ACKNOWLEDGEMENT

The success of this project is the outcome of enormous contribution of various people involved directly or indirectly with the project work. It is a pleasure to express our heartfelt thanks to them all.

It is our pleasure to thank **Mr. Jitendra Chaudhary (Project Guide, CDAC)** for his inspiring guidance, encouragement and supervision to complete the project entitled "Health Information Management System". This project would never have been possible without his guidance and supervision. We would like to acknowledge and thank **Ms**. **Avanti Joshi** for her generous help and guidance during our project work.

We are thankful to our **DHI Course Coordinator, Mrs. Purvi** for her support, encouragement and valuable guidance throughout our work.

Finally we are thankful to the staff members and colleagues who contributed to complete the project tangibly or intangibly.

Jai Sidhojirao Desai	**(PRN : 140240110001)**
R. Lokeshwaran	**(PRN : 140240110002)**
Melrick Dennis Pereira	**(PRN : 140240110003)**
Sourav Ghosh	**(PRN : 140240110005)**

(Feb 2014, DHI Batch Students)

TABLE OF CONTENTS

1.INTRODUCTION

1. ABOUT CDAC-ACTS

Centre for Development of Advanced Computing (C-DAC) is the premier R&D organization of the Department of Information Technology (DIT), Ministry of Communications & Information Technology (MCIT) for carrying out R&D in IT, Electronics and associated areas. Different areas of C-DAC, had originated at different times, many of which came out as a result of identification of opportunities.

- ➢ The setting up of C-DAC in 1988 itself was to build Supercomputers in context of denial of import of Supercomputers by USA. Since then C-DAC has been undertaking building of multiple generations of Supercomputer starting from PARAM with 1 GF in 1988.
- ➢ Almost at the same time, C-DAC started building Indian Language Computing Solutions with setting up of GIST group (Graphics and Intelligence based Script Technology); National Centre for Software Technology (NCST) set up in 1985 had also initiated work in Indian Language Computing around the same period.
- ➢ Electronic Research and Development Centre of India (ER&DCI) with various constituents starting as adjunct entities of various State Electronic Corporations had been brought under the hold of Department of Electronics and Telecommunications (now DIT) in around 1988. They were focusing on various aspects of applied electronics, technology and applications.
- ➢ With the passage of time as a result of creative echo system that got set up in C-DAC, more areas such as Health Informatics, etc., got created; while right from the beginning the focus of NCST was on Software Technologies; similarly C-DAC started its education & training activities in 1994 as a spin-off with the passage of time, it grew to a large efforts to meet the growing needs of Indian Industry for finishing schools.

CDAC-ACTS

Objectives of ACTS

- National initiative to promote finishing schools concept to meet the need of industry requirements.
- Skilled manpower development for knowledge creation in IT & Electronics.
- Offer courses and curriculum to accommodate dynamic nature of ICT requirements.
- Course curriculum with practical centric approach.
- To create industry ready professionals and providing career path.
- Offer basic, intermediate and advanced level courses in IT.

Quality Policy

C-DAC ACTS is an ISO 9001:2008 certified organization. The Quality Management System is effectively implemented at
C-DAC ACTS, Pune. We at C-DAC ACTS are committed to design, develop and deliver IT education, solutions and
services through a network of resources for bringing tangible benefits to our patrons.

Problem Statement

2. Project Scope

The proposed system can be used in any Government or Private Hospital, Clinic or Diagnostic Centre for
maintaining patient demographic details and their records of Medical Images and Clinical Reports. The
system should give role based access and control to the user or provider to take their decisions effectively.
The system can be linked to the TCP/IP network to transmit confidential clinical data across large distances.
Thus it can help in Patient and Patient Data management /transmission across Healthcare Centres.

1.3. Study of Existing Systems

Presently Majority of Health Information Management in Government Hospitals is done manually. All the information
and activities relies on paper work or in a broken, distributed and isolated manner. Automation to maintain patient
details requires a lot of time and user acquaintance is in itself a big challenge.

3. Limitation of Existing Systems

Normally the manual system has several drawbacks. They are as follows :-

- Since the Health Information Management process is carried out manually, it requires huge time.
- In present Government Healthcare systems [Indian Context] one needs to search for every file/ report manually.
- This system of data management leads to errors, mismanagement, patient inconvenience so many times.
- It is very difficult for one to search everything at the same place so, one needs to travel to different places.
- Sometimes the reports/files might not be available when one needs.

1.5. Proposed System Definition

With respect to the standards followed in the analysis phase for the identification of
the modules and sub modules with various functionalities are listed below :-

- ➢ Role Based HealthCare Provider Login Screen.
- ➢ Provider Profile & his/her Patient List Screen.
- ➢ Clinical Data Entry Form Screens for Provider.
- ➢ Patient Demographic/Clinical Data Save/Update Screen.
- ➢ Role Based View of Patient Details Screen.
- ➢ Upload/Download of Documents/Files from/to Network/DB.

2.PROPOSED SYSTEM

The Health Information Management System [HIMS] proposed by us uses HL7 standards on communication channel for image Acquisition/View, Storage/Transmission of Messages and Orders to/from different healthcare modules viz. Doctor's Module, Laboratory Information System [LIS], Radiology Information System [RIS]. We have created ADT_A04, OML_O21, OMI_O23, ORU_R01 messages using HL7 standard based SDKs[v 2.x] to be used for this purpose. The system is built using JAVA EE 6 (JSP/Servlets, JAVA Beans, JDBC Driver) and Oracle DB(11g).

2.1. Feasibility study

Feasibility study is a test of the proposal according to its workability, impact on the organization, ability to meet user needs and effective use of resources. The objective of feasibility study is not to solve the problem but to acquire a sense of its scope. The existing system is a manual system that requires a lot of manual work. And it consumes more time to search details and report from different places. Computers handle the proposed system easily with their speed, accuracy and repeatable tasks for highly important HealthCare Data.

Operational Feasibility

There is no difficulty in implementing the proposed Health Information Management System(HIMS), if the user has the knowledge of usage and working of the system. As user manual for the system will be designed and adequate user practice will be adviced, it is assumed that the user will not face any problem in running the system. The system does not affect the response rate of the computer. Needed assistance has been given to the user wherever needed, along with user manual, training and help menus. Thus the system is found to be operational.

Technical & Economic Feasibility

We are creating the system using open source softwares and associated APIs. The availability of standard basic computing resources with open source softwares installed are sufficient for creating the software system and implementation can be easily done with adequate digital storage and connectivity. Apart from users limited number of support engineers for system maintenance and run time bug fixing will be required for successfull running of the system. Thus, the system is economically viable at organizational level. With the latest storage options like cloud it seems feasible as well.

Time Feasibility

Time Feasibility is a determination of whether a proposed project can be implemented fully within a stipulated time frame. The time frame allotted for our project is one month, which was found acceptable.

2.2. HL7 Standards:

Introduction To HL7 Standards

HL7 and its members provide a framework (and related standards) for the exchange, integration, sharing, and retrieval of electronic health information. These standards define how information is packaged and communicated from one party to another, setting the language, structure and data types required for seamless integration between systems. HL7 standards support clinical practice and the management, delivery, and evaluation of health services, and are recognized as the most commonly used in the world.

HL7 standards are grouped into reference categories:

- **Section 1: Primary Standards -** Primary standards are considered the most popular standards integral for system integrations, inter-operability and compliance. Our most frequently used and in-demand standards are in this category.
- **Section 2: Foundational Standards** -Foundational standards define the fundamental tools and building blocks used to build the standards, and the technology infrastructure that implementers of HL7 standards must manage.
- **Section 3: Clinical and Administrative Domains -** Messaging and document standards for clinical specialties and groups are found in this section. These standards are usually implemented once primary standards for the organization are in place.
- **Section 4: EHR Profiles -** These standards provide functional models and profiles that enable the constructs for management of electronic health records.
- **Section 5: Implementation Guides -** This section is for implementation guides and/or support documents created to be used in conjunction with an existing standard. All documents in this section serve as supplemental material for a parent standard.
- **Section 6: Rules and References -** Technical specifications, programming structures and guidelines for software and standards development.
- **Section 7: Education & Awareness** – We can find HL7's Draft Standards for Trial Use (DSTUs) and current projects here, as well as helpful resources and tools to further supplement understanding and adoption of HL7 standards.

2.2.1. About HL7 :

Founded in 1987, Health Level Seven International (HL7) is a not-for-profit, ANSI-accredited standards developing
organization dedicated to providing a comprehensive framework and related standards for the exchange,
integration, sharing, and retrieval of electronic health information that supports clinical practice and the
management, delivery and evaluation of health services.
The HL7 Strategic Initiatives document is a business plan for products and services and was designed specifically
to meet the business needs of members and stakeholders. Derived from collaborative efforts with members,
government and non-government agencies and other standards development organizations, the Strategic Initiatives
are comprised of five high-level organizational strategies that are supported by a detailed tactical plan with clearly
defined objectives, milestones, and metrics for success.
Hospitals and other healthcare provider organizations typically have many different computer systems used for
everything from billing records to patient tracking. All of these systems should communicate with each other (or
"interface") when they receive new information but not all do so. HL7 specifies a number of flexible standards,
guidelines, and methodologies by which various healthcare systems can communicate with each other. Such
guidelines or data standards are a set of rules that allow information to be shared and processed in a uniform and
consistent manner. These data standards are meant to allow healthcare organizations to easily share clinical
information. Theoretically, this ability to exchange information should help to minimize the tendency for medical
care to be geographically isolated and highly variable.

2.2.2. Messages Used In The Proposed System :

ADT - Register a Patient (Event A04) :

An HL7 message is a hierarchical structure associated with a trigger event. The HL7 standard defines trigger event as
an event in the real world of health care (that) creates the need for data to flow among systems". Each trigger event
is associated with an abstract message that defines the type of data that the message needs to support the trigger
event.
The abstract message is a collection of segments, and includes the rules of repetition and inclusion for those
segments. The abstract message used in the HIMS is associated with the trigger event A04 – Register Patient. They
communicate patient demographic and visit information, as well as the reason why the message is

being sent. ADT messages are typically initiated by the HIS or a registration application, and are used to keep ancillary systems in sync regarding the state of a patient.

This is a basic introduction to HL7 ADT Messages. ADT Messages are extremely common in HL7 processing and are among the most widely used of all message types. They communicate patient demographic and visit information, as well as the reason why the message is being sent. ADT messages are typically initiated by the HIS or a registration application, and are used to keep ancillary systems in sync regarding the state of a patient.

HL7 ADT messages carry patient demographic information for HL7 communications but also provide important information about **trigger events** (such as patient admit, discharge, transfer, registration, etc.). Some of the most important segments in the ADT message are the **PID (Patient Identification) segment**, the PV1 (Patient Visit) segment, and occasionally the IN1 (Insurance) segment. ADT messages are extremely common in HL7 processing and are among the most widely used of all message types.

An A04 event signals that the patient has arrived or checked in as a one-time, or recurring outpatient, and is not assigned to a bed. One example might be its use to signal the beginning of a visit to the Emergency Room (= Casualty, etc.). Some systems refer to these events as outpatient registrations or emergency admissions. *PV1-44 - Admit Date/Time* is used for the visit start date/time.

The ROL - Role Segment is used in this message to communicate providers not specified elsewhere. Person level providers with an ongoing relationship are reported in the ROL segment following the PID/PD1 segments. Providers corresponding to the PV1 data are reported in the ROL segment following the PV1/PV2 segments. Providers related to a specific procedure are reported in the ROL segment following the PR1 segment. Providers related to a specific insurance are reported in the ROL segment following the IN1/IN2/IN3 segments. To communicate the begin- and end-date of the provider, use the *ROL-5 - Role Begin Date/Time* and the *ROL-6 - Role End Date/Time* in the ROL segment, with the applicable *ROL-3 - Role Code*.

OML - laboratory order message (event O21):

The following message structure may be used for the communication of laboratory and other order messages and must be used for lab automation messages where it is required that the Specimen/Container information is within the ORC/OBR segment group. While the ORM message with the OBR segment can be used for backwards compatibility for general lab messages, only the OML message should be used to take advantage of the specimen and container extensions required in laboratory automation.

The trigger event for this message is any change to a laboratory order. Such changes include submission of new
orders, cancellations, updates, etc. OML messages can originate also with a placer, filler, or an interested third party.
The additional patient information (the segments PID, PD1, PV1, PV2, etc, which are sent after the OBR with the
current order (indicated below with words "previous result"), could have been transferred with the previous result,
because the patient demographics related to the previous result can differ from the demographics related to the
current order. The current intent is to only allow references to the same patient as in the header PID.
The SAC segments included in the message allow the transfer of, e.g.: a laboratory order with multiple containers and
multiple test orders related to each container, or laboratory orders with test order requiring multiple containers.
The CTD segment in this trigger is used to transmit temporary patient contact details specific to this order.
In relationship to triggers O21, O33, and O35, this message/trigger (O21) should be used where an order with
multiple samples and optionally multiple containers per order item are to be communicated.

OMI – Imaging Order Message (Event O23) :

This message is used in communication between the information systems involved in the fulfillment of the request
directed to the imaging department, such as a Radiology Information System (RIS) and a Picture Archiving and
Communication System (PACS). For the purpose of the following discussion these systems will be identified as
Imaging Department Information Systems (IDIS). Information contained in the Imaging Procedure Control (IPC)
segment allows multiple IDIS to share the context of Imaging Studies (collections of images acquired, processed,
stored, and interpreted) in Image Management tasks.
The order for the imaging service is communicated between the Order Placer (such as an Order Entry system) and the
Order Filler (such as an RIS). In the imaging department environment, the Order Filler also identifies the set of
procedures (studies) and sub-procedures (procedure steps) that have to be performed in the process of fulfilling the
order. Each sub-procedure is performed using a single device (station). The Order Filler identifies the type of device .
and either a specific device or group of devices (for example, by geographic location) one of which is to be used in
performing the procedure step. Thus, the system performs an aspect of workflow management in the department.
Another information system in the department may be managing storage and distribution of the images within the
department as well as providing them to the enterprise. This system will have to operate within the same context as
the system managing the workflow. This context includes identifiers, content of the order, and details of procedures
and procedure steps that have to be performed to fulfill that particular order.

ORU – Unsolicited Observation Message (Event R01) :

The ORU message is for transmitting laboratory results to other systems. The OUL message is designed to accommodate the laboratory processes of laboratory automation systems.

