

Hospital Management System

Manoj Barman
Enrollment No: 692758
NIELIT A Level
PROJECT

Table of Contents

Introduction of the Project:	5
Objective of the Project:	7
Tools / Environment Used:	8
IDE Used (Eclipse IDE for Java EE Developers):	8
Web Framework: Java Play Framework.....	9
Java Platform, Enterprise Edition (Java EE) :	10
Programming Language (Java) :	11
EXTENSIBLE MARKUP LANGUAGE (XML).....	13
DIA – diagraming tool.....	13
NClass – UML diagraming tool	14
Database - PostgreSQL.....	15
pgAdmin Database Editor	17
Bootstrap (Front End Framework).....	18
Hibernate ORM	19
Summary of technologies used	20
Analysis of the Project:	20
Hospital User GUI.....	21
Login Interface.....	22
Patient Info Interface.....	22
Appointment Management Interface	22
Payment Interface	22
Report Management Interface.....	23
Resource Management Interface	23
Hospital User Service:	23
Feature Usage Logger	23
Usage Timer	23
Access rights controller.....	24
Connectivity controller	24

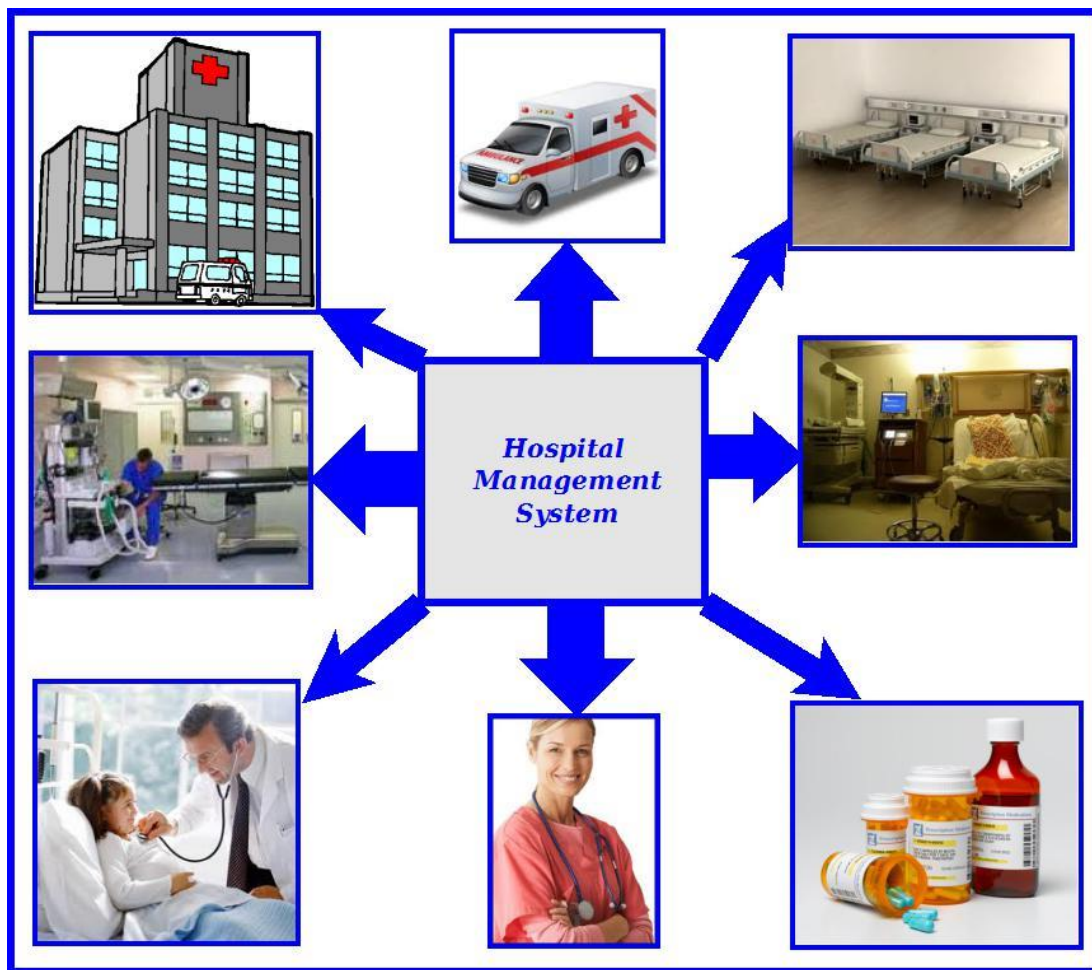
Hospital Controller Engine Interactor	24
Hospital Controller GUI	24
Login Interface.....	24
Access Right Interface.....	24
Billing Interface	24
Medicine Stock & Price Info browser	24
Patient Management Interface	25
Resource Management Interface.....	25
Appointment Management Interface	25
Report Interface	25
Security Management Interface.....	26
Video Surveillance Interface	26
Controller Engine Interactor	26
Hospital Controller Engine	26
Engine controller:	26
GUI Interactor:	26
Database Controller:.....	26
Information Controller:.....	26
Patient Controller:	26
Security Controller:.....	26
Usage Controller:.....	26
Access Controller:.....	27
Employee Controller:.....	27
Search Engine:	27
Billing Handler:	27
Connectivity Controller:.....	27
Hospital Management System Database.....	28
Hardware & Software Requirements for the application to run	29
DESIGN DOCUMENT:	30

