the modules of code.

### Component view includes:

- Components
- Interfaces
- Component diagrams
- Packages

## **Deployment view**

The Deployment view is concerned with the physical deployment of the system, which may differ from the logical architecture of the system. The Deployment view contains processors, devices, processes, and connections between processors and devices. All of this information is diagrammed on a Deployment diagram. There is only one Deployment diagram per system. For example, the system may have logical three-tier architecture. The interface may be separated from the business logic and the database logic. However, the deployment may be two-tiered. The interface may be placed on one machine, while the business and database logic are located on another machine.

Other issues, such as fault tolerance, network bandwidth, disaster recovery, and response time, are also handled using the Deployment view.

### **Deployment view includes:**

Processes  
Processors  
Connectors  
Devices  
Deployment diagram

### **To open the Deployment diagram:**

1. Double-click Deployment View entry in the browser.
2. Rose will open the Deployment diagram for the model.

## **USECASE DESCRIPTION**

**Use cases:** 1. Admissions

2. Doctor Appointments
3. Tests Appointments
4. Bed Allotment
5. Undergo Operation
6. Login
7. Draw Salary
8. Add Doctor/Staff
9. Delete Doctor/Staff
10. Edit Doctor/Staff
11. Prescribe Tests
12. Ward Wise Bed Status
13. Admission/Discharge Reports
14. Patient Information

**Actors:** 1. Receptionist

2. Doctors
3. Staff/Nurses
4. Income
5. Expenditure
6. Records System
7. Information System

**ADMISSIONS:**

This Module helps in registering information about patients and handles patient's query. A unique ID is generated for each patient after registration. This helps in implementing customer relationship management and also maintains medical history of the patient.

**DOCTOR APPOINTMENTS:**

This Module Deals with, when the ID is generated the patient receives the Appointment time & number from the Receptionist and accordingly visit the doctor.

**TESTS APPOINTMENTS:**

This Module Deals with, when the ID is generated the patient receives the Appointment time & number from the Receptionist and accordingly undergoes the tests.

**BED ALLOTMENT:**

This Module handles with allotting the Bed to various patients by checking their ID.

**UNDERGO OPERATION:**

This Module handling with undergoes the various operations by diagnosing the patients.

**LOGIN:**

This Module checks whether the person is a Doctor/Staff and handles various activities such as draw Salary and give Salary.

**DRAW SALARY:**

This Module checks whether the person is a Doctor/Staff and draws salary based on the information.

**ADD DOCTOR/STAFF:**

This Module handles the activities such as adding Doctor/Staff information into the database.

**DELETE DOCTOR/STAFF:**

This Module handles the activities such as deleting Doctor/Staff information into the database.

**EDIT DOCTOR/STAFF:**

This Module handles the activities such as editing Doctor/Staff information into the database.

**PRESCRIBE TESTS:**

This Module handles various activities such as Doctor Diagnoses the patient, gives treatment & gives suggestions to the patients, & prescribes laboratory tests & medicines.

**WARDWISE BED STATUS;**

This Module takes care of medical equipment, doctor visit, vitals recording, patient case sheet, diet ordering, blood requisition, transfer intimation and discharge intimation etc. It also deals with the maintenance of the wards, inter- and intra-ward transfers.

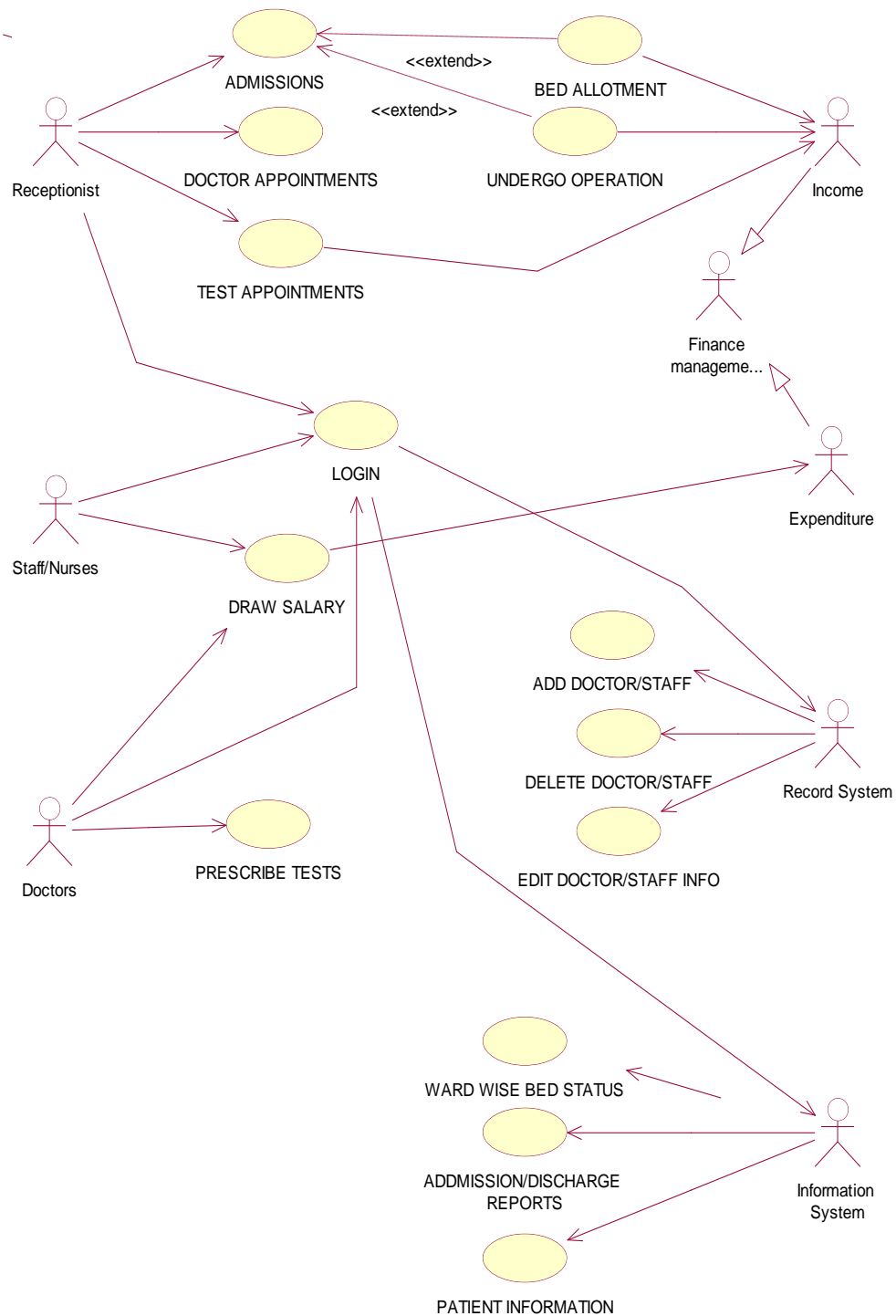
**ADMISSION/DISCHARGE REPORTS:**

This Module helps in generating patient's discharge summary, which includes patient's health at the time of discharge, medical history, various diagnosis and drug prescriptions, history of present illness and course in hospital.