With the segment (OBX) defined in this chapter, and the OBR defined in Chapter 4, one can construct almost any clinical report as a multi-level hierarchy, with the PID segment defined in Chapter 3 at the upper level, an order record (OBR) at the next level with one or more observation records (OBX), followed by the specimen information (SPM) and one or more observations (OBX) directly associated with the specimen.One result segment (OBX) is transmitted for each component of a diagnostic report, such as an EKG or obstetrical ultrasound or electrolyte battery.

The CTD segment in this trigger is used to transmit temporary patient contact details specific to this order.

Many report headers (OBR) may be sent beneath each patient segment, with many separate observation segments (OBX) related to the order / observation request beneath each OBR. OBX segments that are related to specimens immediately follow the SPM segments. Note segments (NTE) may be inserted at different locations in the message.

The note segment applies to the entity that immediately precedes it, i.e., the patient if it follows the PID segment, the observation request if it follows the OBR segment, and the individual result if it follows the OBX segment.

2.2.3. Introduction to DICOM Standards :

ACR (the American College of Radiology) and NEMA (the National Electrical Manufacturers Association) formed a joint committee to develop a Standard for Digital Imaging and Communications in Medicine.

This DICOM Standard was developed according to the NEMA Procedures.

This Standard is developed in liaison with other Standardization Organizations including CEN TC251 in Europe and JIRA in Japan, with review also by other organizations including IEEE, HL7 and ANSI in the USA.

The DICOM Standard is structured as a multi-part document using the guidelines established in the following document:

— ISO/IEC Directives, 1989 Part 3: Drafting and Presentation of International Standards.

This document is one part of the DICOM Standard which consists of the following parts:

PS 3.1: Introduction and Overview
PS 3.2: Conformance
PS 3.3: Information Object Definitions
PS 3.4: Service Class Specifications
PS 3.5: Data Structure and Encoding
PS 3.6: Data Dictionary
PS 3.7: Message Exchange
PS 3.8: Network Communication Support for Message Exchange
PS 3.9: Retired
PS 3.10: Media Storage and File Format for Data Interchange
PS 3.11: Media Storage Application Profiles
PS 3.12: Storage Functions and Media Formats for Data Interchange
PS 3.13: Retired
PS 3.14: Grayscale Standard Display Function
PS 3.15: Security and System Management Profiles
PS 3.16: Content Mapping Resource
PS 3.17: Explanatory Information
PS 3.18: Web Access to DICOM Persistent Objects (WADO)

2.3. Design

2.3.1. Data Flow Diagram :

A **data flow diagram (DFD)** is a graphical representation of the "flow" of data through an information system, modeling its process aspects. Often they are a preliminary step used to create an overview of the system which can later be elaborated. DFD's can also be used for the visualization of the data processing (structured design).

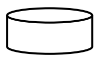
A DFD shows what kind of information will be input to and output from the systems, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of processes or information about whether processes will operate in sequence or in parallel (which is shown on a flowchart).

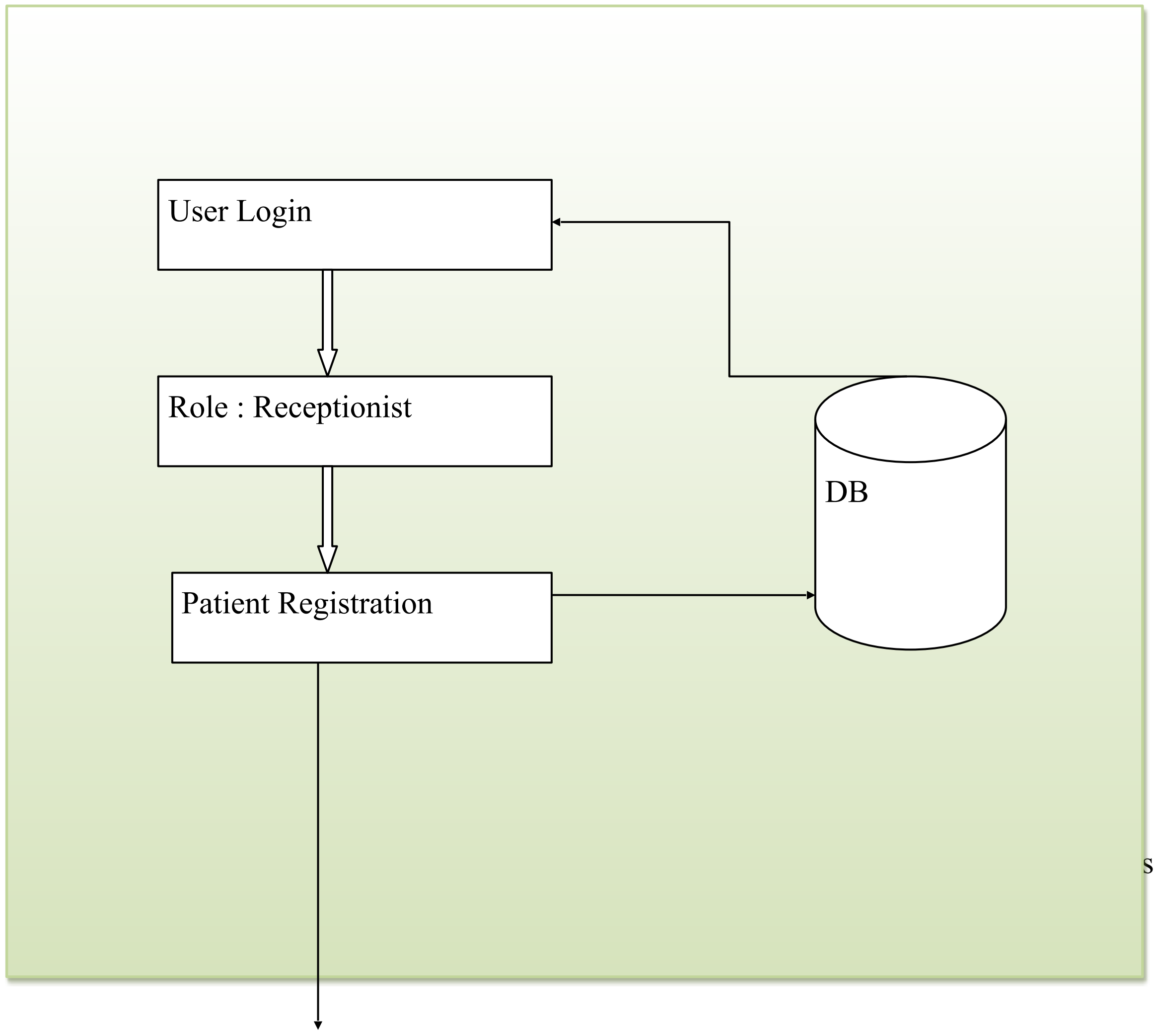
This context-level DFD is next "exploded", to produce a Level 1 DFD that shows some of the details of the system being modeled. The Level 1 DFD shows how the system is divided into sub-systems (processes), each of which deals with one or more of the data flow to or from an external agent, and which together provides all the functionality of the system as a whole.

It is the common practice to draw the context-level data flow diagram first, which shows the interaction between the system and external agents which acts as data sources and data sinks. On the context diagram the system's interactions with the outside world are modeled purely in terms of data flows across the system boundary. The context diagram shows the entire system as a single process, and gives no clues about its internal organization.

Data Flow Diagrams :

Receptionist's Panel :

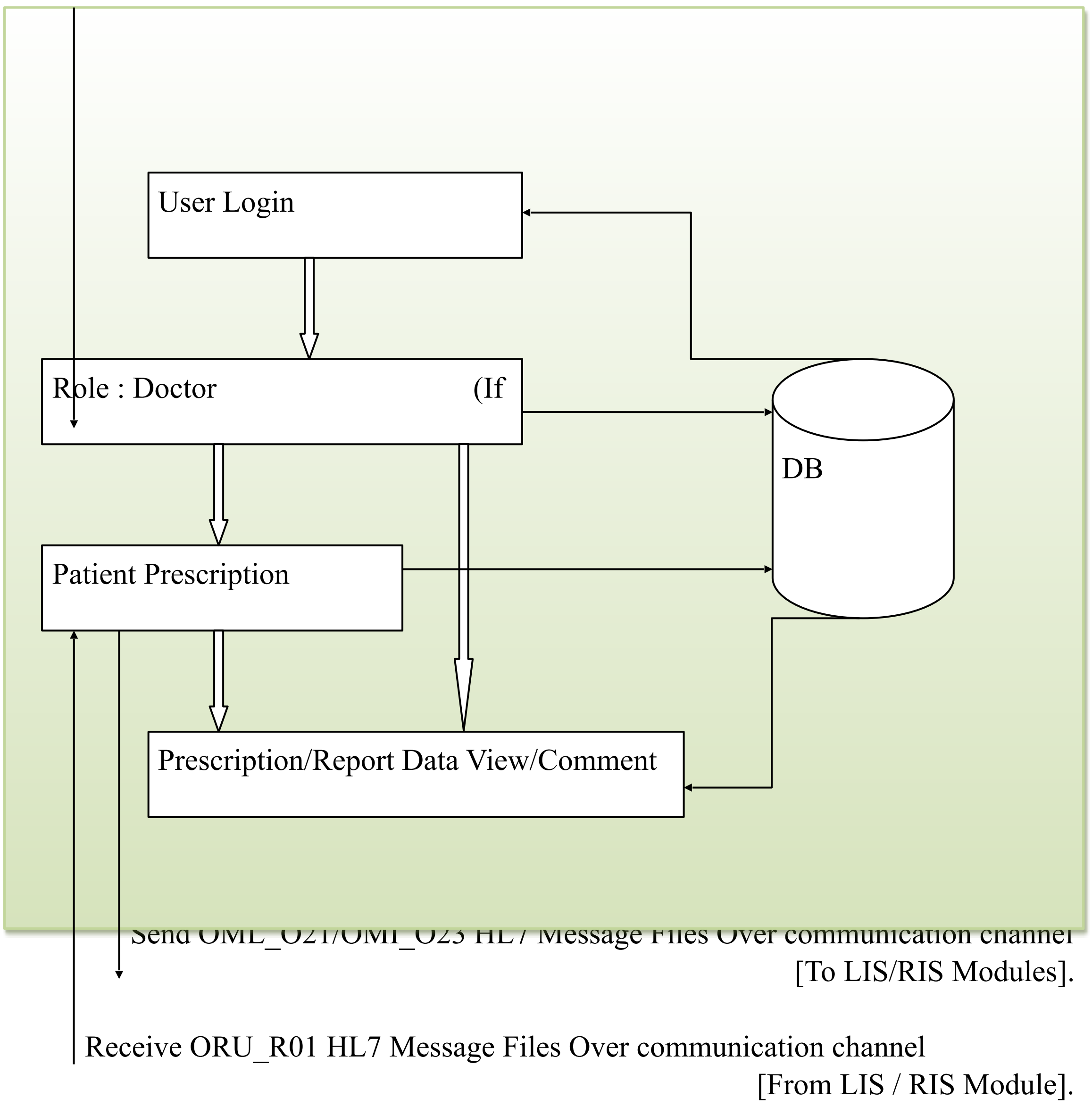




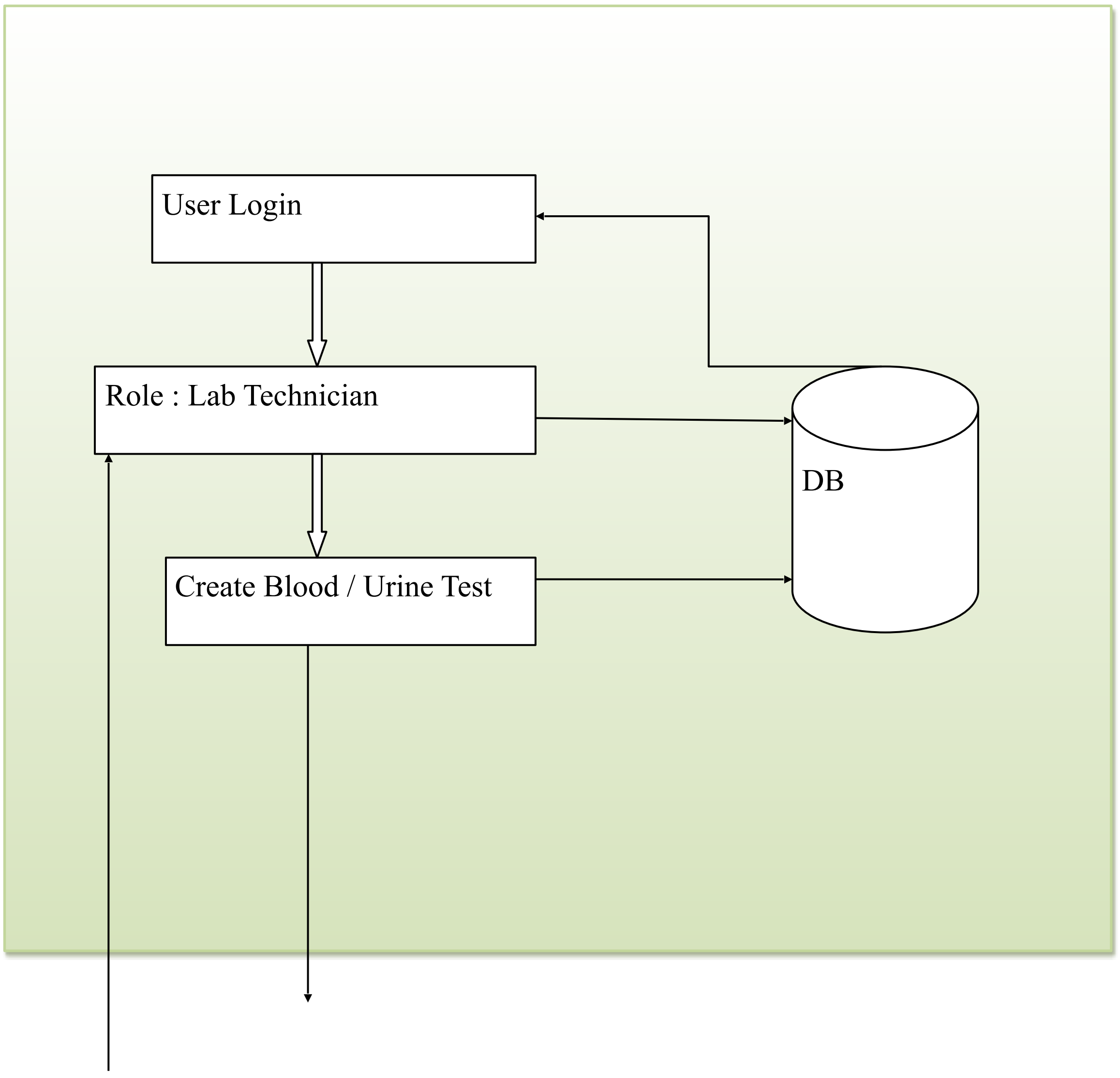
s Module]

Doctor’s Panel :

Receive ADT_A04 HL7 Message Files Over communication channel [From Receptionist’s Module]


User Login
Role : Doctor (If
DB
Patient Prescription
Prescription/Report Data View/Comment
Send OML_O21/OMI_O23 HL7 Message Files Over communication channel [To LIS/RIS Modules].
Receive ORU_R01 HL7 Message Files Over communication channel [From LIS / RIS Module].