Class Diagram	30
Data flow diagram	32
DFD-level 0	32
DFD level 1	33
DFD level 2	34
DFD - level 3	34
Gyant Chart.....	35
PERT Chart.....	35
Tracking Gyantt	36
E-R Diagram:.....	36
The user interface:	37
PROGRAMME CODE:	39
Source Code Hierarchy	39
Source Code	40
TESTING	84
UNIT TESTCASES:	84
INTEGRATION TESTCASES:	85
SYSTEM TESTCASES:	86
UNIT TEST RESULT:	87
INTEGRATION TEST RESULT:	88
SYSTEM TEST RESULT:.....	89
GUI Screenshots:	90
Organization	99
Future Scope and Further enhancement of the Project:.....	100
BIBLOGRAPHY:.....	101

Introduction of the Project:

Hospital Management System is meant for managing and maintaining the activities of a Hospital. Hospital management system helps to manage patients, doctors and hospital resources. It helps to utilize the hospital resources in an efficient manner.

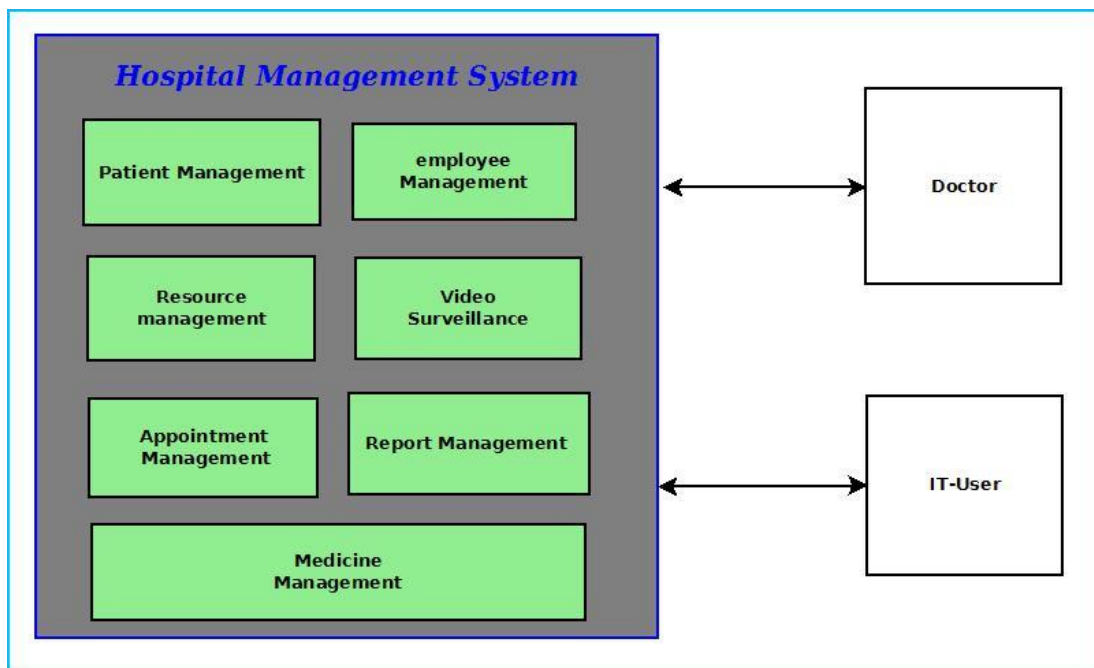
This software will be installed on every machine of the Hospital and one of the machines will act as a controller. The IT admin of the Hospital uses the Hospital controller machine to control other machines. Following diagram displays the interaction between Hospital controller and the user machines.