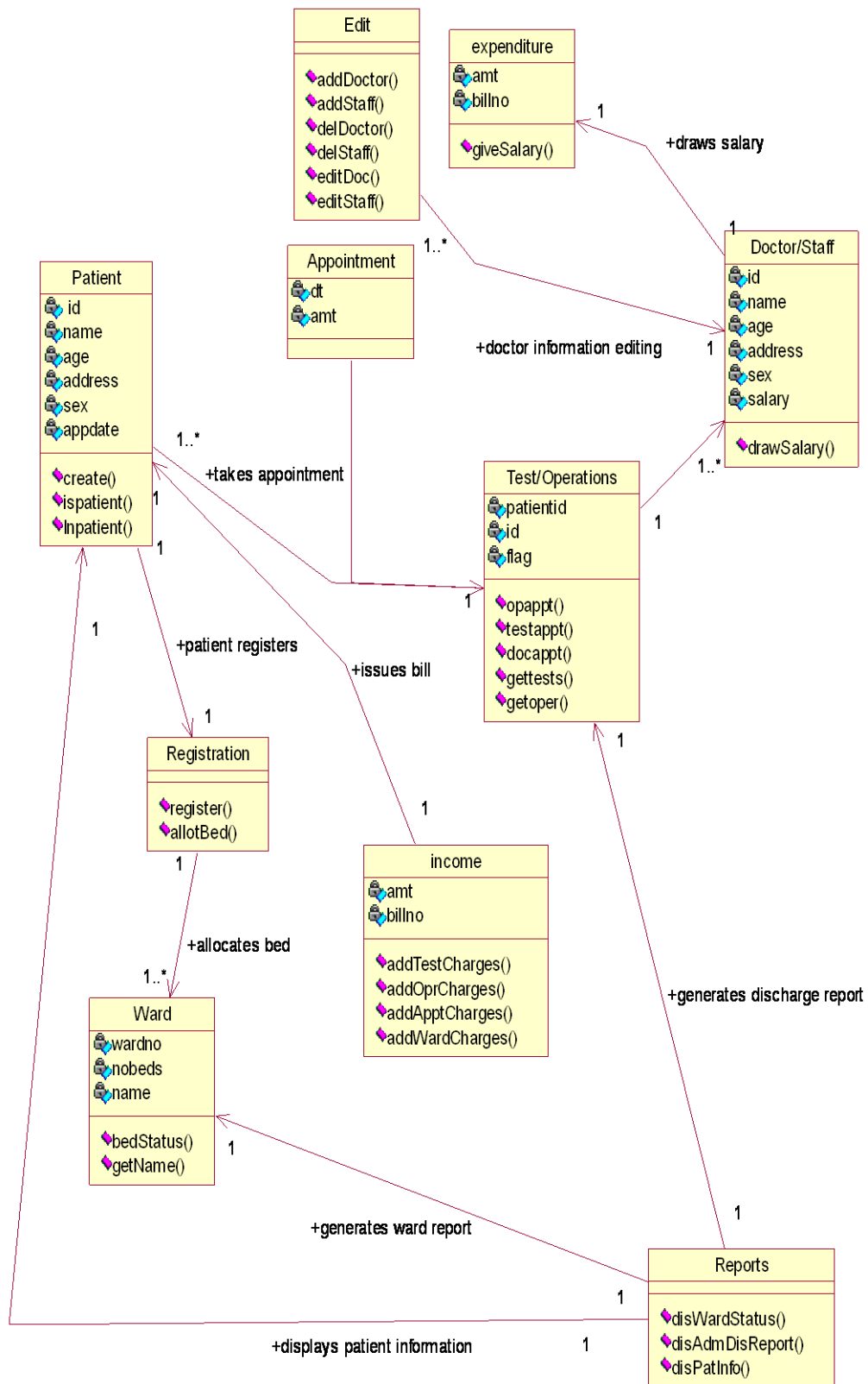
**PATIENT INFORMATION:**

This Module helps in generating the patient information which is provided by doctor.