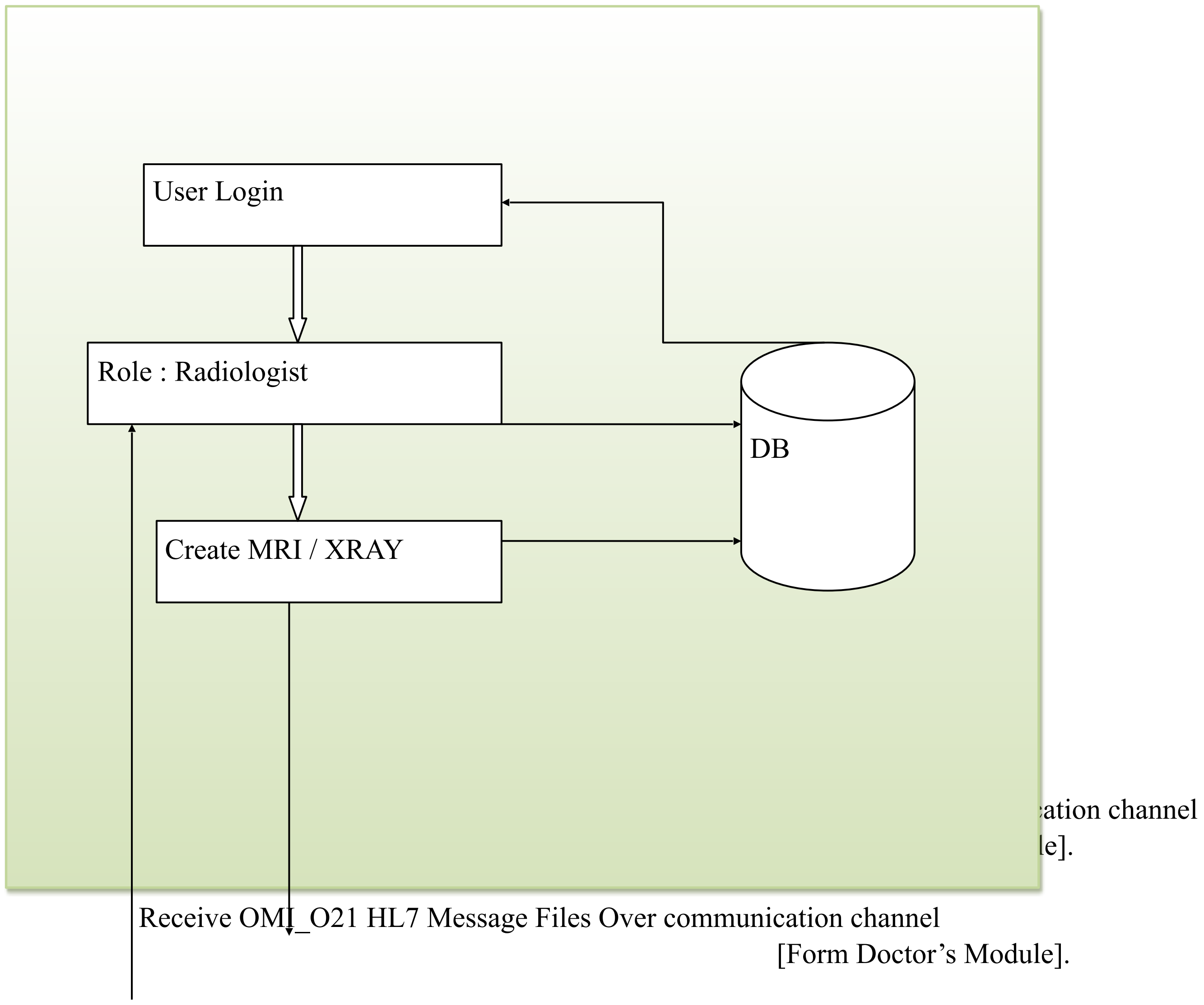
LIS Panel :



Send ORU_R01 HL7 Message Files Over communication channel [To Doctor’s Module].

Receive OML_O21 HL7 Message Files Over communication channel [From Doctor’s Module].

<u>RIS Panel :</u>



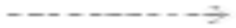
2.3.2. UML Diagrams :

About UML :

The unified modeling language is a standard language for specifying, visualizing, constructing and documenting the software system and its components. It is a graphical language, which provides a vocabulary and set of semantics and rules. UML focuses on the conceptual and physical representation of the system that must be constructed. It is used to understand, design, configure, maintain, and control information about systems.
It also captures the information about the static structure and dynamic behavior of the system. Even though UML is not a programming language, its tools can provide code generators from UML into various object oriented programming languages.

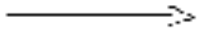
Relationships in the UML:

There are four kinds of relationships in the UML
Dependency : It is a semantic relationship between two things in which a change to one thing may affect the
semantics of the other thing .Graphically dependency is rendered as a dashed line, possibly directed and occasionally
including a level. It is represented as follows :

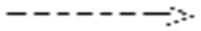


Association : Association is a relationship between two objects. In other words, association defines the multiplicity between objects. one-to-one, one-to-many, many-to-one, many-to-many all these words define an association between objects. Aggregation is a special form of association. Composition is a special form of aggregation.:

Generalization: Generalization uses a “is-a” relationship from a specialization to the generalization class. Common structure and behaviour are used from the specializtion to the generalized class. At a very broader level this can be understood as inheritance. Generalization is also called a “Is-a” relationship.:



Realization: Realization is a relationship between the blueprint class and the object containing its respective implementation level details. This object is said to realize the blueprint class. This can also be understood as the relationship between the interface and the implementing class.



USE CASE DIAGRAM :

Use-case diagrams graphically represent system behavior (use cases). These diagrams present a high level view of how the system is used as viewed from an outsider’s (actor’s) perspective. A use-case diagram may contain all or some of the use cases of a system. A use-case diagram can contain :

- Actors ("things" outside the system)
- Use cases (system boundaries identifying what the system should do)
- Interactions or Relationships between Actors and Use Cases in the system including the associations, dependencies, and generalizations.

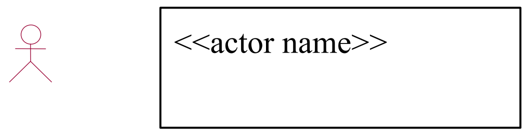
Use-case diagrams can be used during analysis to capture the system requirements and to understand how the
system should work. During the design phase, you can use use-case diagrams to specify the behavior of the system as implemented.

Identification of Actors:

Definition: An actor is someone or something that:

- Interacts with or uses the system.
- Provides input to and receives information from the system.
- Is external to the system and has no control over the use cases.

Graphical Representation:



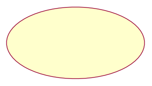
The actors identified in the system are:

.User
.Creator
.Verifier
.Approver

Identification of Use cases:

Definition: Use case is a sequence of transactions performed by a system that yields measurable result of values for a particular actor. The use cases are all the ways the system may be used.

Graphical Representation:



Use case name

Use cases identified by this system are:

➢ Receptionist Logs in
➢ Registers Patients by filling forms
➢ Submits the forms for DB Storage
➢ HL7 ADT Patient files get created
➢ File is transmitted through network to Doctor's Module

➢ Receptionist Logs Out
➢ Doctor Logs In
➢ HL7 ADT Patient file Data is extracted on Doctor's module
➢ Doctor creates Patient Prescription and suggests Lab, Radiology Tests
➢ Submits the forms for DB Storage
➢ HL7 OMI,OML Patient Prescription files get created
➢ File is transmitted through network to RIS, LIS modules
➢ Doctor Logs Out
➢ HL7 OMI,OML Patient Prescription files are extracted for test data on RIS, LIS modules
➢ Lab Assistant, Radiologist Logs in
➢ Both create suggested Lab, Radiology Test Reports, Images from their modules
➢ Both Submit the forms for DB Storage
➢ HL7 ORU file report, DCM Image file gets created
➢ File is transmitted through network to Doctor's module
➢ Lab Assistant, Radiologist Logs Out
➢ HL7 Files are extracted at Doctor's module for Doctor to view.
➢ Doctor Recommends medications.
➢ Doctor Logs out

Flow of Events:

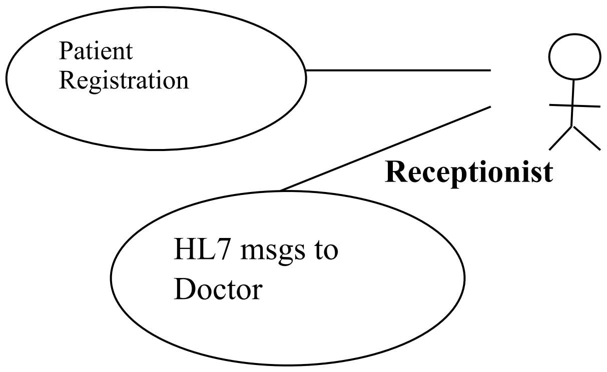
Flow of events should include

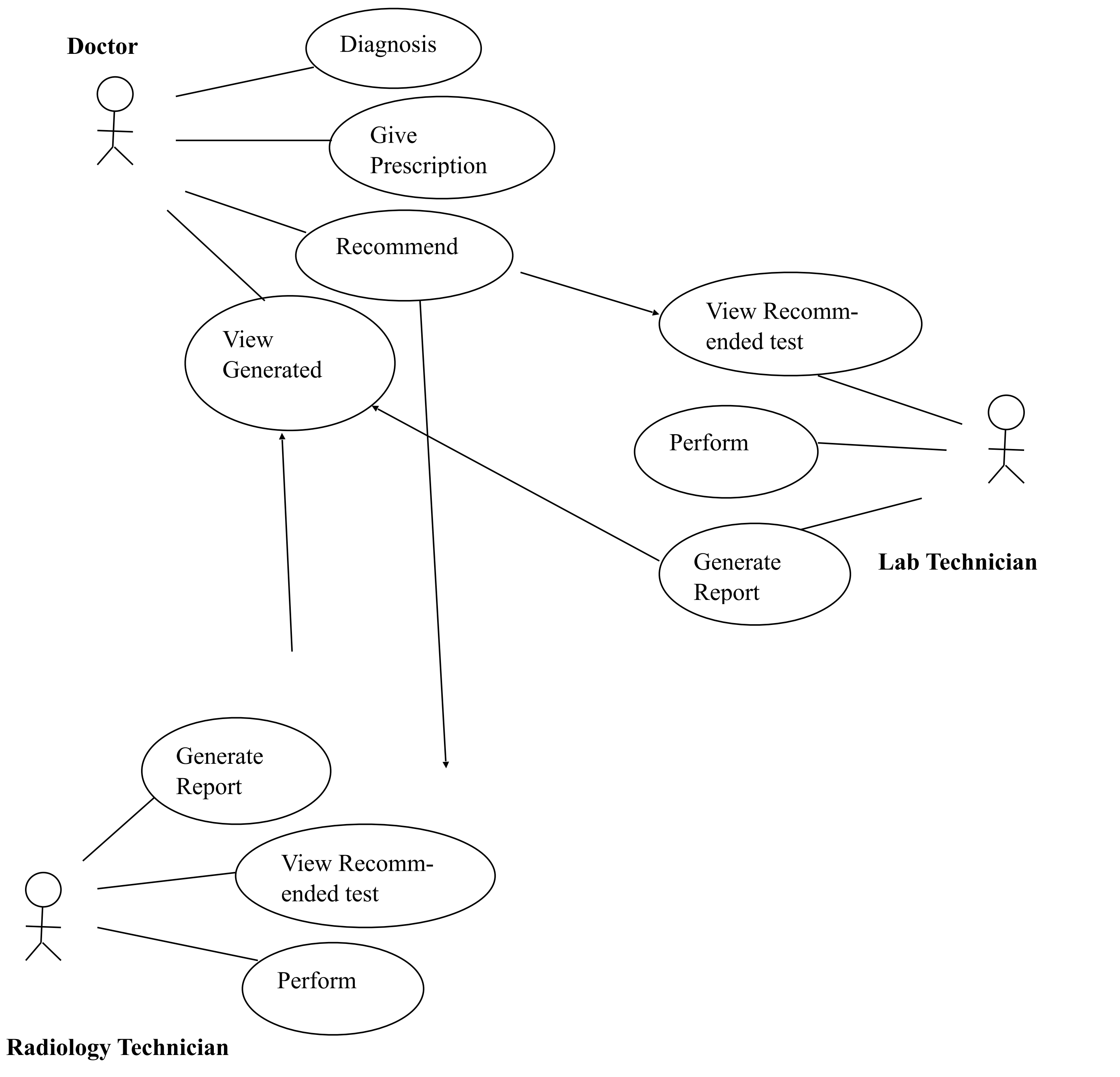
➢ When and how the use case starts and ends
➢ What interaction the use case has with the actors
➢ What data is needed by the use case
➢ The normal sequence of events for the use case
➢ The description of any alternate or exceptional flows

Relations:

Association Relationship:
An association provides a pathway for communication. The communication can be between use cases, actors,
classes or interfaces.
By default, the association tool on the toolbox is unidirectional and drawn on a diagram with a single arrow at one
end of the association. The end with the arrow indicates who or what is receiving the communication. Bidirectional communication is used to provide the two way communication.