The main features of this software are given below:

- Resource Usage tracking
- Patient management.
- Patient Identity info saving
- Security management
- Medicine stock & pricing information query.
- Bill preparation
- Monthly / daily report
- Supply management & payment tracking.
- Centralized Report Printing Facility
- Video Surveillance.

The feature and user of the Hospital Management System are displayed in the diagram below.



The Hospital management software will be used by the employee of the Hospital to control the Hospital and the doctors have to use the Hospital management software interface to access/use patient related information.

Objective of the Project:

The objectives of this project are given below:

- Hospital management system will provide a software solution for managing Hospital more efficiently and effectively.
- It will enable Hospital administrator to implement an automatic Hospital management system by which a centralized system can keep track of activities happening in hospital.
- Hospital management system will track the Hospital patients, employees & resources.
- To provide a faster & more organized way of serving patients.
- Hospital management system will provide better ways for validating patient data and searching patient's previous treatment history which will replace cumbersome paper work with efficient computerized datasheets.
- By accomplishing this project I could learn new technologies like Java EE, , MySQL, XML and I am able to be involved with the complete software development lifecycle.

Tools / Environment Used:

This software will follow Object Oriented Programming Paradigm and use below mentioned areas.

IDE Used (Eclipse IDE for Java EE Developers):



In computer programming, Eclipse is an integrated development environment (IDE). It contains a base workspace and an extensible plug-in system for customizing the environment. Written mostly in Java, Eclipse can be used to develop applications. By means of various plug-ins, Eclipse may also be used to develop applications in other programming languages: Ada, ABAP, C, C++, COBOL, Fortran, Haskell, JavaScript, Lasso, Natural, Perl, PHP, Prolog, Python, R, Ruby (including Ruby on Rails framework), Scala, Clojure, Groovy, Scheme, and Erlang. It can also be used to develop packages for the software Mathematica. Development environments include the Eclipse Java development tools (JDT) for Java and Scala, Eclipse CDT for C/C++ and Eclipse PDT for PHP, among others.

The initial codebase originated from IBM VisualAge.[2] 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[3]). It was one of the first IDEs to run under GNU Classpath and it runs without problems under IcedTea.

Web Framework: Java Play Framework



Play is a high-productivity Java and Scala web application framework that integrates the components and APIs you need for modern web application development.

Play is based on a lightweight, stateless, web-friendly architecture and features predictable and minimal resource consumption (CPU, memory, threads) for highly-scalable applications thanks to its reactive model, based on Iteratee IO. Play is an open source web application framework, written in Scala and Java, which follows the model–view–controller (MVC) architectural pattern. It aims to optimize developer productivity by using convention over configuration, hot code reloading and display of errors in the browser.

Support for the Scala programming language has been available since version 1.1 of the framework. In version 2.0, the framework core was rewritten in Scala. Build and deployment was migrated to SBT, and templates use Scala instead of Groovy.

From other Java frameworks:

- Stateless: Play 2 is fully RESTful - there is no Java EE session per connection.
- Integrated unit testing: JUnit and Selenium support is included in the core.
- API comes with most required elements built-in.
- Static methods: all controller entry points are declared as static (or equivalently, in Scala, functions on Scala objects). After requests were made for this to be customisable, Play 2.1 now supports other styles of controllers, so controllers need not be static/Scala objects; however, this is still the default.
- Asynchronous I/O: due to using JBoss Netty as its web server, Play can service long requests asynchronously rather than tying up HTTP threads doing business logic like Java EE frameworks that don't use the asynchronous support offered by Servlet 3.0.[16]
- Modular architecture: like Rails and Django, Play comes with the concept of modules.
- Native Scala support: Play 2 uses Scala internally, but also exposes both a Scala API, and a Java API that is deliberately slightly different to fit in with Java conventions, and Play is completely interoperable with Java.

Java Platform, Enterprise Edition (Java EE) :



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.

Java Platform, Enterprise Edition (Java EE) is the standard in community-driven enterprise software. Java EE is developed using the Java Community Process, with contributions from industry experts, commercial and open source organizations, Java User Groups, and countless individuals. Each release integrates new features that align with industry needs, improves application portability, and increases developer productivity. Today, Java EE offers a rich enterprise software platform, and with over 20 compliant Java EE 6 implementations to choose from, low risk and plenty of options.