# USE CASE DIAGRAM



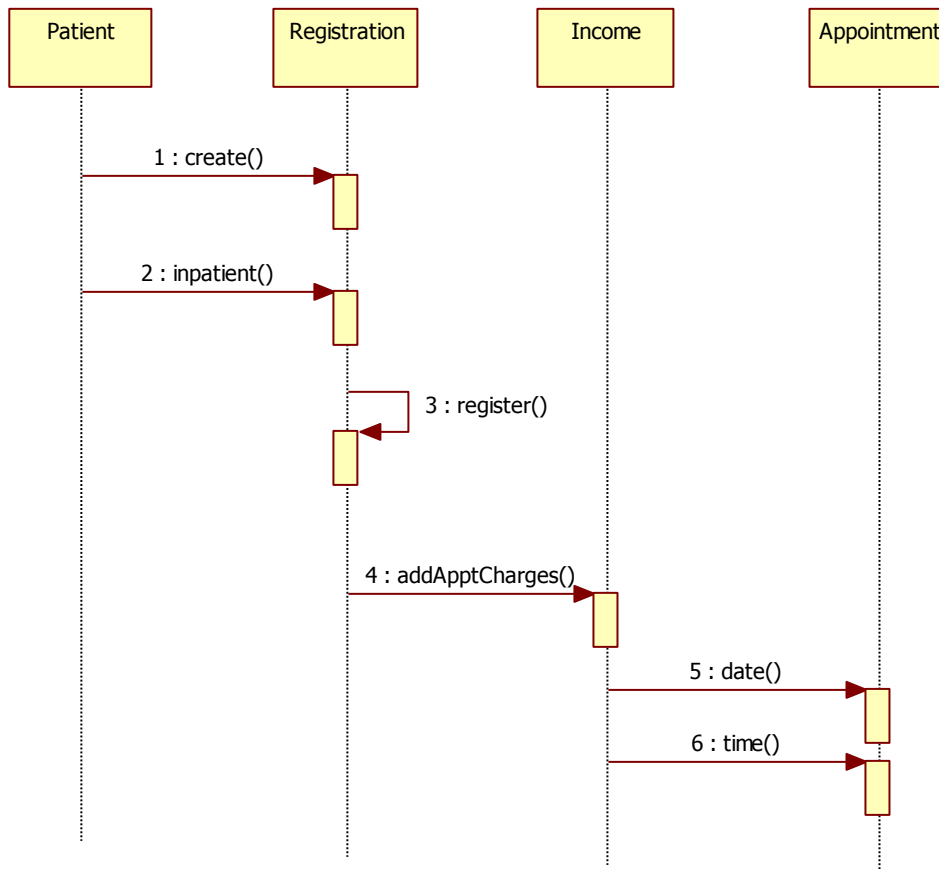
# CLASSDIAGRAM



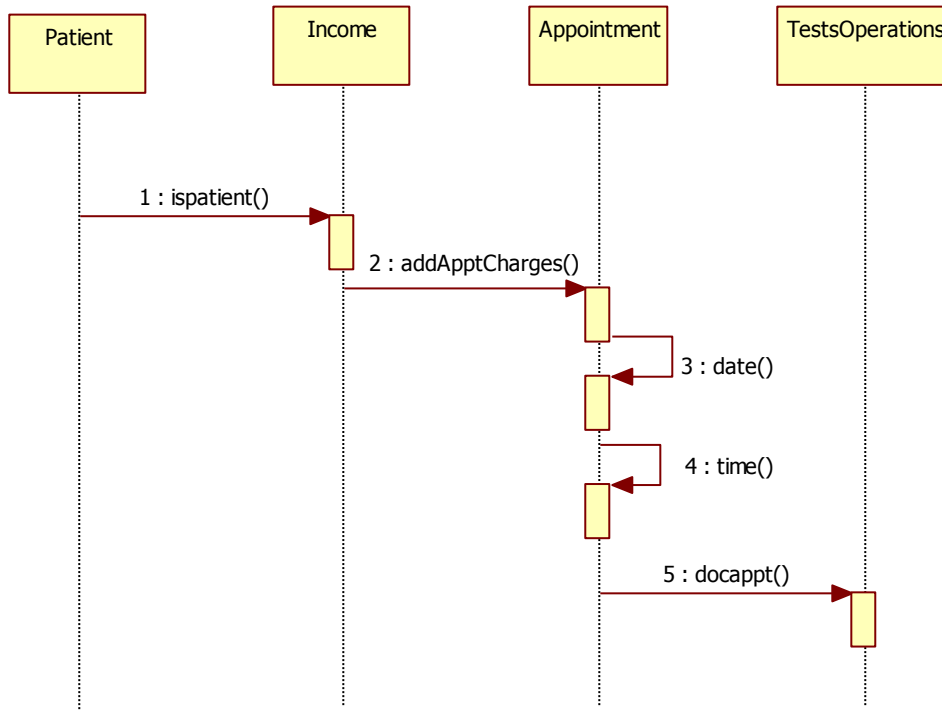


# SEQUENCE DIAGRAMS

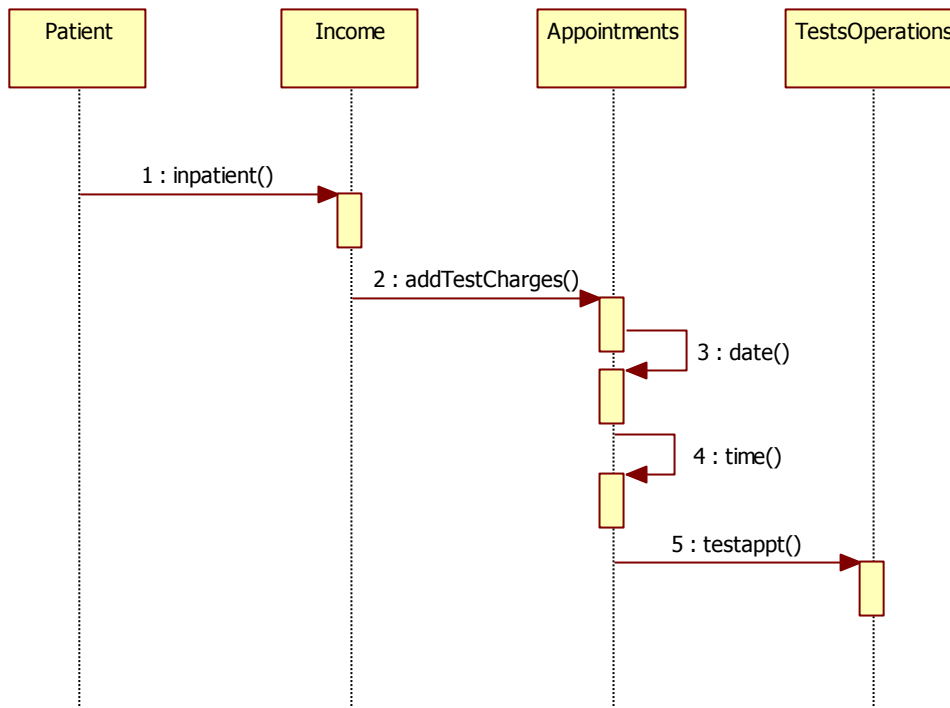
## ADMISSIONS:



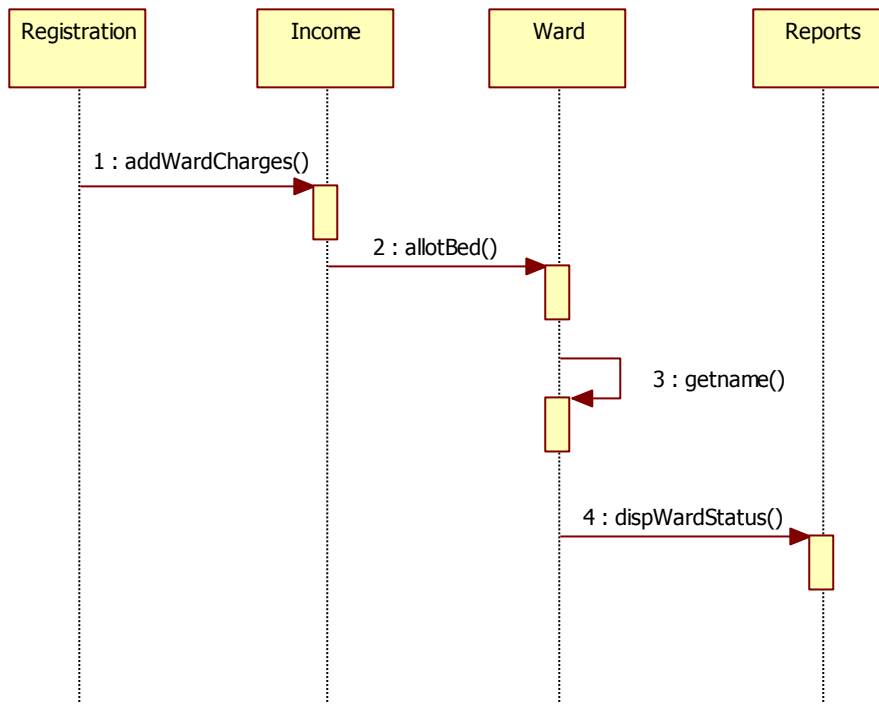
### DOCTOR APPOINTMENTS:



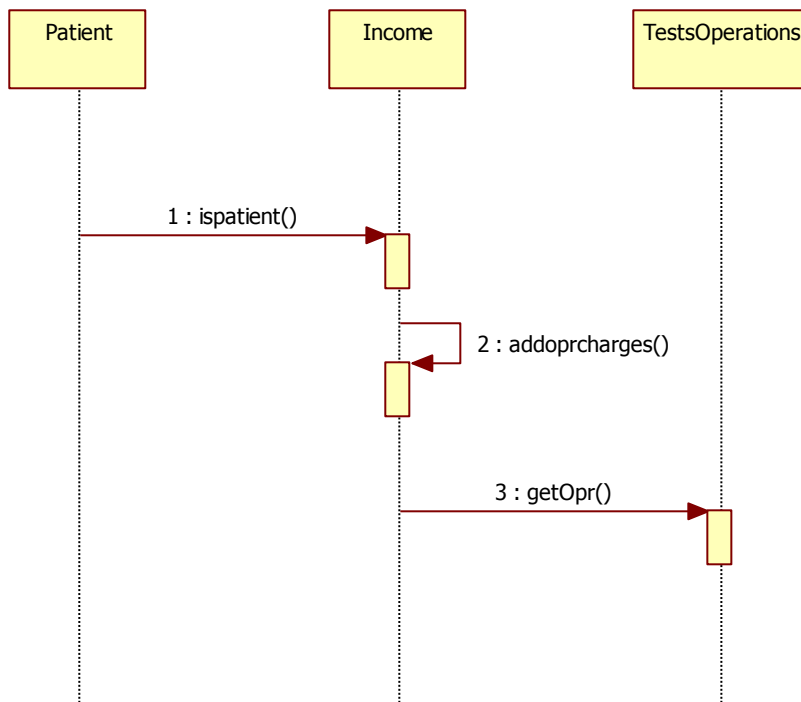
### TESTS APPOINTMENTS:



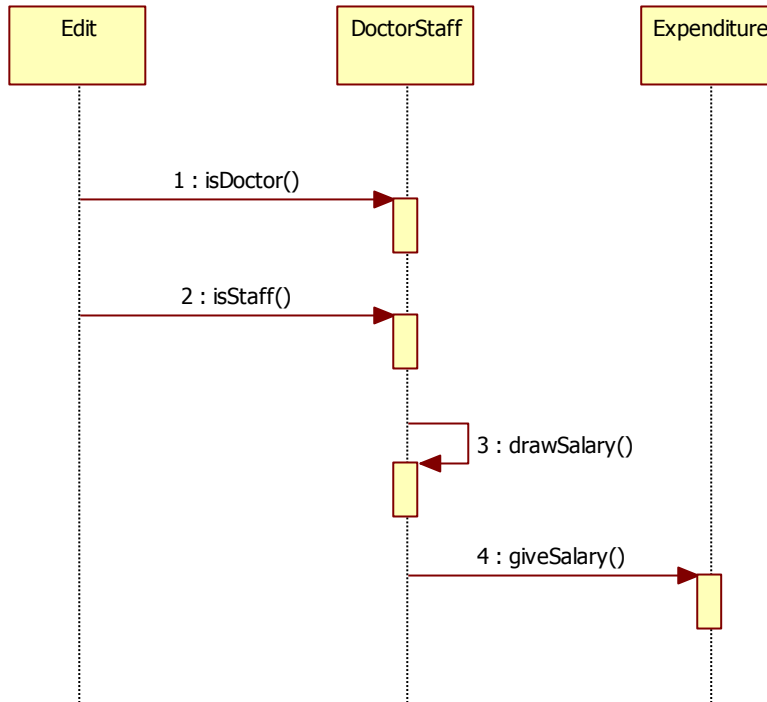
### BED ALLOTMENT:



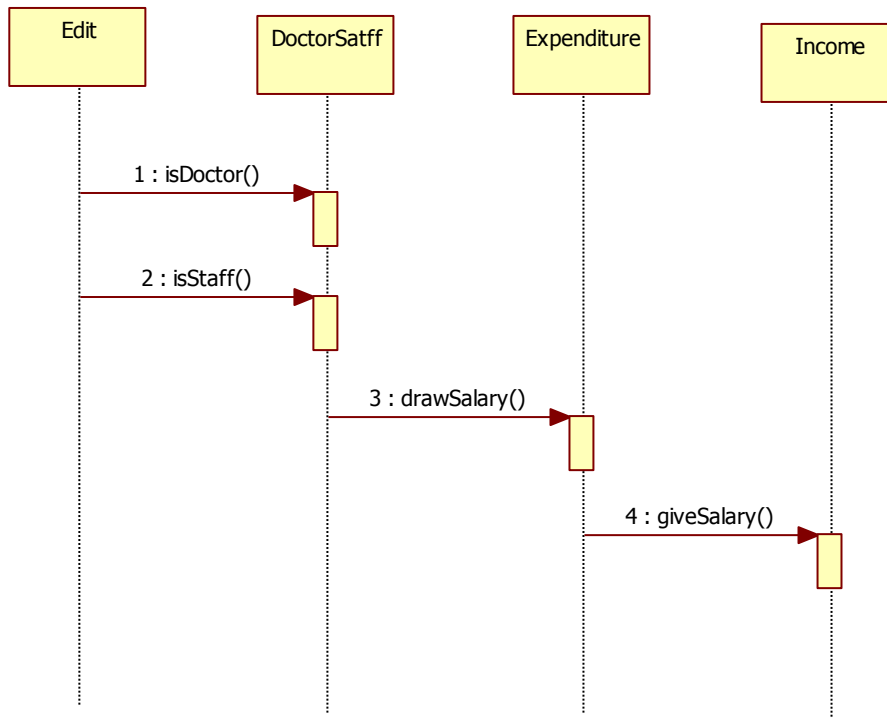
### UNDERGO OPERATION:



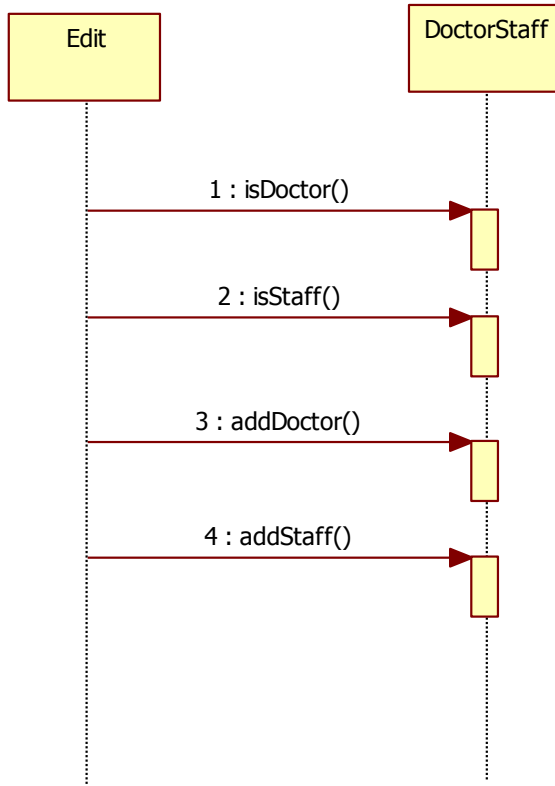
### LOGIN:



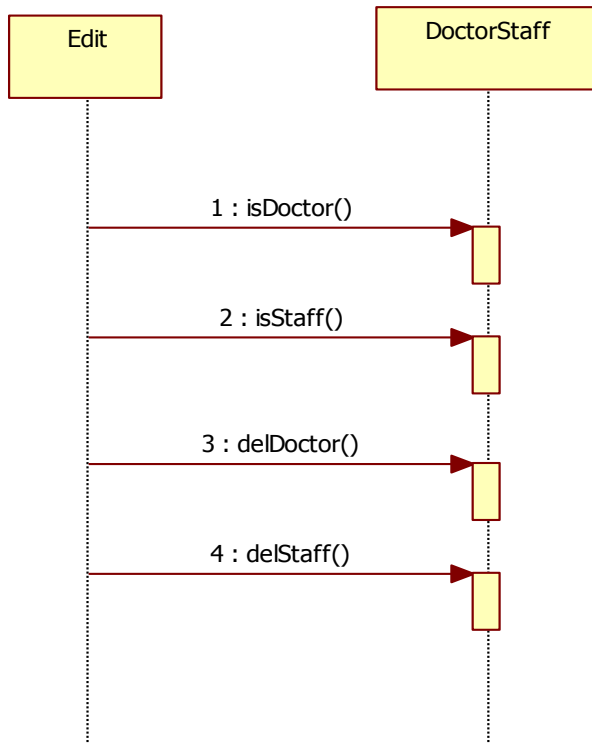
### DRAW SALARY:



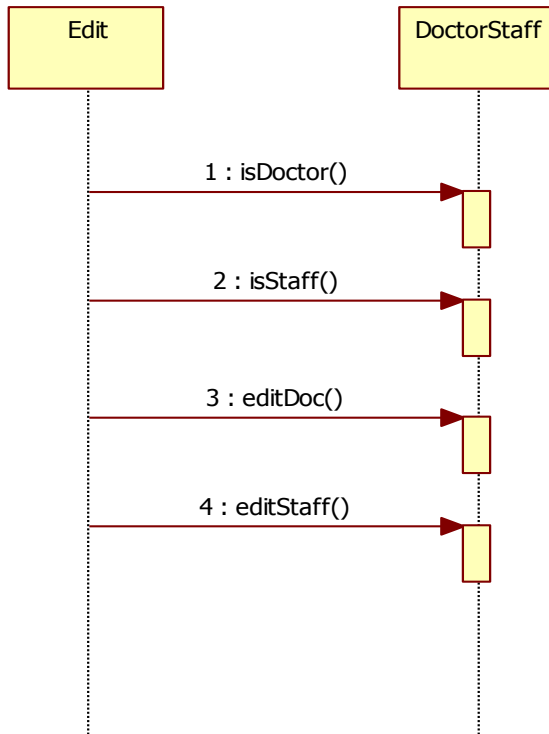
### ADD DOCTOR/STAFF:



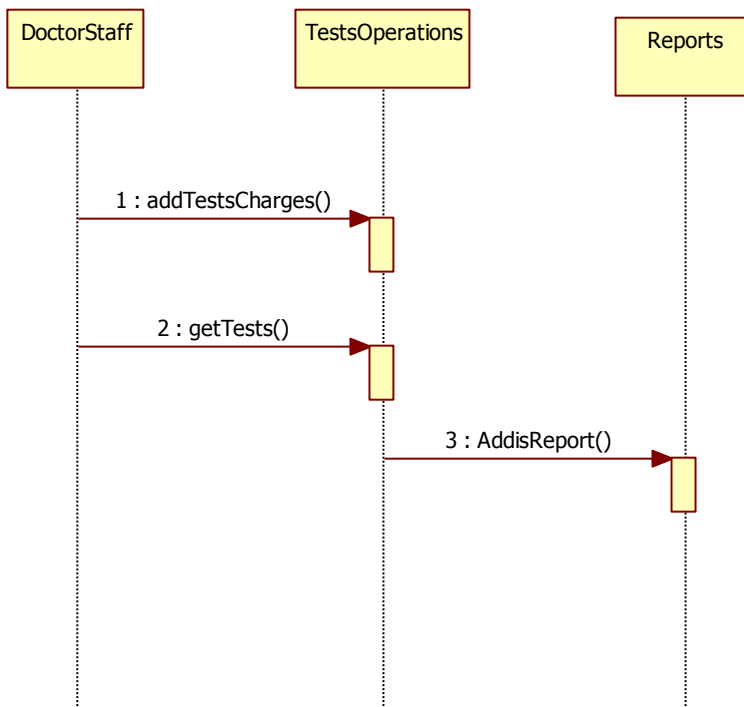
### DELETE DOCTOR/STAFF:



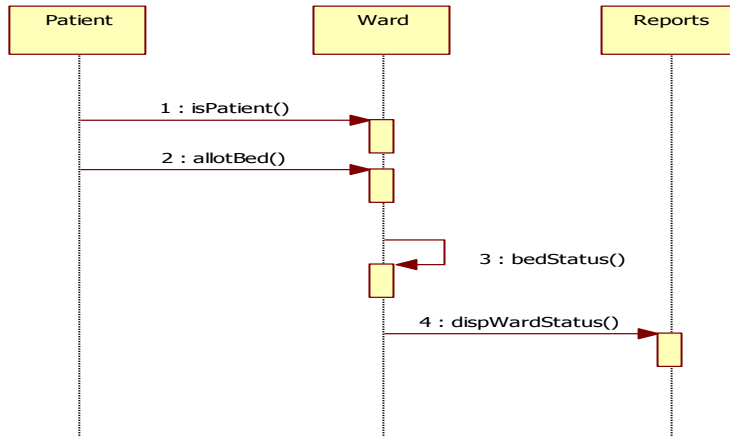
### EDIT DOCTOR/STAFF:



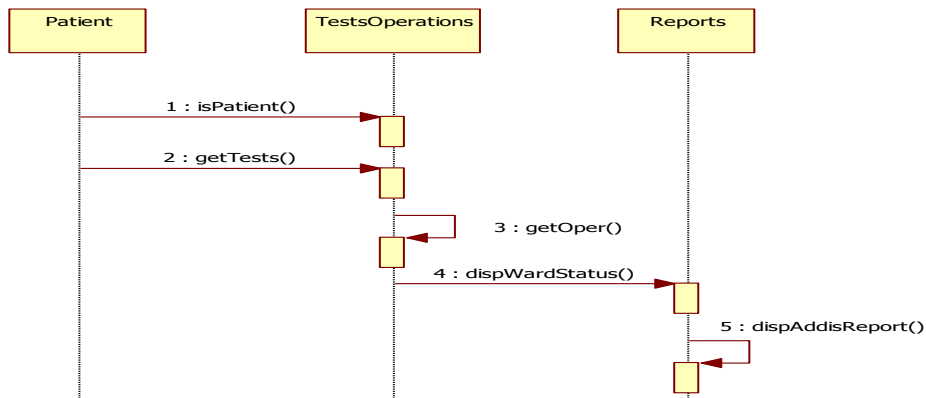
### PRESCRIBE TESTS:



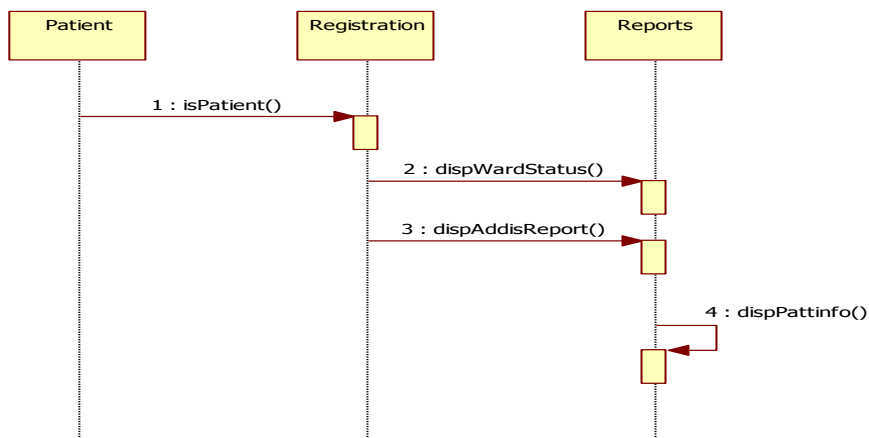
### WARDWISE BED STATUS:



### ADMISSION/DISCHARGE REPORTS:

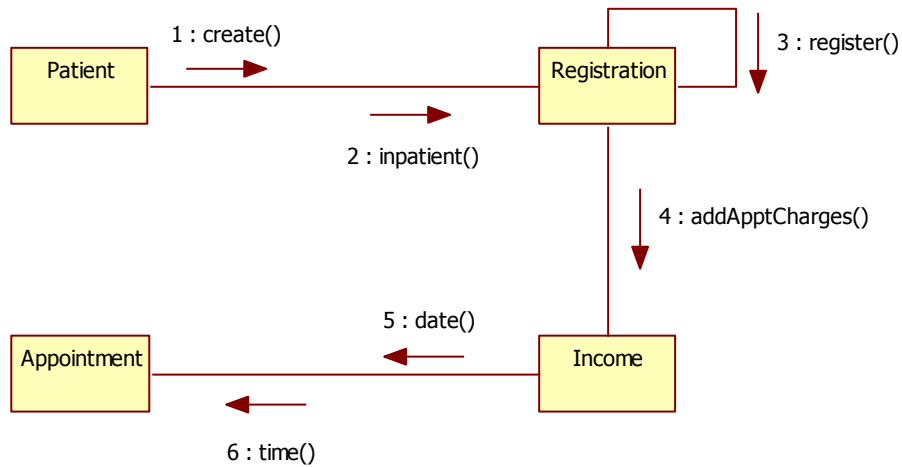


### PATIENT INFORMATION:

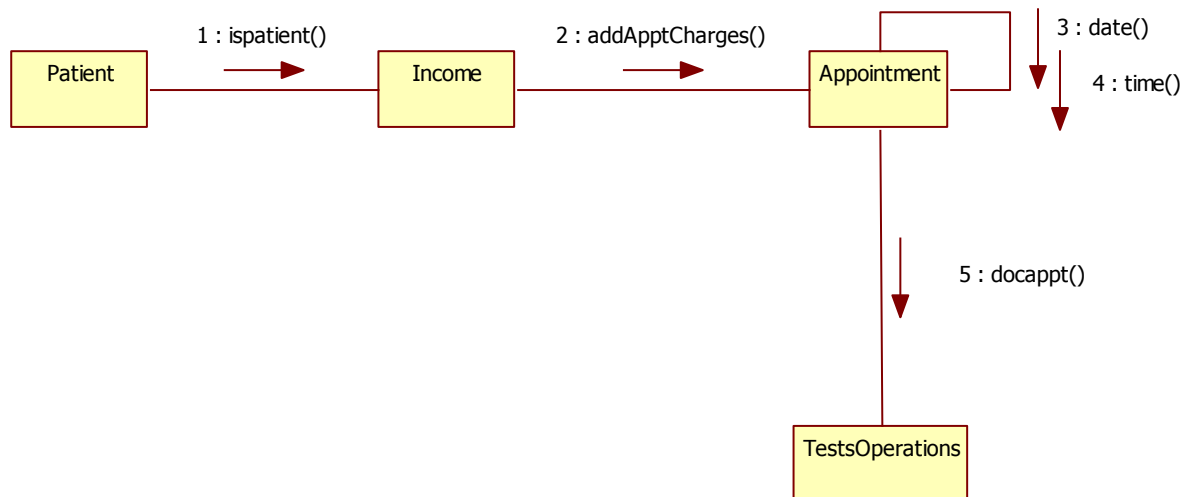


# COLLABORATION DIAGRAMS

## ADMISSIONS:

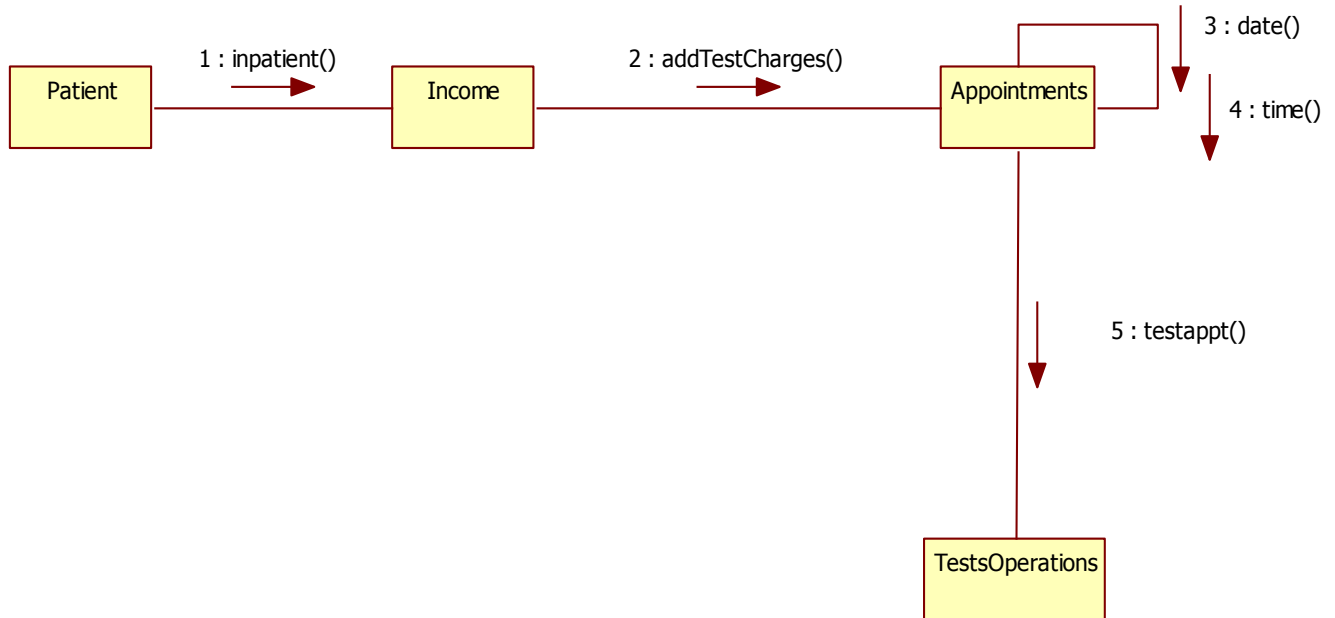


## DOCTOR APPOINTMENTS:

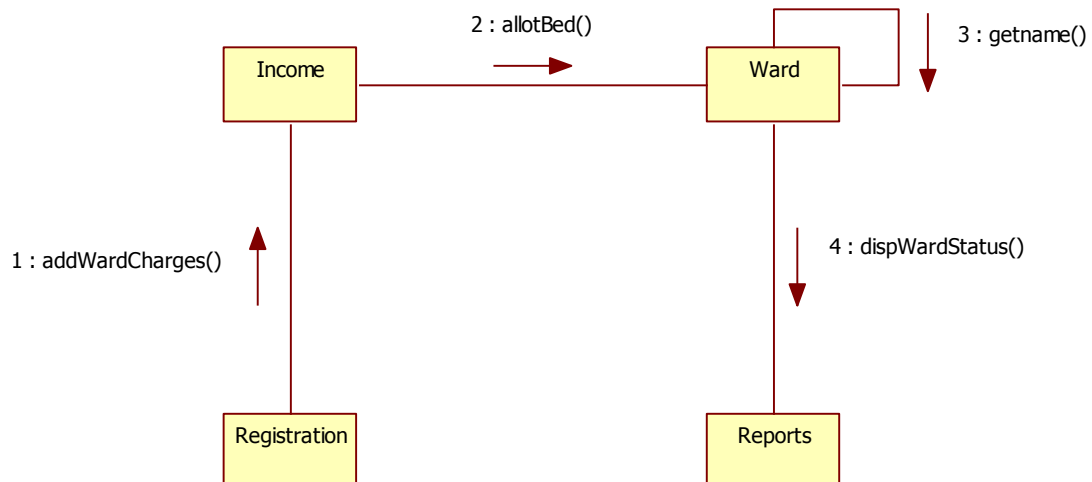




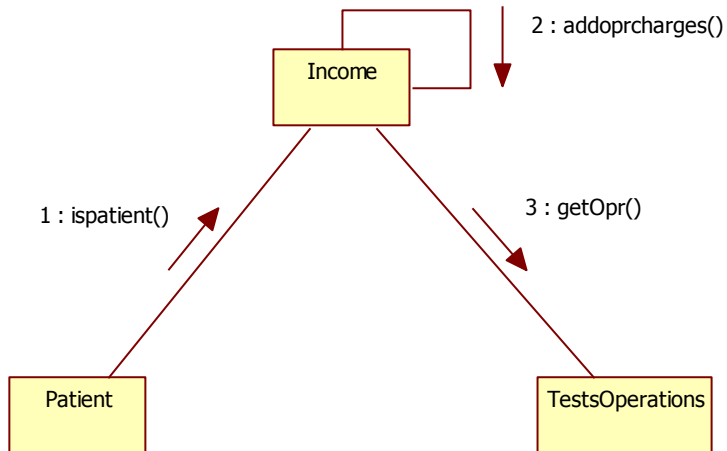
### TESTS APPOINTMENTS:



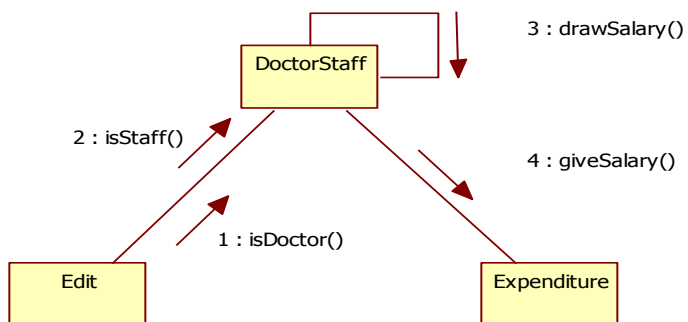
### BED ALLOTMENT:



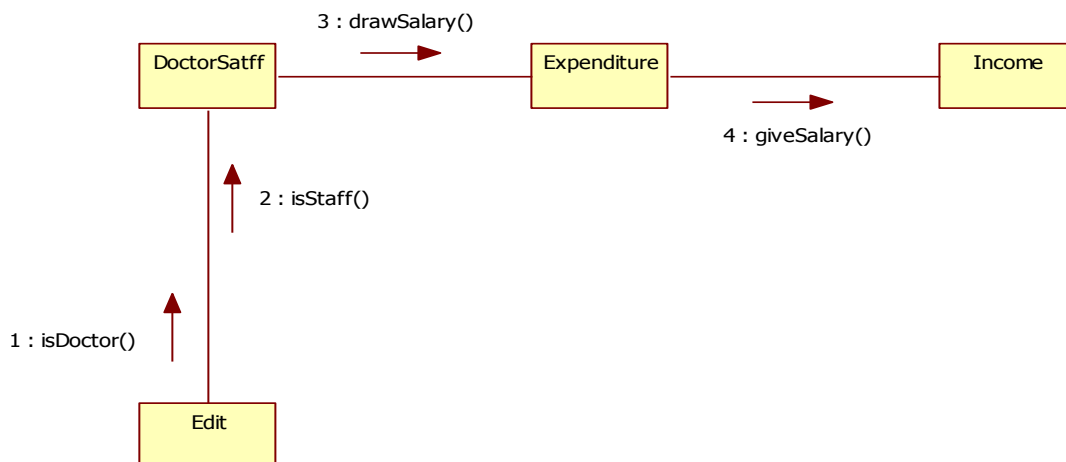
### UNDERGO OPERATION:



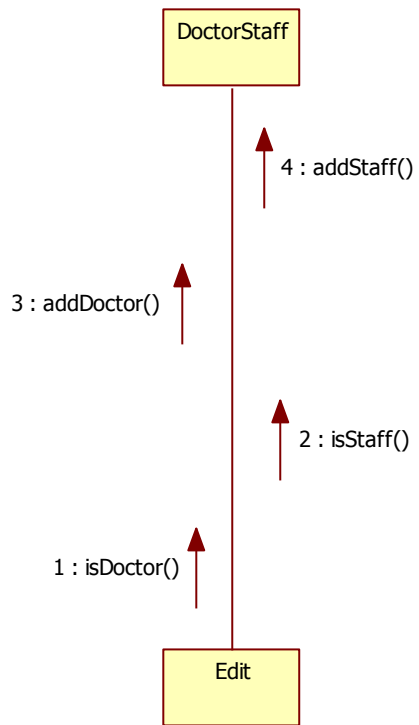
### LOGIN:



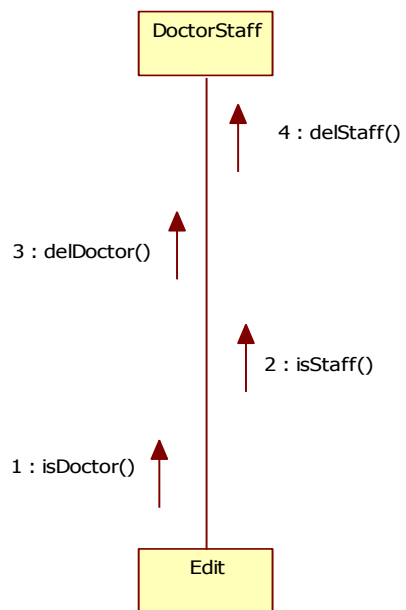
### DRAW SALARY:



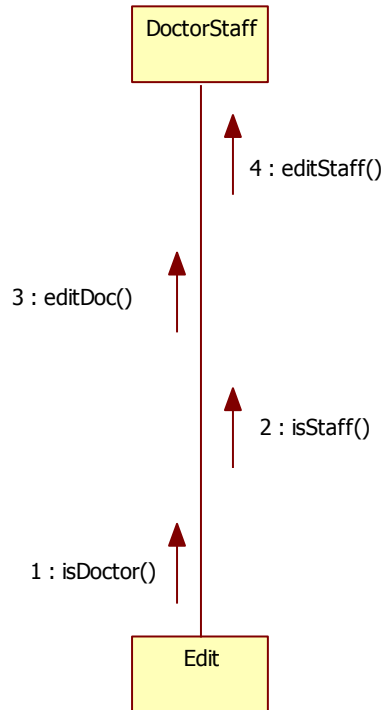
### ADD DOCTOR/STAFF:



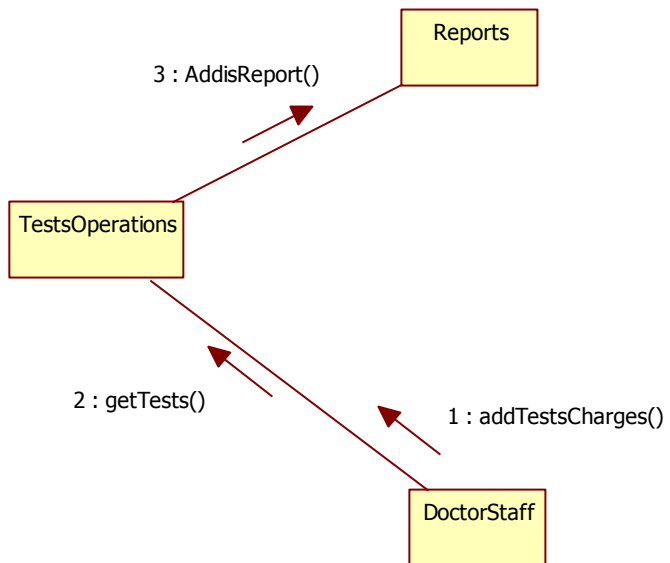
### DELETE DOCTOR/STAFF:



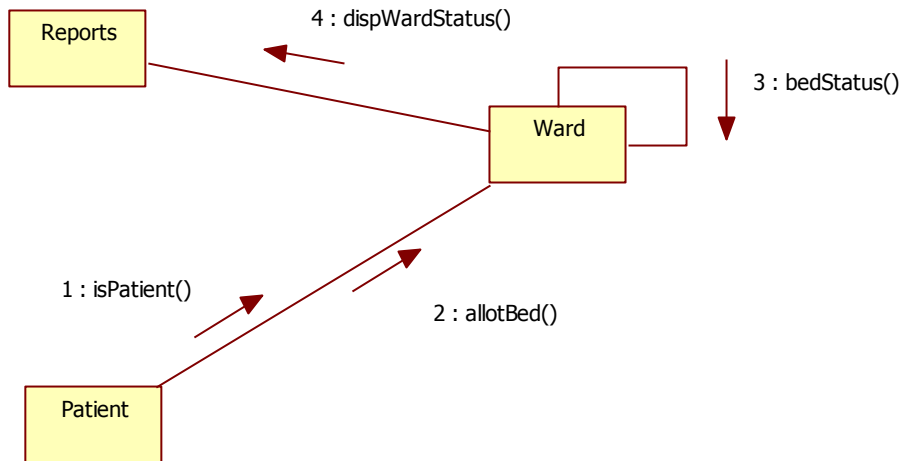
### EDIT DOCTOR/STAFF:



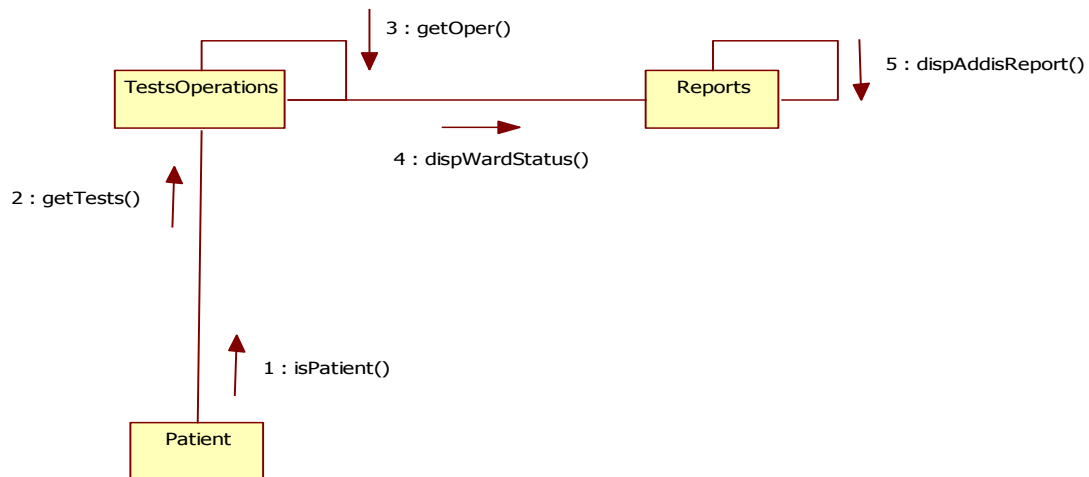
### PRESCRIBE TESTS:



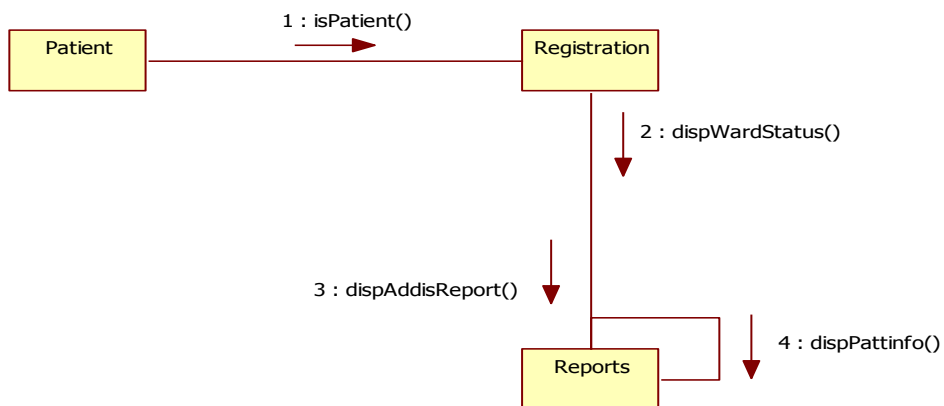
### WARDWISE BED STATUS:



### ADMISSION/DISCHARGE REPORTS:

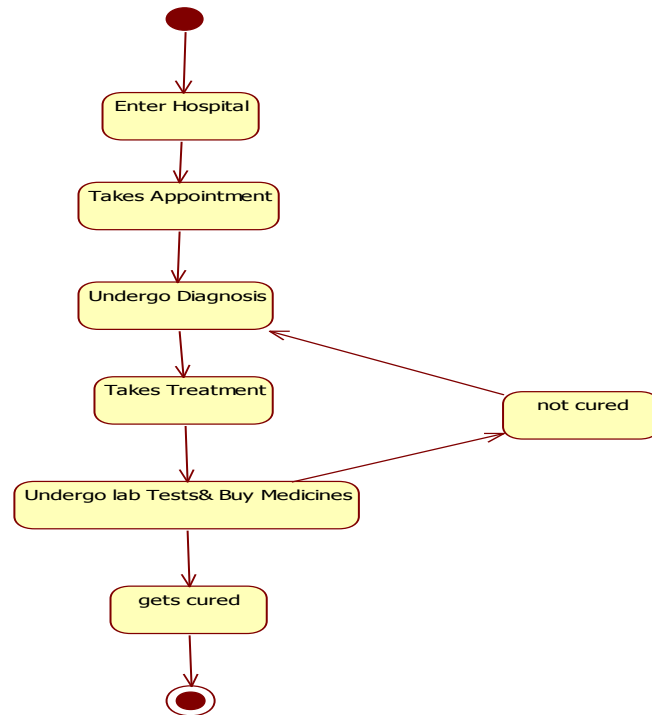


### PATIENT INFORMATION:

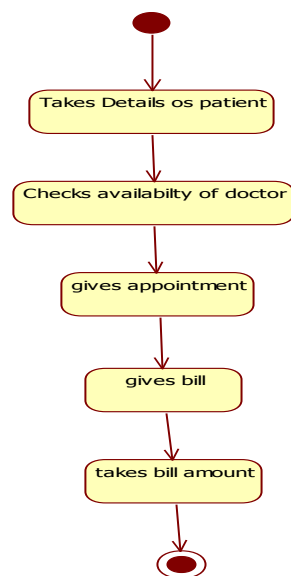


# STATECHART DIAGRAMS

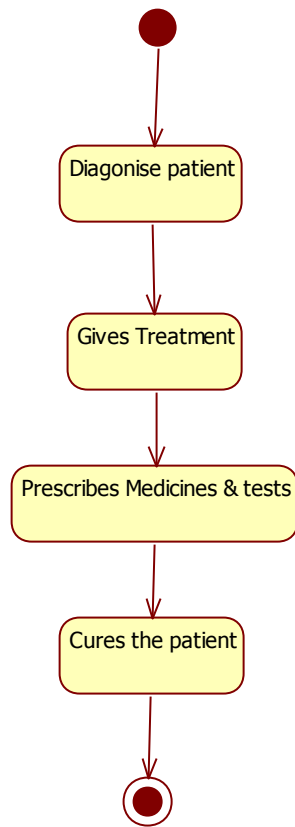
## PATIENT:



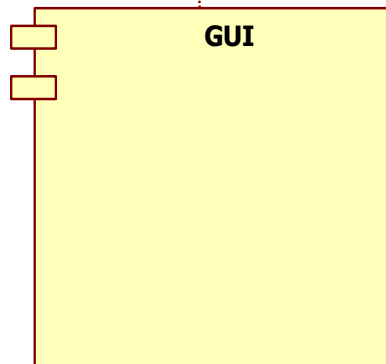
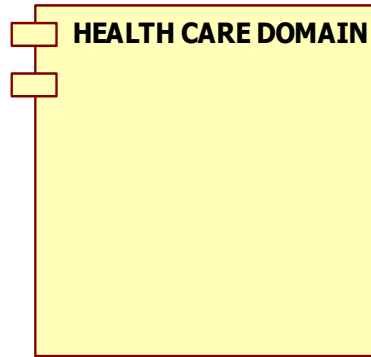
## RECPTIONIST:



DOCTOR:

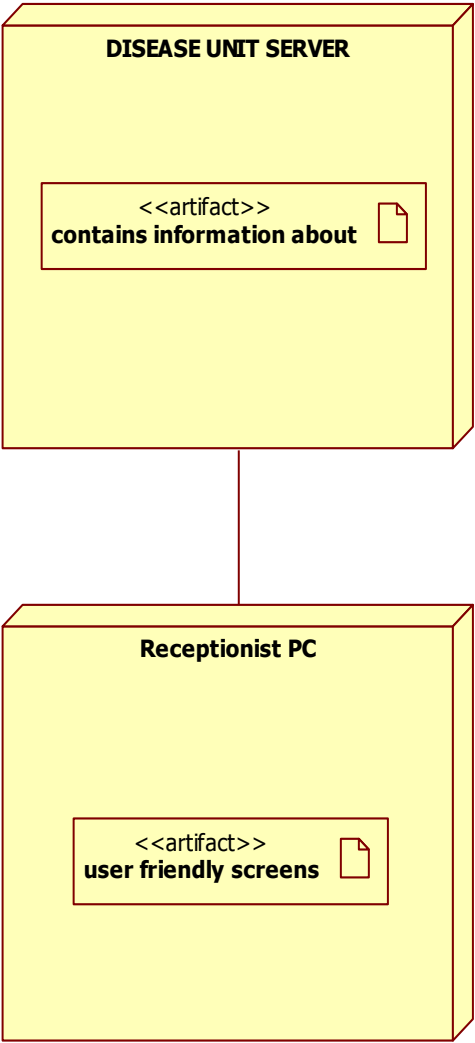


# COMPONENT DIAGRAM





# DEPLOYMENT DIAGRAM



## **SCOPE**

This project is made in such a way that it can be easily modified in more advance form if need arises. The database of this software maintains the records of the details of patients who give their tests in the hospital. This record will have a lot of use in the future related to any enquiry. The transactions are also revealed in this project. Any financial enquiry can be satisfied through this software. In future any fields can be added and deleted to and from the tables. Thus, this project has a wide application in the future. It can be used in any Hospital, Clinic, Dispensary or Pathology labs for maintaining patient details and their test results.

Computers are always changing. There are bugs to fix, enhancement to add and optimization to make. So changes have to be done in older version to make it applicable for current use and of current version to cater the need of future though efforts have been made to develop and error free system but no system are perfect, room for improvement is always there. Proper documentation has been done so that it will be easy in future to handle any breakdown or any other type of system activity.

The system has been developed putting the best efforts at all the levels of development.

There is a no. of factors that attribute to further improvement into a better package like

- ✓ The system has been developed keeping a personal computer user in mind .it can be improve into a system that supports network access.
- ✓ Graphical output has always been in case of analysis of data as compared to numerical data facility for generating graphical representation of the growth.

## **CONCLUSION**

The project **Hospital Management System (HMS)** is for computerizing the working in a hospital. The software takes care of all the requirements of an average hospital and is capable to provide easy and effective storage of information related to patients that come up to the hospital. It generates test reports; provide prescription details including various tests, diet advice, and medicines prescribed to patient and doctor. It also provides billing facility on the basis of patient's status.

# **BIBLIOGRAPHY**

## **Books**

- Mastering UML WITH Rational Rose  
By Wendy Boggs,  
Michael Boggs
- The Unified Modeling Language User Guide  
By Grady Booch,  
James Rumbaugh,  
Ivar Jacobson

## **Web sites**

[www.ecst.csuchico.edu](http://www.ecst.csuchico.edu)

[www.cs.fsu.edu](http://www.cs.fsu.edu)

[www.iranzik.com](http://www.iranzik.com)