Use Case Diagram :





2.3.3. CLASS Diagrams :

The class diagram is the main building block of object oriented modeling. It is used both for general conceptual modeling of the system of the application, and for detailed modeling, translating the models into programming

codes. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main
objects and or interactions in the application and the objects to be programmed. In the class diagram these classes
are represented with boxes which contain three parts

A class with 3 sections:

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

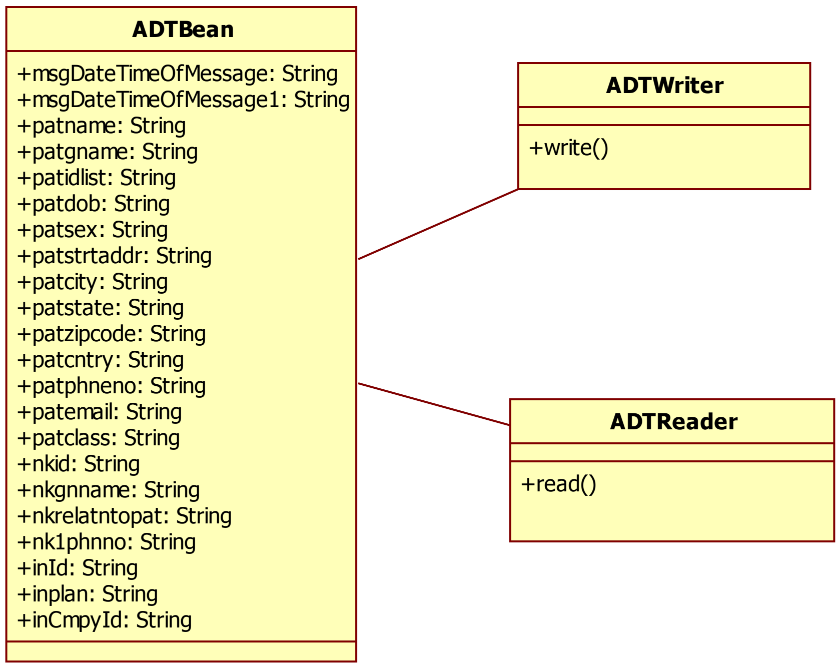
Class Diagrams contain the following things:

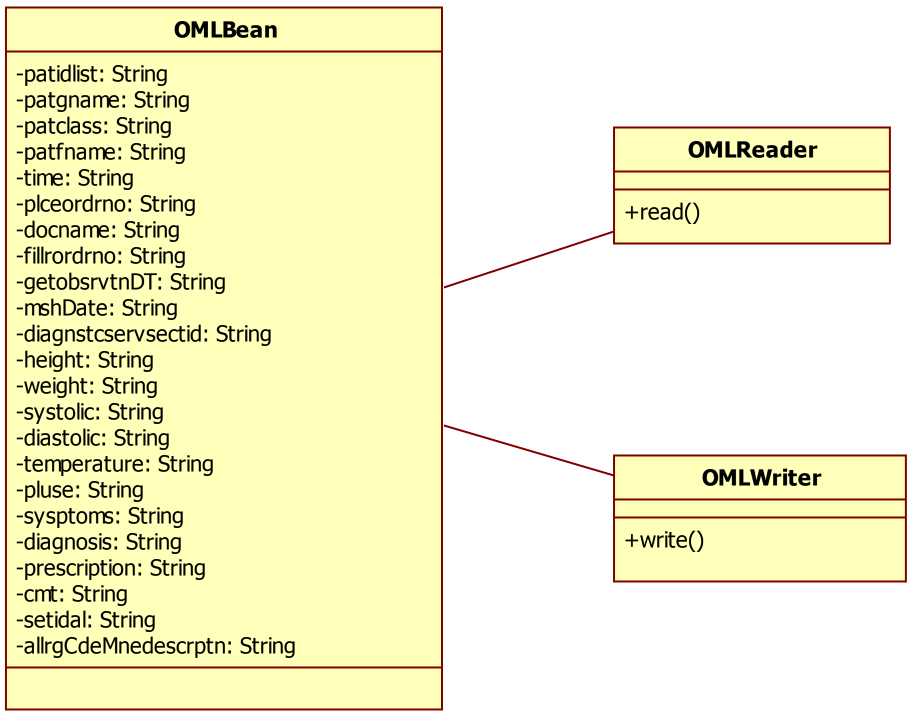
- Classes
- Interfaces
- Collaborations
- Dependency, Generalization, and association relationships

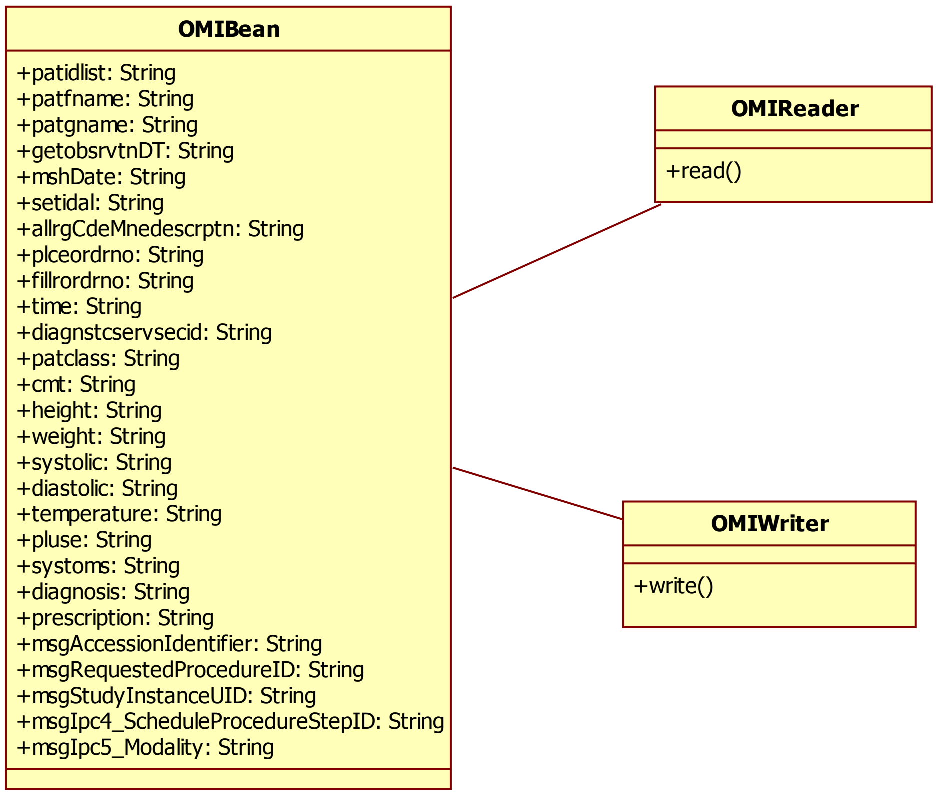
Class: A class is a description of a set of objects that share the same attributes, operations, relationships, and semantics.

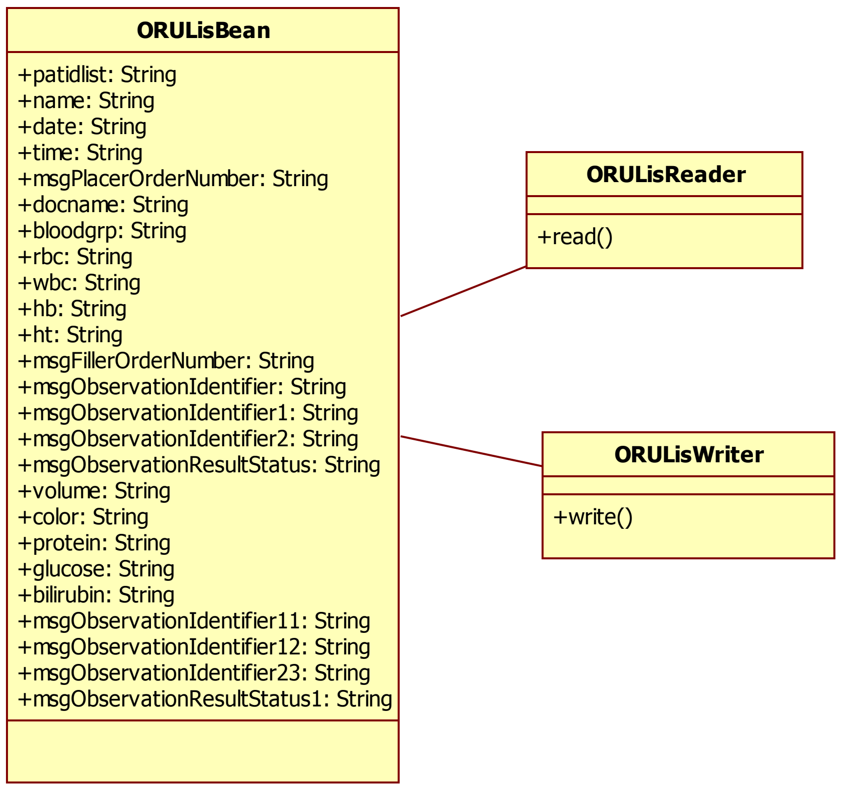
Interface: An interface is a collection of operations that specify a service of a class or component. It may represent the complete behavior of a class or component or only a part of that behavior. An interface is rendered as a circle together with its name.

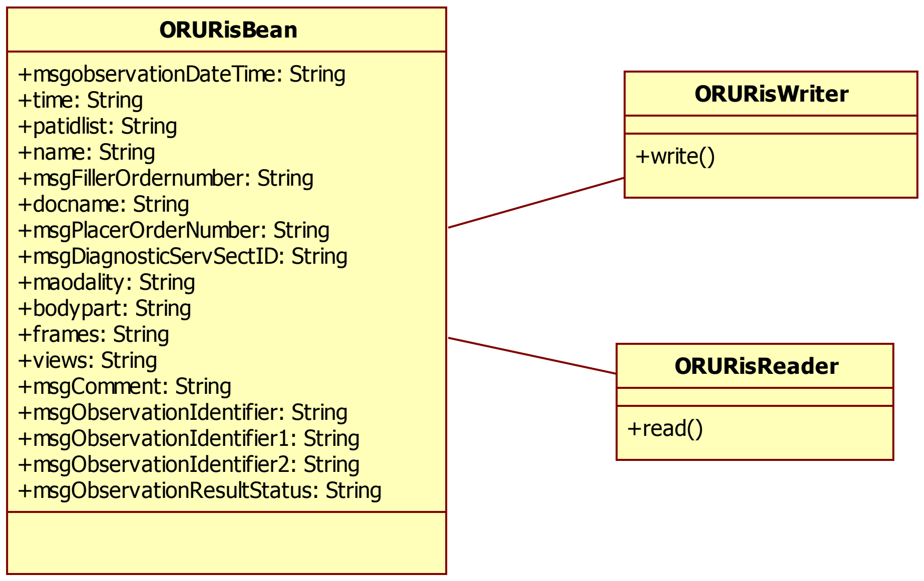
Collaboration: It is a society of roles and other elements that work together to provide some cooperative behavior that's bigger than the sum of all the elements. Collaboration is rendered as an ellipse with dashed lines, usually including only its name.


ADTBean
+msgDateTimeOfMessage: String
+msgDateTimeOfMessage1: String
+patname: String
+patgname: String
+patidlist: String
+patdob: String
+patsex: String
+patstrtaddr: String
+patcity: String
+patstate: String
+patzipcode: String
+patcntry: String
+patphneno: String
+patemail: String
+patclass: String
+nkid: String
+nkgnname: String
+nkrelatntopat: String
+nk1phnno: String
+inId: String
+inplan: String
+inCmpyId: String
ADTWriter
+write()
ADTReader
+read()


OMLBean
-patidlist: String
-patgname: String
-patclass: String
-patfname: String
-time: String
-plceordrno: String
-docname: String
-fillrordrno: String
-getobsrvtnDT: String
-mshDate: String
-diagnstcservsectid: String
-height: String
-weight: String
-systolic: String
-diastolic: String
-temperature: String
-pluse: String
-sysptoms: String
-diagnosis: String
-prescription: String
-cmt: String
-setidal: String
-allrgCdeMnedescrptn: String
OMLReader
+read()
OMLWriter
+write()


OMIBean
+patidlist: String
+patfname: String
+patgname: String
+getobsrvtnDT: String
+mshDate: String
+setidal: String
+allrgCdeMnedescrptn: String
+plceordrno: String
+fillrordrno: String
+time: String
+diagnstcservsecid: String
+patclass: String
+cmt: String
+height: String
+weight: String
+systolic: String
+diastolic: String
+temperature: String
+pluse: String
+systoms: String
+diagnosis: String
+prescription: String
+msgAccessionIdentifier: String
+msgRequestedProcedureID: String
+msgStudyInstanceUID: String
+msgIpc4_ScheduleProcedureStepID: String
+msgIpc5_Modality: String
OMIReader
+read()
OMIWriter
+write()


ORULisBean
+patidlist: String
+name: String
+date: String
+time: String
+msgPlacerOrderNumber: String
+docname: String
+bloodgrp: String
+rbc: String
+wbc: String
+hb: String
+ht: String
+msgFillerOrderNumber: String
+msgObservationIdentifier: String
+msgObservationIdentifier1: String
+msgObservationIdentifier2: String
+msgObservationResultStatus: String
+volume: String
+color: String
+protein: String
+glucose: String
+bilirubin: String
+msgObservationIdentifier11: String
+msgObservationIdentifier12: String
+msgObservationIdentifier23: String
+msgObservationResultStatus1: String
ORULisReader
+read()
ORULisWriter
+write()


ORURisBean
+msgobservationDateTime: String
+time: String
+patidlist: String
+name: String
+msgFillerOrdernumber: String
+docname: String
+msgPlacerOrderNumber: String
+msgDiagnosticServSectID: String
+maodality: String
+bodypart: String
+frames: String
+views: String
+msgComment: String
+msgObservationIdentifier: String
+msgObservationIdentifier1: String
+msgObservationIdentifier2: String
+msgObservationResultStatus: String
ORURisWriter
+write()
ORURisReader
+read()

2.4. Technologies Used :

Abbreviations and Definitions

- **UI:** User Interface
- **TCP/IP**: Transfer Control Protocol / Internet Protocol
- **Oracle11g:** Database
- **JDBC**: Java Database Connectivity
- **JDK**: Java Development Kit
- **ODBC**: Open Database Connectivity
- **JSP:** Java server pages

2.4.1. J2EE :

Java Platform, Enterprise Edition or Java EE is Oracle's enterprise Java computing platform. The platform provides an API and runtime environment for developing and running enterprise software, including network and web services, and other large-scale, multi-tiered, scalable, reliable, and secure network applications. Java EE extends the Java Platform, Standard Edition (Java SE), providing an API for object-relational mapping, distributed and multi-tier architectures, and web services. The platform incorporates a design based largely on modular components running on an application server. Software for Java EE is primarily developed in the Java programming language. The platform emphasizes Convention over configuration and annotations for configuration. Optionally XML can be used to override annotations or to deviate from the platform defaults.