Java Platform, Enterprise Edition 7 (Java EE 7) offers new features that enhance HTML5 support, increase developer productivity, and further improves how enterprise demands can be met. Java EE 7 developers will write less boilerplate code, have better support for the latest Web applications and frameworks, and gain access to enhanced scalability and richer, simpler functionality. Enterprises will benefit from new features that enable portable batch processing and improved scalability.

Programming Language (Java) :



Java is a computer programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that code that runs on one platform does not need to be recompiled to run on another. Java applications are typically compiled to bytecode (class file) that can run on any Java virtual machine (JVM) regardless of computer architecture. Java is, as of 2014, one of the most popular programming languages in use, particularly for client-server web applications, with a reported 9 million developers.

Java was originally developed by James Gosling at Sun Microsystems (which has since merged into Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++, but it has fewer low-level facilities than either of them.

The original and reference implementation Java compilers, virtual machines, and class libraries were developed by Sun from 1991 and first released in 1995. As of May 2007, in compliance with the specifications of the Java Community Process, Sun relicensed most of its Java technologies under the GNU General Public License. Others have also developed alternative implementations of these Sun technologies, such as the GNU Compiler for Java (bytecode compiler), GNU Classpath (standard libraries), and IcedTea-Web (browser plugin for applets).

There were five primary goals in the creation of the Java language:

- It should be "simple, object-oriented and familiar"
- It should be "robust and secure"
- It should be "architecture-neutral and portable"
- It should execute with "high performance"
- It should be "interpreted, threaded, and dynamic"

OpenJDK is another notable Java SE implementation that is licensed under the GPL. The implementation started when Sun began releasing the Java source code under the GPL. As of Java SE 7, OpenJDK is the official Java reference implementation.

The goal of Java is to make all implementations of Java compatible. Historically, Sun's trademark license for usage of the Java brand insists that all implementations be "compatible". This resulted in a legal dispute with Microsoft after Sun claimed that the Microsoft implementation did not support RMI or JNI and had added platform-specific features of their own. Sun sued in 1997, and in 2001 won a settlement of US\$20 million, as well as a court order enforcing the terms of the license from Sun. As a result, Microsoft no longer ships Windows with Java.

Platform-independent Java is essential to Java EE, and an even more rigorous validation is required to certify an implementation. This environment enables portable server-side applications.

EXTENSIBLE MARKUP LANGUAGE (XML)

Extensible Markup Language (XML) is a simple, very flexible text format derived from SGML (ISO 8879). Originally designed to meet the challenges of large-scale electronic publishing, XML is also playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere.

Extensible Markup Language (XML) is a set of rules for encoding documents in machine-readable form. It is defined in the XML 1.0 Specification produced by the W3C, and several other related specifications, all gratis open standards.

XML's design goals emphasize simplicity, generality, and usability over the Internet.[6] It is a textual data format with strong support via Unicode for the languages of the world. Although the design of XML focuses on documents, it is widely used for the representation of arbitrary data structures, for example in web services.

Many application programming interfaces (APIs) have been developed that software developers use to process XML data, and several schema systems exist to aid in the definition of XML-based languages.

As of 2009, hundreds of XML-based languages have been developed,[7] including RSS, Atom, SOAP, and XHTML. XML-based formats have become the default for most office-productivity tools, including Microsoft Office (Office Open XML), OpenOffice.org (OpenDocument), and Apple's iWork.

System.Xml is the core XML manipulation technology for .NET languages.

DIA – diagramming tool



Dia is a GTK+ based diagram creation program for GNU/Linux, Mac OS X, Unix, and Windows. Dia is roughly inspired by the commercial Windows program 'Visio,' though more geared towards informal diagrams for casual use. It can be used to draw many different kinds of diagrams. It currently has special objects to help draw entity relationship diagrams, UML diagrams, flowcharts, network diagrams, and many other diagrams. It is also possible to add support for new shapes by writing simple XML files, using a subset of SVG to draw the shape.

It can load and save diagrams to a custom XML format (gzipped by default, to save space), can export diagrams to a number of formats, including EPS, SVG, XFIG, WMF and PNG, and can print diagrams (including ones that span multiple pages).