The platform was known as Java 2 Platform, Enterprise Edition or J2EE until the name was changed to Java Platform, Enterprise Edition or Java EE in version 5. The current version is called Java EE 7.Java EE is defined by its specification. As with other Java Community Process specifications, providers must meet certain conformance requirements in order to declare their products as Java EE compliant.

Java EE includes several API specifications, such as JDBC, RMI, e-mail, JMS; web services, XML, etc., and defines how to coordinate them. Java EE also features some specifications unique to Java EE for components. These include Enterprise JavaBeans, Connectors, Servlets, Java Server Pages and several web service technologies. This allows developers to create enterprise applications that are portable and scalable, and that integrate with legacy technologies. A Java EE application server can handle transactions, security, scalability, concurrency and management of the components that are deployed to it, in order to enable developers to concentrate more on the business logic of the components rather than on infrastructure and integration tasks.

Oracle Application Server

In computing, the **Oracle Application Server 11g** ("g" stands for *grid*), consists of an integrated, standards-based software platform. It forms part of Oracle Corporation's Fusion Middleware technology stack. The heart of Oracle Application Server consists of Oracle HTTP Server (based on Apache HTTP Server) and OC4J (Oracle AS Containers for Java EE) which deploys Java EE-based applications. The latest version of OC4J offers full compatibility with the Java EE 1.6 specifications.

Oracle Application Server became the first platform designed for grid computing as well as with full life-cycle support for service-oriented architecture (SOA).

The current release of Oracle Application Server (11g), does not feature a metadata repository ties, relying instead on metadata repositories provided in previous releases.

Oracle Application Server 11*g* Enterprise Edition is an application platform suite (APS) that offers a comprehensive solution for developing, integrating, and deploying all of your enterprise's applications, portals, and Web sites. Enterprise Edition combines robust business integration, leading J2EE performance, and capabilities for enterprise portal, real-time business activity monitoring, business intelligence, identity management, and wireless deployment.

Eclipse

In computer programming, **Eclipse** is a multi-language Integrated development environment (IDE) comprising a base workspace and an extensible plug-in system for customizing the environment. It is written mostly in Java. It can be used to develop applications in Java and by means of various plug-ins, other programming languages including C, C++, COBOL, Fortran, Haskell, JavaScript, Perl, PHP, Python, Ruby (including Ruby on Rails framework), Closure, Groovy, Scheme, and Erlang. It can also be used to develop packages for the software Mathematical Development environments include the Eclipse Java development tools (JDT) for Java and Scale, Eclipse CDT for C/C++ and Eclipse PDT for PHP, among others.

The initial codebase originated from IBM Visual Age. The Eclipse software development kit (SDK), which includes the Java development tools, is meant for Java developers. Users can extend its abilities by installing plug-ins written for the Eclipse Platform, such as development toolkits for other programming languages, and can write and contribute their own plug-in modules.

Released under the terms of the Eclipse Public License, Eclipse SDK is free and open source software (although it is incompatible with the GNU General Public License).

The Eclipse Public License (EPL) is the fundamental license under which Eclipse projects are released. Some projects require dual licensing, for which the Eclipse Distribution License (EDL) is available, although use of this license must be applied for and is considered on a case-by-case basis.

The Eclipse was originally released under the Common Public License, but was later relicensed under the Eclipse Public License. The Free Software Foundation has said that both licenses are free software licenses, but are incompatible with the GNU General Public License (GPL). Mike Milinkovich, of the Eclipse Foundation commented that moving to the GPL would be considered when version 3 of the GPL was released.

Database: Apache Derby (previously distributed as **IBM Cloudscape**) is a Relational Database management System (RDBMS) developed by the Apache Software Foundation that can be embedded in Java programs.

Currently Derby comes with Java 7 and has been branded as "JavaDB" but is exactly the same bit-for-bit as Derby is. For developers wanting to use Java 6, they can still download Derby as before, but for developers requiring JRE 7 or later, Derby is included in the Java API.

Java:
Introduction to Java
Java is a high level, third-generation programming language, like C, Fortran, Perl and many others. It is a platform for
distributed computing – a development and run-time environment that contains built-in support for the World Wide Web.

History of Java
Java development began at Sun Microsystems in 1991, the same year the World Wide Web was conceived. Java's creator, James Gosling did not design java for the Internet. His Objective was to create a common development environment for consumer electronic devices which was easily portable from one device to another.This effort evolved into a language , code named Oak and later renamed Java that retains much of the syntax and power of c++ , but is simpler and more platform independent.

Java Features
- ➢ Simplicity
- ➢ Orientation
- ➢ Platform Independence

➢ Security
➢ High Performance
➢ Multi Threading
➢ Dynamic linking.
➢ Garbage Collection.

One of the most important features of Java is Platform Independence which makes it famous and suitable language for World Wide Web.

Why java is Platform Independent?

Java is Platform Independent because of Java Virtual Machine (JVM).

Java Virtual Machine (JVM)
The client application or operating system must have a java byte-code interpreter to execute byte-code instructions. The interpreter is a part of a lager program called the JVM. The JVM interprets the byte code into native code and is available on a platform that supports java.

JavaBeans with JSP: Java Server Pages (JSP) is another Java technology for developing web applications. JSP was released during the time servlet technology had gained popularity as one of the best web technologies available. JSP is not meant to replace servlets, however. In fact, JSP is an extension of the servlet technology, and it is common practice to use both servlets and JSP pages in the same web applications. The JSP Technology is based on JSP API that consists of two packages

➢ Javax.servlet.jsp
➢ Javax.servlet.jsp.tagext

One of the most useful features of Java Server Pages is that JSP directly supports the use of JavaBeans through the <jsp:useBean> tag. Java Beans, as you probably know, are Java classes that can be loaded by name and otherwise conform to a specific set of rules. These rules include the following:

➢ Being a public class: public class JavaBean
➢ Having a public, no argument constructor: public JavaBean ()
➢ Using private data fields only: private String message
➢ Providing public, no-argument access methods for its private data fields:
    public getMessage()

    public setMessage(String message)

➢ Supporting introspection — the ability of an external class to query a bean for its behavior.

A big advantage of using JavaBeans in JSP applications is that they allow you to implement the logic of the JSP page as a separate Java class and "plug it in." This approach offers these significant advantages:

➢ Separation of content from logic
➢ Reusable plug-in components for common tasks

When they are used with Java Server Pages, JavaBeans have two primary functions. They can be used as logic blocks, where their primary purpose is to separate logic from display, and as storage classes, where their primary purpose is data storage.

Javascript:
Javascript is used for client side validation. Client side validation controls perform input checking in serve code. When the user submits a form to the server, the validation controls are invoked to review the users input, control bye control. If an error has occurred in any of the input controls, the page itself is set to an invalid state so one can test for validity before code runs.

HTML:

HTML is a language that puts the face on the web by helping to prepare documents for online publication. These documents are also called as web documents and each HTML document is known as web pages. HTML is a standard language the all Web browser can understand and interpret. HTML is the way of representing text linking that text to other kinds of resources including sound files graphics files, multimedia files etc. that allows these kinds of data to be displayed together, to set them and reinforce one another. As delivered by the web server, HTML is nothing more than a plain text file. HTML instructions divide the text of a document into blocks called elements.

Oracle11g :

Oracle11g Database provides efficient, reliable, secure data management for high-end applications such as high-volume on-line transaction processing (OLTP) environments, query-intensive data warehouses, and demanding Internet applications. Oracle also offers several additional optional database products that enhance the capabilities of Oracle9*i* Database for specific application requirements.

Java Database Connectivity (JDBC) :

JDBC is a Java-based data access technology (Java Standard Edition platform) from Oracle Corporation. This technology is an API for the Java programming language that defines how a client may access a database. It provides methods for querying and updating data in a database. JDBC is oriented towards relational databases. A JDBC-to-ODBC bridge enables connections to any ODBC-accessible data source in the JVM host environment.
The Java Database Connectivity (JDBC) API is the industry standard for database-independent connectivity between the Java programming language and a wide range of databases – SQL databases and other tabular data sources, such as spreadsheets or flat files. The JDBC API provides a call-level API for SQL-based database access.

JDBC technology allows you to use the Java programming language to exploit "Write Once, Run Anywhere" capabilities for applications that require access to enterprise data.

JDBC allows multiple implementations to exist and be used by the same application. The API provides a mechanism for dynamically loading the correct Java packages and registering them with the JDBC Driver Manager. The Driver Manager is used as a connection factory for creating JDBC connections.

JDBC connections support creating and executing statements. These may be update statements such as SQL's CREATE, INSERT, UPDATE and DELETE, or they may be query statements such as SELECT. Additionally, stored procedures may be invoked through a JDBC connection. JDBC represents statements using one of the following classes:

- Statement – the statement is sent to the database server each and every time.
- PreparedStatement – the statement is cached and then the execution path is pre-determined on the database server allowing it to be executed multiple times in an efficient manner.
- Callable Statement – used for executing stored procedures on the database.

Update statements such as INSERT, UPDATE and DELETE return an update count that indicates how many rows were affected in the database. These statements do not return any other information.

Query statements return a JDBC row result set. The row result set is used to walk over the result set. Individual columns in a row are retrieved either by name or by column number. There may be any number of rows in the result set. The row result set has metadata that describes the names of the columns and their types.

Features :

The strut-2 framework is designed for the compilation of the entire development cycle including of building, developing and maintaining the whole application. It is very extensible as each class of the

framework is based on an Interface and all the base classes are given an extra application and even you can add your own. The basic platform requirements are Servlet API 2.4, JSP API 2.0 and Java 5.
Some of the general features of the current Apache Strut 2 framework are given below.

Architecture – First the web browser request a resource for which the Filter Dispatcher decides the suitable action. Then the Interceptors use the required functions and after that the Action method executes all the functions like storing and retrieving data from a database. Then the result can be seen on the output of the browser in HTML, PDF, images or any other.

Tags - Tags in JSP 2 allow creating dynamic web applications with less number of coding. Not only these tags contain output data but also provide style sheet driven markup that in turn helps in creating pages with less code. Here the tags also support validation and localization of coding that in turn offer more utilization. The less number of codes also makes it easy to read and maintain.

MVC :
The Model View Controller in Strut 2 framework acts as a coordinator between application's model and web view. Its Controller and View components can come together with other technology to develop the model. The framework has its library and markup tags to present the data dynamically.

Configuration –
Provides a deployment descriptor to initialize resources in XML format. The initialization takes place simply by scanning all the classes using Java packages or you can use an application configuration file to control the entire configuration. Its general-purpose defaults allow using struts directly Out of the box. Configuration files are re-loadable that allows changes without restarting a web container.

Other Features:

- All framework classes are based on interfaces and core interfaces are independent from HTTP.
- Check boxes do not require any kind of special application for false values.
- Any class can be used as an action class and one can input properties by using any JavaBeans directly to the action class.
- Strut 2 actions are spring friendly and so easy to spring integration.
- AJAX theme enables to make the application more dynamic.
- Portal and servlet deployment are easy due to automatic port let support without altering code.

Oracle Database Application Development:
SQL and PL/SQL form the core of Oracle's application development stack. Not only do most enterprise back-ends run SQL, but Web applications accessing databases do so using SQL (wrappered by Java classes as JDBC), Enterprise Application Integration applications generate XML from SQL queries, and content-repositories are built on top of SQL tables. It is a simple, widely understood, unified data model. It is used standalone in many applications, but it is also invoked directly from Java (JDBC), Oracle Call Interface (OCI), Oracle C++ Call Interface (OCCI), or XSU (XML SQL Utility). Stored packages, procedures, and triggers can all be written in PL/SQL or in Java.

2.4.2. Oracle SQL :
SQL is the programming language that defines and manipulates the database. SQL databases are relational databases, which means that data is stored in a set of simple relations.

SQL Statements:
All operations on the information in an Oracle database are performed using SQL statements. A SQL statement is a string of SQL text. A statement must be the equivalent of a complete SQL sentence.

Only a complete SQL statement can run successfully. A SQL statement can be thought of as a very simple, but powerful, computer program or instruction.

Oracle PL/SQL :

PL/SQL is Oracle's procedural language extension to SQL. PL/SQL combines the ease and flexibility of SQL with the

procedural functionality of a structured programming language, such as IF ... THEN, WHILE, and LOOP.

When designing a database application, consider the following advantages of using stored PL/SQL:

- PL/SQL code can be stored centrally in a database. Network traffic between applications and the database is reduced, so application and system performance increases. Even when PL/SQL is not stored in the database, applications can send blocks of PL/SQL to the database rather than individual SQL statements, thereby reducing network traffic.
- Data access can be controlled by stored PL/SQL code. In this case, PL/SQL users can access data only as intended by application developers, unless another access route is granted.
- PL/SQL blocks can be sent by an application to a database, running complex operations without excessive network traffic.
- Oracle supports PL/SQL Server Pages, so your application logic can be invoked directly from your Web pages.

2.4.3. HL7 APIs :

HL7 v2 Java API (HAPI) is a full featured Java API which we can use to add HL7 capabilities to our applications. This is a messaging interface built specifically for healthcare Industry. It provides Open Source, Object-Oriented HL7 2.x parser for Java which is used for developing applications based on the emerging HL7 specification.

2.4.4. Testing Methods :

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or services under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risk of software implementation. Test techniques include, but are not limited to, the process of executing a program or application with the intent of finding software bugs (errors or other defects).

Software testing can be stated as the process of validating and verifying that a computer program/ application/product:

- Meet the requirement that guided its design and development,
- Works as expected, can be implemented with the same characteristics, and satisfy the needs of stakeholders.

Software testing, depending on the testing methods employed, can be implemented at any time in the development processes. Traditionally most of the effort occurs after the requirements have been defined and the coding process has been completed, but in the agile approaches most of the test effort is on-going. As such, the methodology of the test is governed by the chosen software development methodology.

Testing levels

Tests are frequently grouped by where they are added in the software development process, or by the level of specificity of the test. The main levels during the development process as defined by the SWEBOK guide are unit-integration and system testing that are distinguished by the test target without implying a specific process model. Other test levels are classified by the testing objective.

Unit Testing

It focuses verification effort on smallest unit of software design-the software component or module. Important controls parts are tested to uncover errors within the boundary of module. Unit test is White Box oriented.

Integration Testing

It conducts tests to uncover errors associated with interfacing. The objective to take unit tested components & build a program structure that has been dedicated by design. Software components may be integrated in an iterative way or all together (Big Bang).Normally the former is considered a better practice since it allows interface issues to be localized more quickly and fixed.

Integration testing works to expose defect in interface and interaction between integrated components (Modules).Progressively large group of software components corresponding to elements of the architectural Design are integrated and tested until the software works as a System.

System Testing

System testing tests a completely integrated system to verify that it meets its requirements.

System Integration Testing

System integration testing verifies that a system is integrated to any external or third party system defines in the system requirements.

Acceptance Testing

Acceptance testing can mean one of the two things:

A smoke test is used as an acceptance test prior to introducing a new build to the main testing process, i.e. before integration of regression.

Acceptance testing is perform by the customer, often in their lab environment on their own HW, is known as user acceptance test (UAT). Acceptance testing may be performing as part of the hand off the process between any two phases of development.

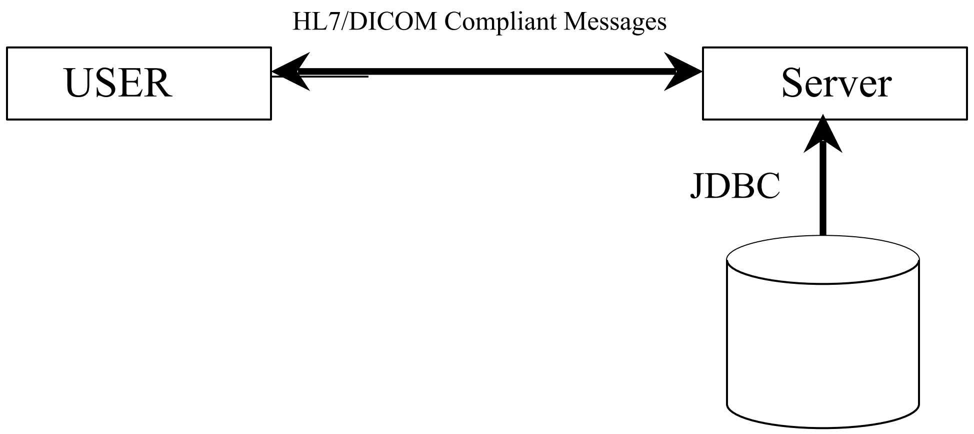
3.OVER ALL DESCRIPTION

3.1. Product Perspective:

This section covers the various phases of the system which is used to develop the solution for the requirements requested. It covers user interfaces associated with the system along with their functionalities. The section also deals with the basic software and hardware requirements needed for the system. The various User Characteristics, Constraints associated and the Assumptions and Dependencies are also discussed under this section.

3.2. Specific Requirements :

3.2.1. System Interface :



The above diagram gives a perspective of the full product. It is Client-Server architecture, where the clients initiate a request and the server processes the request that came from the client and responds with the processed data/information.

3.2.2. User Interface :

User interface is used to provide communication between users and system. Our product should have communication between them. As HIMS is a web-based application and it should get input from users for processing. Firstly the system will require its users to register as a Healthcare Provider/Administrator or

Owning Organization and when they will register, they can use the various services provided to them by the system like Patient Data Storage/management/transmission. Then, the system will study to find an optimized result for users according to the inputs given by the users. After that, the system will give appropriate output for users via user interface.

User interface elements should be easy to understand. Part of user interface should be well-organized on screen and the parts should be concatenated right. When users look at the interface, they understand which pane is used for which purpose. Each task of an interface should be specified clearly and users should use them correctly. For example, when users press to any button on interface, they should know which operations are done by pressing this button.

The user interface should be easy to learn. When users use the user interface, they should know which element is used to which operations. We should teach using of the user interface to users simply. If it is hard to learn, then teaching will take long time and there will be an extra cost for teaching of product.

The interface actions and elements should be consistent. When users press any button, required actions should be done by the system.

The screen layout and color of the user interface should be appealing. When users look at the screen, it will have a nice vision. Colors will be selected clearly, thus eyes of users won't be tired.

3.2.3. Hardware Interface :

HIMS is a web-based project. So any personal computer, which has an internet browser, is enough to use this

system.

3.2.4. Software Interface :

In HIMS, users will use application program via the user interface program. When database management system access is required, the system establishes a connection to the database management system, once the connection is created; the client program can communicate with the database management system. The Java database connectivity (JDBC) Standard provides application programming interface (API), which allows client side program to call database management system. A user can actually connect to several database management system and send query and transaction requests using JDBC API, which are then processed at server site. Any query results are sent back to user, which can process or display result as needed.

When Clinical Data transmission is required, HL7 & DICOM API based application programs will be used to create HL7, DICOM Standard Compliant files and the same be sent over TCP/IP network. API based Programs to extract the clinical data from the received files will be used to regenerate the Data for viewing.

- ➢ **Operating System**: Microsoft Windows 7.
- ➢ **IDE**: Eclipse Indigo
- ➢ **Database Server**: Oracle11g.
- ➢ **Server Side Language**: JSP, Servlets, Java Beans.
- ➢ **Client Side Language / Front End**: HTML, Java Script.
- ➢ **Communication between Client & Server**: Apache Tomcat 6.0.

3.2.5. Communication Interface :

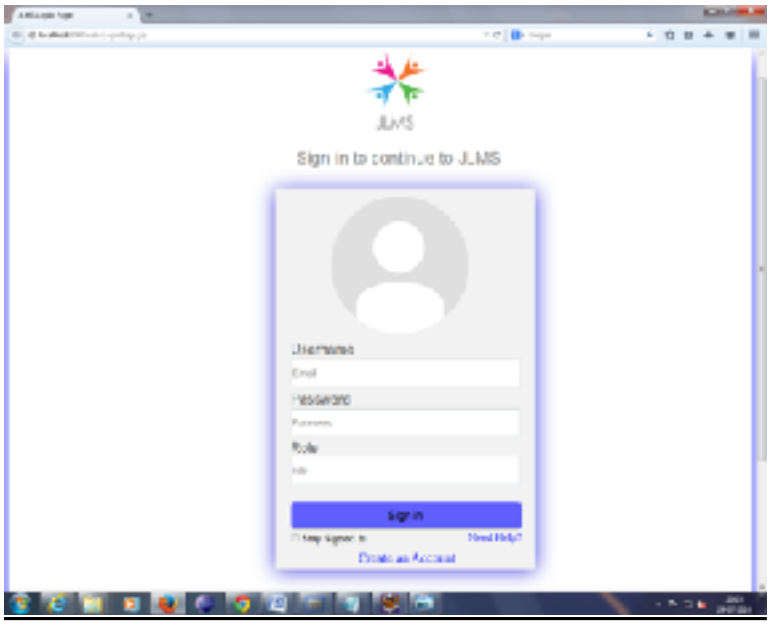
Hypertext Transfer Protocol (HTTP) shall be used to provide a communication between system and users. Because, HIMS is a web-based system which should be reached on World Wide Web by users. HTTP is a request/response protocol between a client and a server. The client making a HTTP request, such as web browser, is referred to as the user agent. The responding server, which stores or creates resources such as HTML files and images, is called the origin server. In between the user agent and origin server may be several intermediaries, such as proxies, gateways, and tunnels. HTTP is not constrained to using TCP/IP and its supporting layers, although this is its most popular application on internet, or on other networks. HTTP only presumes a reliable transport; any protocol that provides such guarantees can be used. Typically, an HTTP client initiates a request by establishing a Transmission Control protocol (TCP) connection to a particular port on a host (port 80 by default). An HTTP server listening on that port waits for the client to send a request message.

The communication messages will be HL7 messages which comply to the HL7 medical standards

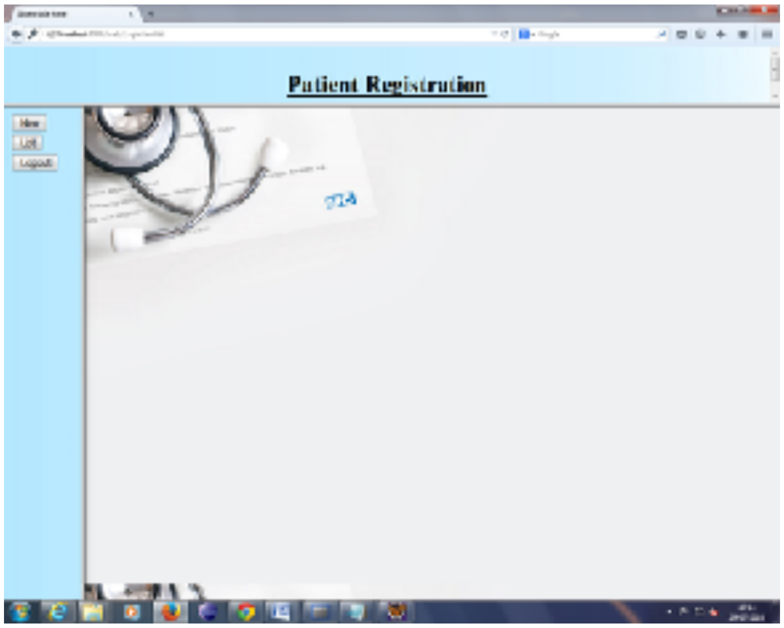
.

3.3. Product Functionality [User Interfaces] :

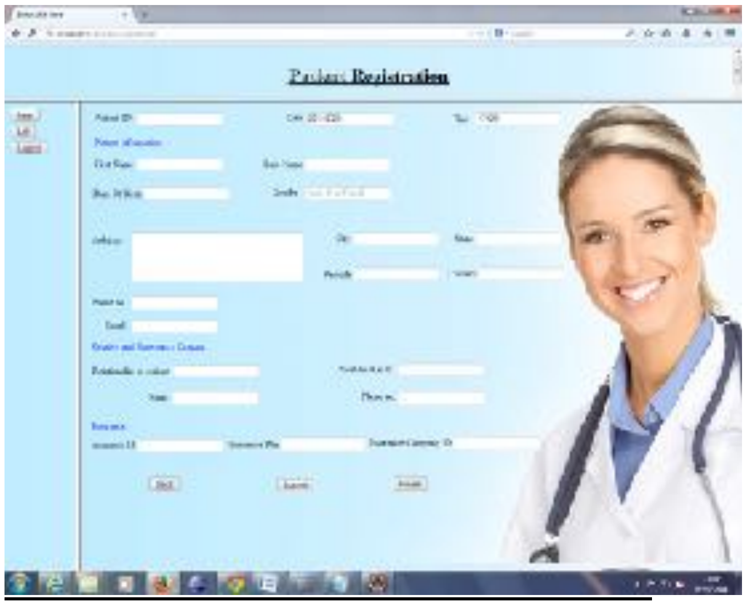
User Login Page :



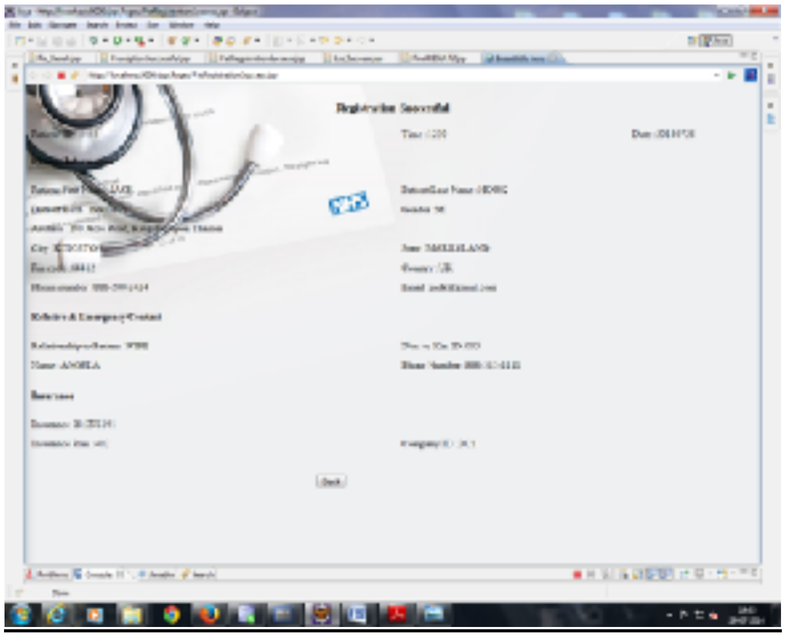
Receptionist's Home Page :



Patient Registration Form :



Receptionist’s View Registered Patient Page :

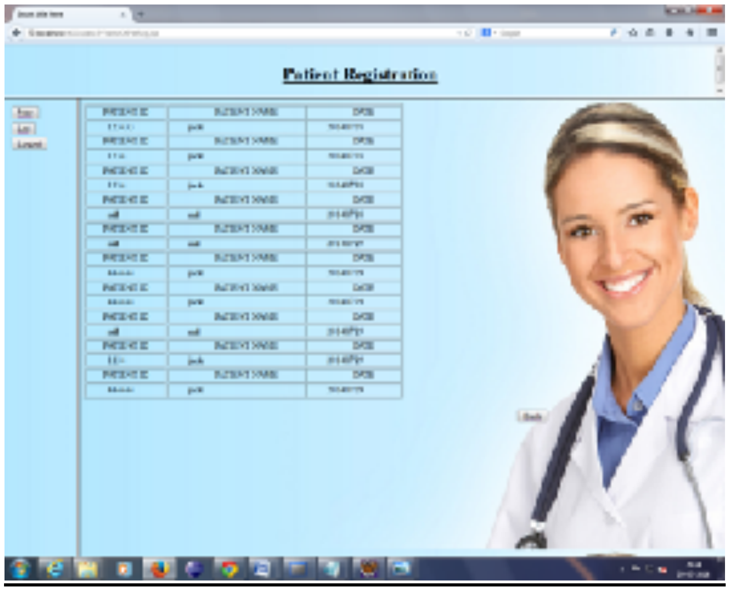


Registration Successful

Name: ANGELA

Back

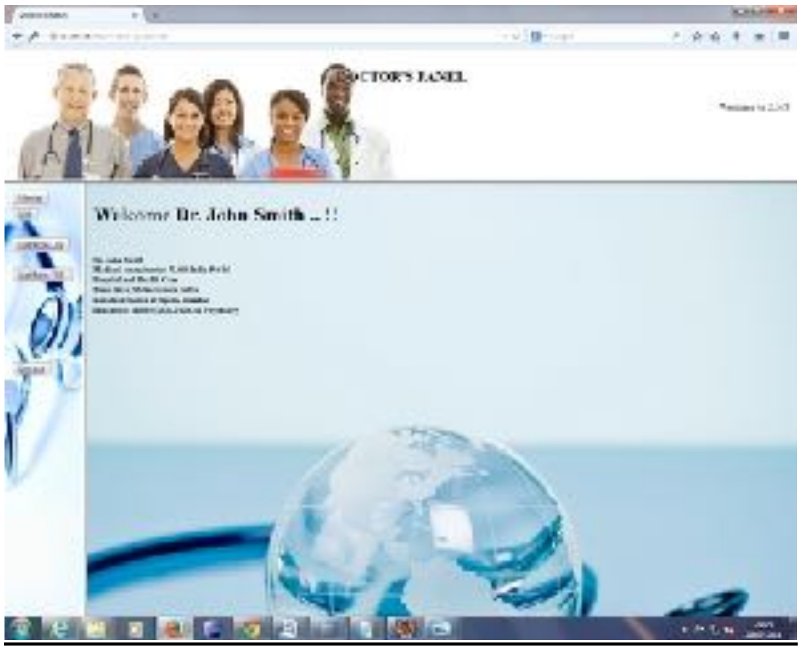
Registered Patient List :