NClass – UML diagraming tool



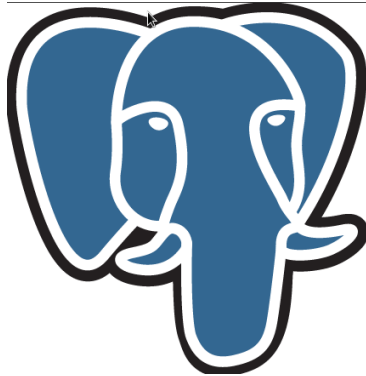
NClass is a free tool to easily create UML class diagrams with full C# and Java language support. The user interface is designed to be simple and user-friendly for easy and fast development. Properties, enums, delegates and other language specific elements are fully supported with strict syntactical and semantical verification.

Design your application with just a few clicks - the main goal is to provide a simple but powerful class designer that is very intuitive to use. Diagram styles help you to create professional looking diagrams, just like in Visual Studio or other commercial products. Furthermore, you can generate code from your models or you can also import classes from existing .NET assemblies.

Features

- Full C# and Java support with many language specific elements
- Simple and easy to use user interface
- Inline class editors with syntactic parsers for easy and fast editing
- Source code generation
- Reverse engineering from .NET assemblies (thanks to Malte Ried)
- Configurable diagram styles
- Printing / saving to image
- Multilingual user interface
- Mono support for non-Windows users

Database - PostgreSQL



PostgreSQL
the world's most advanced open source database

PostgreSQL is a powerful, open source object-relational database system. It has more than 15 years of active development and a proven architecture that has earned it a strong reputation for reliability, data integrity, and correctness. It runs on all major operating systems, including Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64), and Windows. It is fully ACID compliant, has full support for foreign keys, joins, views, triggers, and stored procedures (in multiple languages). It includes most SQL:2008 data types, including INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL, and TIMESTAMP. It also supports storage of binary large objects, including pictures, sounds, or video. It has native programming interfaces for C/C++, Java, .Net, Perl, Python, Ruby, Tcl, ODBC, among others, and exceptional documentation.

An enterprise class database, PostgreSQL boasts sophisticated features such as Multi-Version Concurrency Control (MVCC), point in time recovery, tablespaces, asynchronous replication, nested transactions (savepoints), online/hot backups, a sophisticated query planner/optimizer, and write ahead logging for fault tolerance. It supports international character sets, multibyte character encodings, Unicode, and it is locale-aware for sorting, case-sensitivity, and formatting. It is highly scalable both in the sheer quantity of data it can manage and in the number of concurrent users it can accommodate. There are active PostgreSQL systems in production environments that manage in excess of 4 terabytes of data. Some general PostgreSQL limits are included in the table below.

Limit	Value
Maximum Database Size	Unlimited
Maximum Table Size	32 TB
Maximum Row Size	1.6 TB
Maximum Field Size	1 GB
Maximum Rows per Table	Unlimited
Maximum Columns per Table	250 - 1600 depending on column types
Maximum Indexes per Table	Unlimited

PostgreSQL prides itself in standards compliance. Its SQL implementation strongly conforms to the ANSI-SQL:2008 standard. It has full support for subqueries (including subselects in the FROM clause), read-committed and serializable transaction isolation levels. And while PostgreSQL has a fully relational system catalog which itself supports multiple schemas per database, its catalog is also accessible through the Information Schema as defined in the SQL standard. Data integrity features include (compound) primary keys, foreign keys with

restricting and cascading updates/deletes, check constraints, unique constraints, and not null constraints. It also has a host of extensions and advanced features. Among the conveniences are auto-increment columns through sequences, and LIMIT/OFFSET allowing the return of partial result sets. PostgreSQL supports compound, unique, partial, and functional indexes which can use any of its B-tree, R-tree, hash, or GiST storage methods. GiST (Generalized Search Tree) indexing is an advanced system which brings together a wide array of different sorting and searching algorithms including B-tree, B+-tree, R-tree, partial sum trees, ranked B+-trees and many others. It also provides an interface which allows both the creation of custom data types as well as extensible query methods with which to search them. Thus, GiST offers the flexibility to specify what you store, how you store it, and the ability to define new ways to search through it --- ways that far exceed those offered by standard B-tree, R-tree and other generalized search algorithms. GiST serves as a foundation for many public projects that use PostgreSQL such as OpenFTS and PostGIS. OpenFTS (Open Source Full Text Search engine) provides online indexing of data and relevance ranking for database searching. PostGIS is a project which adds support for geographic objects in PostgreSQL, allowing it to be used as a spatial database for geographic information systems (GIS), much like ESRI's SDE or Oracle's Spatial extension. Other advanced features include table inheritance, a rules systems, and database events. Table inheritance puts an object oriented slant on table creation, allowing database designers to derive new tables from other tables, treating them as base classes. Even better, PostgreSQL supports both single and multiple inheritances in this manner. The rules system, also called the query rewrite system, allows the database designer to create rules which identify specific operations for a given table or view, and dynamically transform them into alternate operations when they are processed. The events system is an interprocess communication system in which messages and events can be transmitted between clients using the LISTEN and NOTIFY commands, allowing both simple peer to peer communication and advanced coordination on database events. Since notifications can be issued from triggers and stored procedures, PostgreSQL clients can monitor database events such as table updates, inserts, or deletes as they happen.