Patient Registration

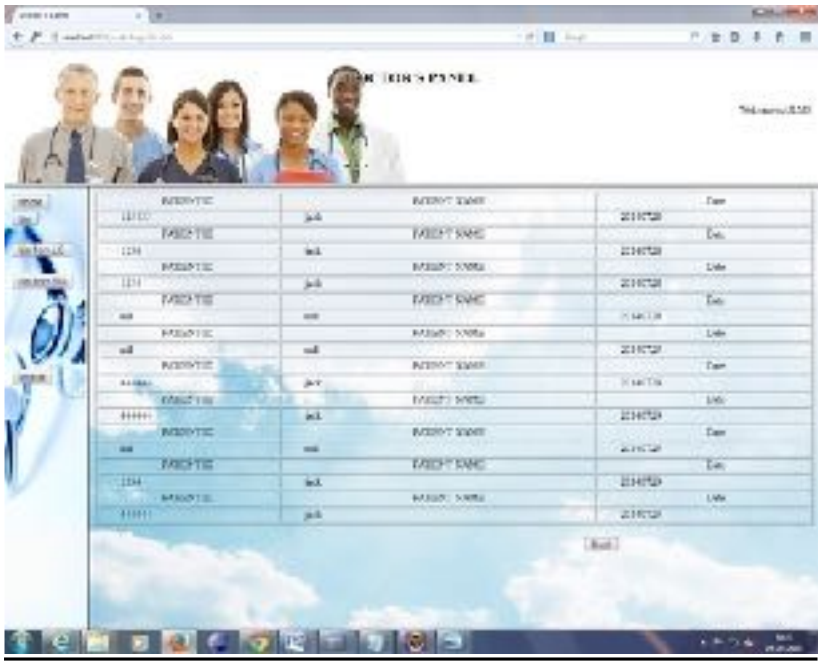
PATIENT ID	PATIENT NAME	DOB
[illegible]	[illegible]	[illegible]

Doctor’s Home Page :



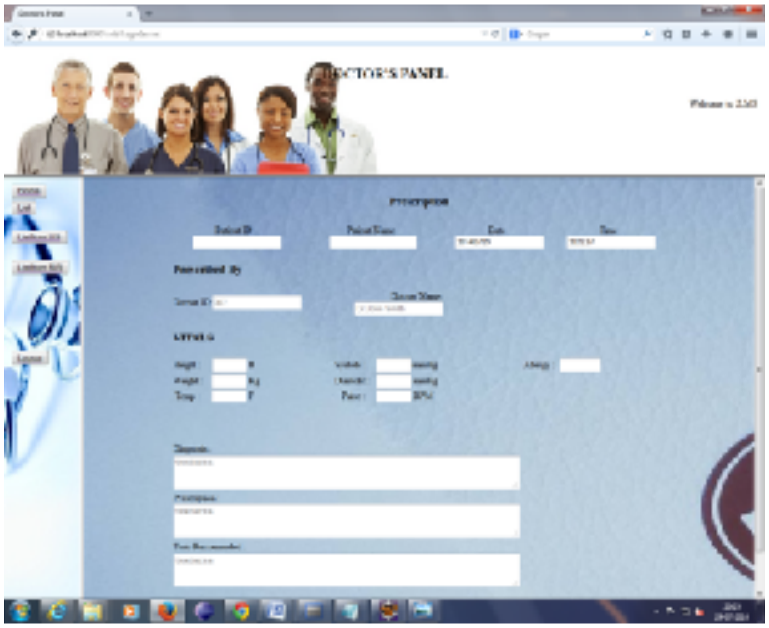
Welcome Dr. John Smith ..!!

Doctor’s Patient List Page :



PATIENT ID		PATIENT NAME	Date
[illegible]	jack		[illegible]
PATIENT ID		PATIENT NAME	Date
1234	jack		[illegible]
PATIENT ID		PATIENT NAME	Date
1234	jack		[illegible]
PATIENT ID		PATIENT NAME	Date
[illegible]	[illegible]		[illegible]
PATIENT ID		PATIENT NAME	Date
[illegible]	[illegible]		[illegible]
PATIENT ID		PATIENT NAME	Date
[illegible]	[illegible]		[illegible]
PATIENT ID		PATIENT NAME	Date
[illegible]	jack		[illegible]
PATIENT ID		PATIENT NAME	Date
[illegible]	[illegible]		[illegible]
PATIENT ID		PATIENT NAME	Date
1234	jack		[illegible]
PATIENT ID		PATIENT NAME	Date
[illegible]	jack		[illegible]

Patient Prescription Form :



DOCTOR'S PANEL

Welcome [illegible]

Prescription

Patient ID | Patient Name | Date | Time

Prescribed By

Doctor ID | Doctor Name

Vitals

Height ft | Systolic mmHg | Allergy

Weight Kg | Diastolic mmHg

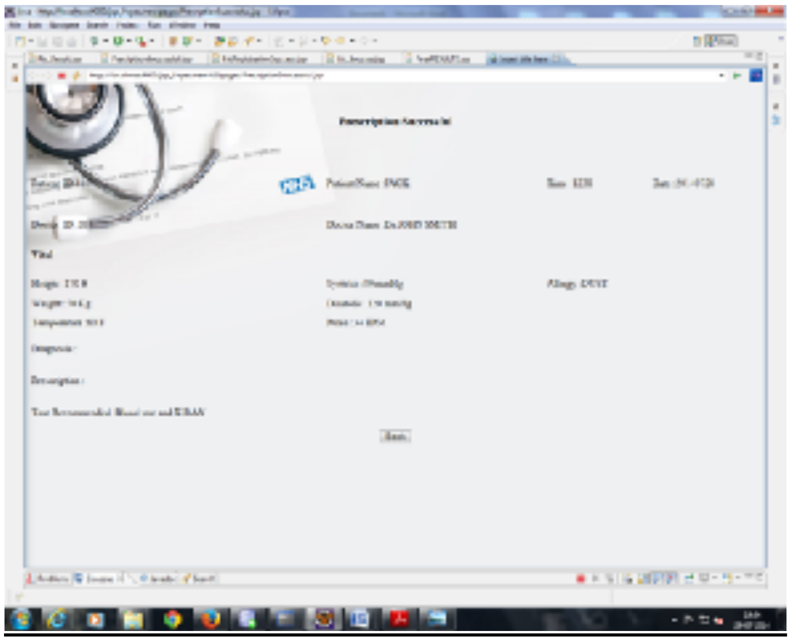
Temp F | Pulse BPM

Diagnosis

Prescription

Test Recommended

Doctor's View Prescription Page :



Prescription Successful

Patient ID [illegible] | Patient Name: [illegible] | Time: [illegible] | Date: [illegible]

Doctor ID: [illegible] | Doctor Name: Dr.JOHN SMITH

Vital

Height: [illegible] | Systolic: [illegible] | Allergy: DUST

Weight: [illegible] | Diastolic: [illegible] mmHg

Temperature: [illegible] | Pulse: [illegible] BPM

Diagnosis :

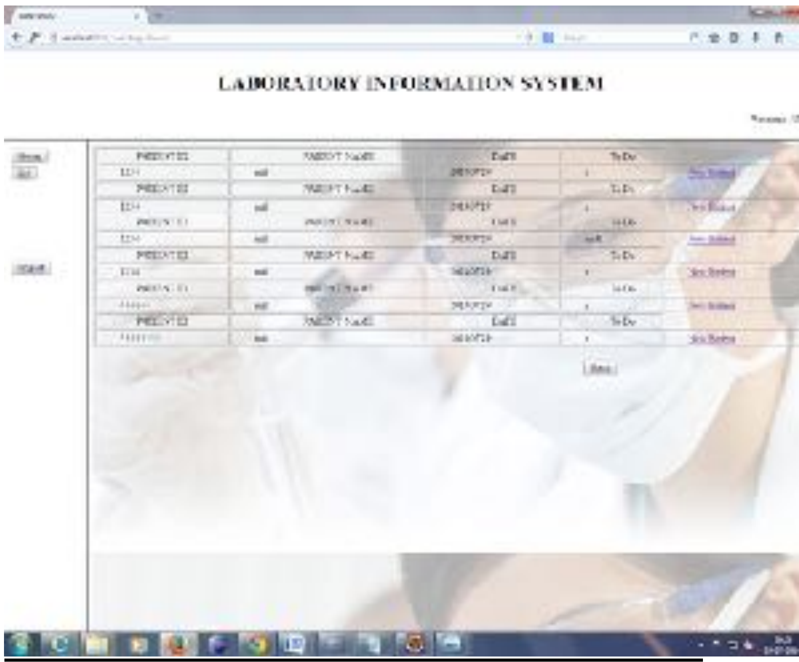
Prescription :

Test Recommended: [illegible]

Back

LIS Home Page :

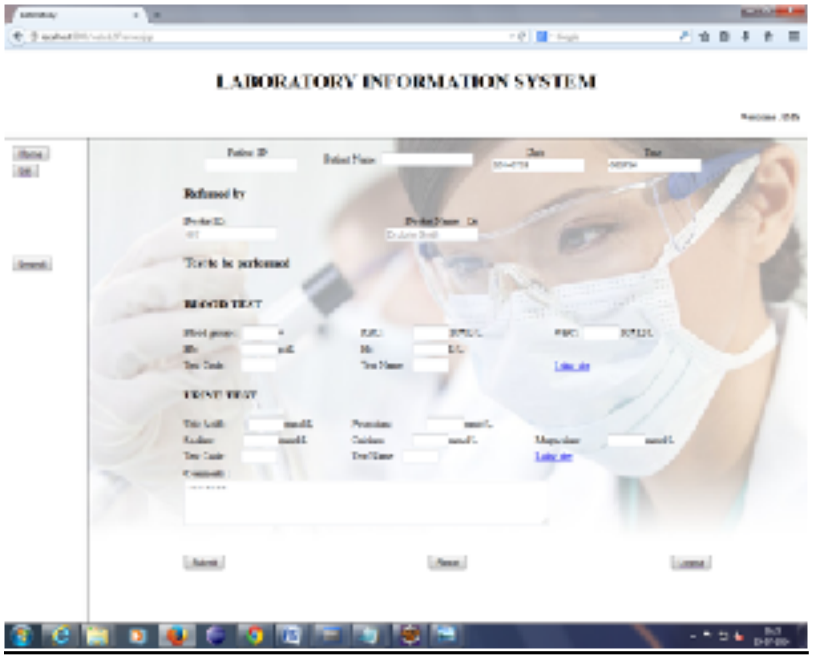
LIS List Of Orders :



LABORATORY INFORMATION SYSTEM

PATIENT ID	PATIENT NAME	DATE	To Do	
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
PATIENT ID	PATIENT NAME	DATE	To Do	
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
PATIENT ID	PATIENT NAME	DATE	To Do	
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
PATIENT ID	PATIENT NAME	DATE	To Do	
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
PATIENT ID	PATIENT NAME	DATE	To Do	
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
PATIENT ID	PATIENT NAME	DATE	To Do	
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Blood / Urine Test Form :



LABORATORY INFORMATION SYSTEM

Patient ID [illegible] Patient Name [illegible] Date [illegible] Time [illegible]

Referred by

Doctor ID [illegible] Doctor Name [illegible]

Tests to be performed

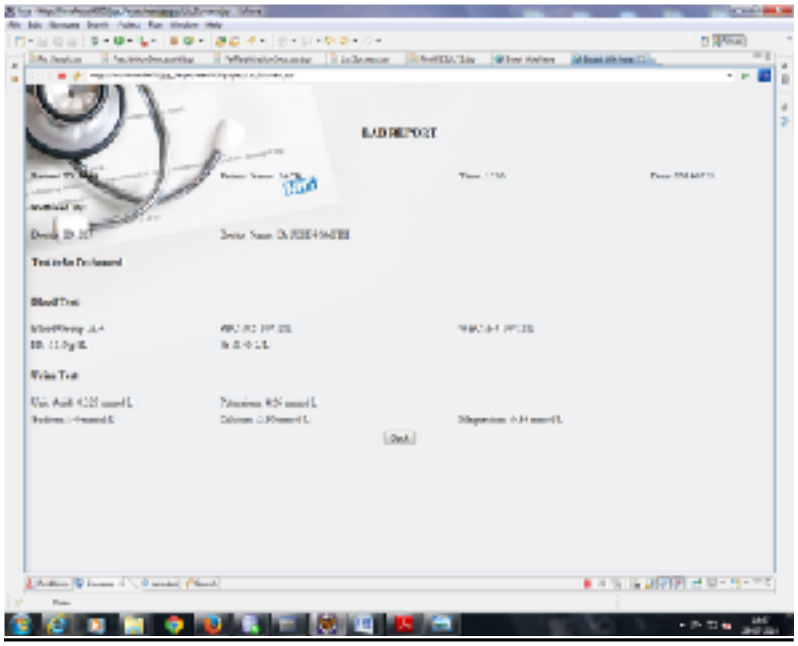
BLOOD TEST

Blood group [illegible] RBC [illegible] WBC [illegible]
Hb [illegible] Plt [illegible]
Test Code [illegible] Test Name [illegible]

URINE TEST

Uric Acid [illegible] Potassium [illegible]
Sodium [illegible] Calcium [illegible] Magnesium [illegible]
Test Code [illegible] Test Name [illegible]
Comments :

LIS View Report Page :



LAB REPORT

Patient ID [illegible] Patient Name [illegible] Time [illegible] Date [illegible]

Referred by

Doctor ID [illegible] Doctor Name [illegible]

Tests to be Performed

Blood Test

Blood Group [illegible] RBC [illegible] WBC [illegible]
Hb [illegible] Plt [illegible]

Urine Test

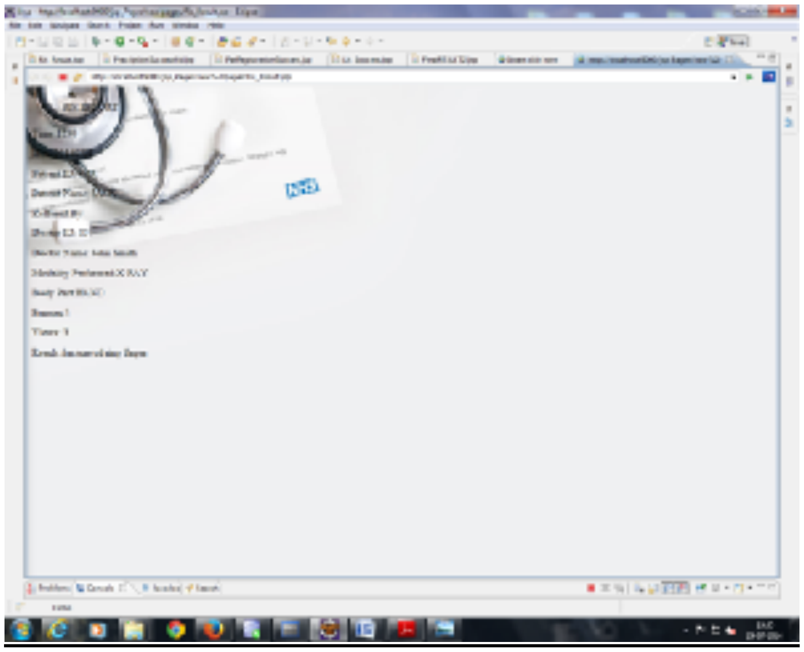
Uric Acid [illegible] Potassium [illegible]
Sodium [illegible] Calcium [illegible] Magnesium [illegible]

RIS Home Page :

RIS List Of Orders :

MRI / XRAY Form :

RIS View Report Page :



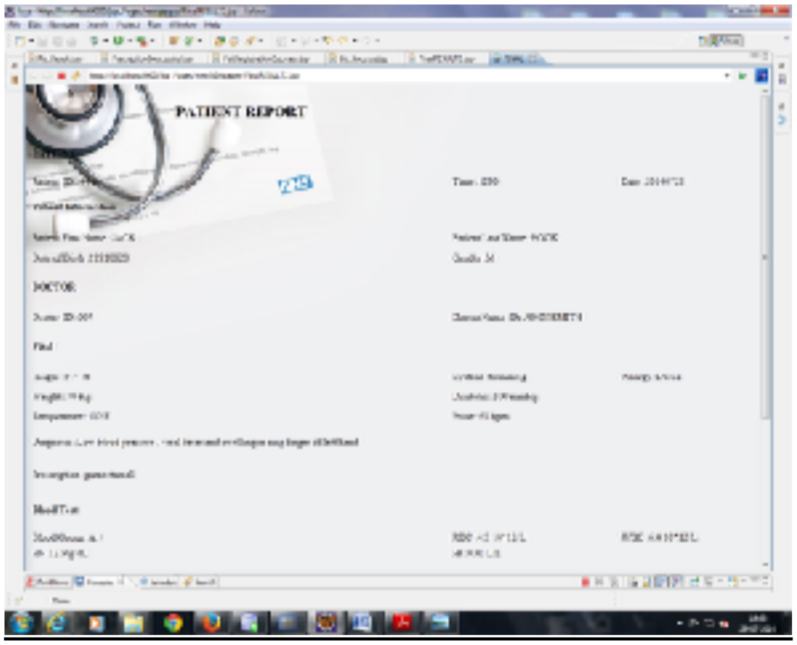
[illegible]

Doctor Name: John Smith

Modality Performed:X RAY

[illegible]

Patient Report :



PATIENT REPORT

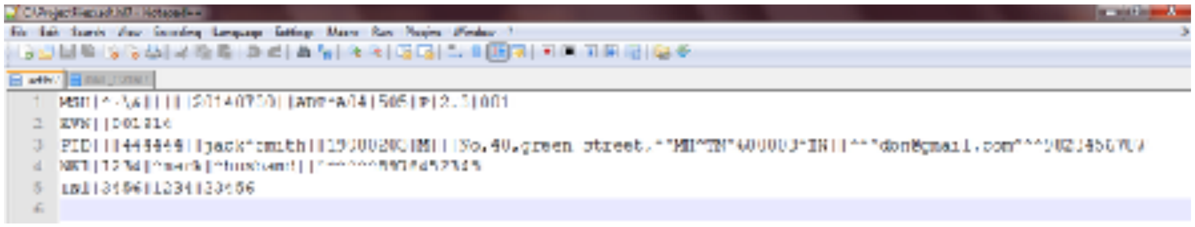
[illegible]

DOCTOR

[illegible]

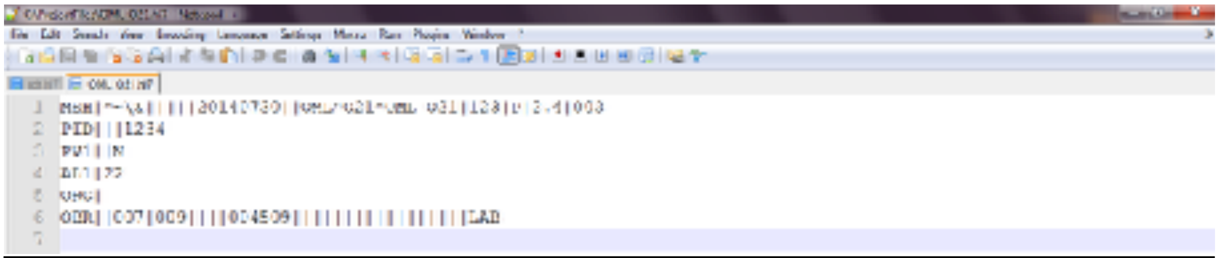
Sample ADT Messages :

ADT^A04^ADT_A04 HL7 message:



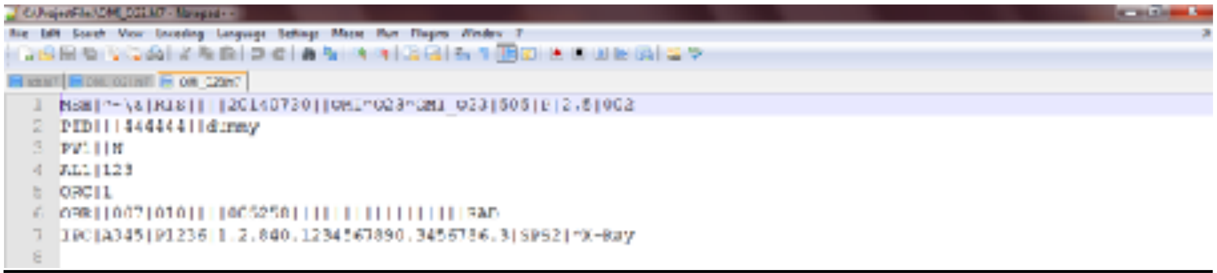
```
MSH|^~\&|||||20140730||ADT^A04|505|P|2.5|001
EVN||201314
PID|||444444||jack^smith||19900203|M|||No.40,green street,^"MH"^TN"400003^IN||^^^don@gmail.com^^^[illegible]
NK1|1|23d|^mark|^husband||^^^^^9874523645
IN1|3456|1234|123456
```

OML^O21^OML_O21 HL7 message:



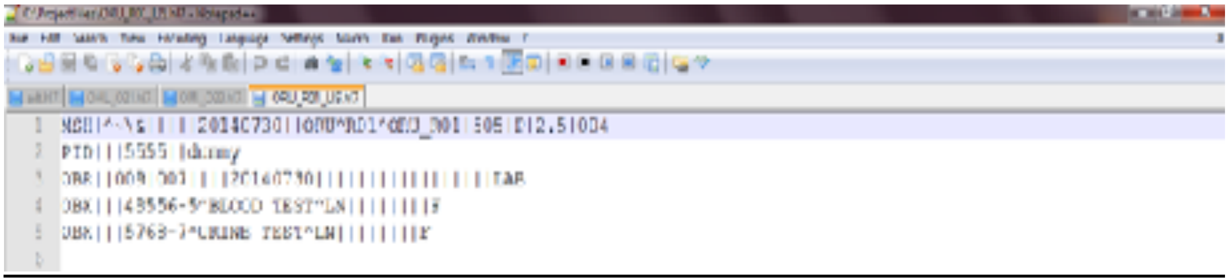
```
MSH|^~\&|||||20140730||OML^O21^OML_O21|128|P|2.4|003
PID|||1234
PV1||N
BLT|72
ORC|
OBR||C07|009|||024509||||||||||||||LAB
```

OMI^O23^OMI_O23 HL7 message:



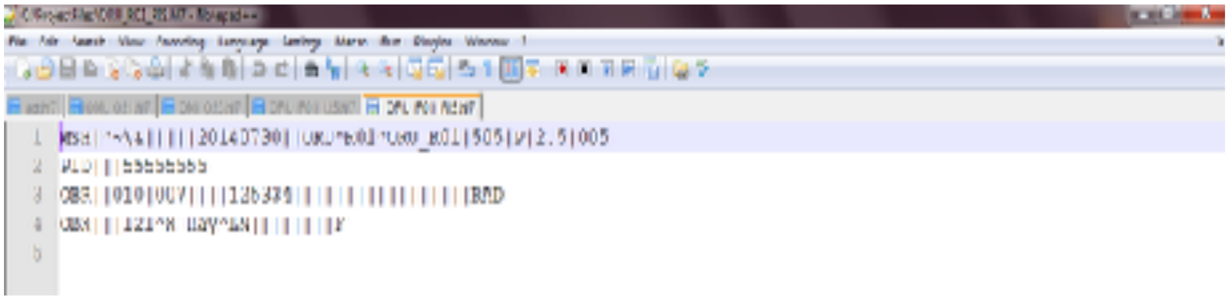
```
MSH|^~\&|RIS|||20140730||OMI^O23^OMI_O23|505|P|2.5|002
PID|||44444||dummy
PV1||N
AL1|123
ORC|1
OBR||007|010|||0C5250||||||||||||||RAD
IPC|A345|P1236|1.2.840.1234567890.3456786.3|SPS2|^X-Ray
```

<u>ORU^R01^ORU_R01 HL7 message from LIS:</u>



```
MSH|^~\&||||20140730||ORU^R01^ORU_R01|505|P|2.5|004
PID|||5555||dummy
OBR||009|001|||20140730||||||||||||||LAB
OBX|||43556-5^BLOOD TEST^LN|||||||F
OBX|||5763-7^URINE TEST^LN|||||||F
```

<u>ORU^R01^ORU_R01 HL7 message from RIS:</u>



```
MSH|^~\&||||20140730||ORU^R01^ORU_R01|505|P|2.5|005
PID|||55555555
OBR||010|007||||126378|||||||||||||RAD
OBX|||121^X RAY^LN|||||||F
```

3.3.1. User Characteristics :

3..3.1. User Characteristics :

The users are only those clients that are connected to Internet connection.

Every user should be comfortable with computer and Internet Browsing. He must have basic knowledge of filling

forms and uploading.

3.3.2. Performance Requirements :

Functional and properly integrated system installation is presumed.

3.3.3. Design Constraints :

It is assumed that the data entered is meaningful.

The database server will be up and running all the time (24 hrs) so that the client can use application any time.

3.4. Software System Attributes :

3.4.1. Reliability :

When user wants to call the system over a given period of time, the system should correctly deliver services as expected by the user. The reliability of the system shall be good if it delivers services as specified. Otherwise, reliability is bad and it shall produce unexpected output. So, program should be customized according to the situation.

3.4.2. Security :

The system should resist accidental or deliberate intrusions, when users operate on the system. If the system should not resist accidental or deliberate intrusions, then important data – such as patient report, medical images, username, passwords etc. – which belongs to user can be stolen by hacker. Thus, security of the system shall be low and trust of users shall be ruined. So, security of the system, data encryption, access control is very important for users.

3.4.3. Maintainability :

When the system is used, new requirements may emerge. When these requirements are emerged, the system should be changeable to accommodate these requirements for maintaining the usefulness of the system. If the system is not maintainable, then the system cannot be modified for new requirements. In this situation, a new system should be developed to provide for new requirements. The maintainability is important to avoiding unnecessary cost. The Object Oriented Features of Java makes it easy to maintain the code and functionalities.

3.4.4. Availability :

Server will be in working state for 24 x 7. When the system has any request at any given time, system should be available, it should be up and running and able to deliver useful service at this time. The availability of the system shall be good if it delivers services when it is requested. Otherwise, if requests are not responded at any given time then it implies bad availability.

3.4.5. Portability :

Since the application is developed solely using Java and Oracle database, which are portable on any Operating System, the application also can be ported on any platform. The application can be accessed by a system of any configuration provided that it has LAN connectivity.

4.FUTURE ENHANCEMENTS

1. **Evidence Based Medicine System For Doctor/Students.**

2. **Add DICOM Standards [include WADO].**

3. **Add Scheduling System To All The Modules.**

4. **Add PACS Storage System For DICOM Storage.**

5. **Integrate CDACs/Any Other Telemedicine System For Doctor/Patient Reach.**

5.Bibliography & Web References

http://www.wikipedia.com

http://www.w3schools.com

http://www.oracle.org

http://www.google.com

http://medinfo.cdac.in/repo/docs/tut/HL7%20v2.1/Homepage.html

https://medinfo.cdac.in/repo/docs/tut/DCM%20v2.1/Homepage.html

http://docs.oracle.com/javase/7/docs/api/

http://hl7api.sourceforge.net/

http://www.hl7.org/implement/standards/product_matrix.cfm?Family=EHR

http://search.loinc.org/