pgAdmin Database Editor



pgAdmin is the most popular and feature rich Open Source administration and development platform for PostgreSQL, the most advanced Open Source database in the world. The application may be used on Linux, FreeBSD, Solaris, Mac OSX and Windows platforms to manage PostgreSQL 7.3 and above running on any platform, as well as commercial and derived versions of PostgreSQL such as Postgres Plus Advanced Server and Greenplum database.

pgAdmin is designed to answer the needs of all users, from writing simple SQL queries to developing complex databases. The graphical interface supports all PostgreSQL features and makes administration easy. The application also includes a syntax highlighting SQL editor, a server-side code editor, an SQL/batch/shell job scheduling agent, support for the Slony-I replication engine and much more. Server connection may be made using TCP/IP or Unix Domain Sockets (on *nix platforms), and may be SSL encrypted for security. No additional drivers are required to communicate with the database server.

pgAdmin is developed by a community of PostgreSQL experts around the world and is available in more than a dozen languages. It is Free Software released under the PostgreSQL License.

Bootstrap (Front End Framework)



Bootstrap is a free collection of tools for creating websites and web applications. It contains HTML and CSS-based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions. In June 2014 it was the No.1 project on GitHub with 69,000+ stars and 25,000+ forks,[1] with a user base including MSNBC and NASA.

Bootstrap is compatible with the latest versions of all major browsers. It gracefully degrades when used on older browsers such as Internet Explorer 8.

Since version 2.0 it also supports responsive web design. This means the layout of web pages adjusts dynamically, taking into account the characteristics of the device used (desktop, tablet, mobile phone). Starting with version 3.0, Bootstrap adopted a mobile first design philosophy, emphasizing responsive design by default.

Bootstrap is open source and available on GitHub. Developers are encouraged to participate in the project and make their own contributions to the platform. Recently, community members have translated Bootstrap's documentation into various languages, including Chinese, Spanish and Russian.

Bootstrap is modular and consists essentially of a series of LESS stylesheets that implement the various components of the toolkit. A stylesheet called bootstrap.less includes the components stylesheets. Developers can adapt the Bootstrap file itself, selecting the components they wish to use in their project.

Adjustments are possible to a limited extent through a central configuration stylesheet. More profound changes are possible by the LESS declarations.

The use of LESS stylesheet language allows the use of variables, functions and operators, nested selectors, as well as so-called mixins.

Since version 2.0, the configuration of Bootstrap also has a special "Customize" option in the documentation. Moreover, the developer chooses on a form the desired components and adjusts, if necessary, the values of various options to their needs. The subsequently generated package already includes the pre-built CSS style sheet.

Grid system and responsive design comes standard with a 1170 pixel wide, grid layout. Alternatively, the developer can use a variable-width layout. For both cases, the toolkit has four variations to make use of different resolutions and types of devices: mobile phones,

portrait and landscape, tablets and PCs with low and high resolution. Each variation adjusts the width of the columns.

Hibernate ORM



Hibernate ORM (Hibernate in short) is an object-relational mapping library for the Java language, providing a framework for mapping an object-oriented domain model to a traditional relational database. Hibernate solves object-relational impedance mismatch problems by replacing direct persistence-related database accesses with high-level object handling functions. Hibernate is a free software that is distributed under the GNU Lesser General Public License. Hibernate's primary feature is mapping from Java classes to database tables (and from Java data types to SQL data types). Hibernate also provides data query and retrieval facilities. It generates SQL calls and relieves the developer from manual result set handling and object conversion. Applications using Hibernate are portable to supported SQL databases with little performance overhead. Mapping Java classes to database tables is accomplished through the configuration of an XML file or by using Java Annotations. When using an XML file, Hibernate can generate skeleton source code for the persistence classes. This is unnecessary when annotations are used. Hibernate can use the XML file or the annotations to maintain the database schema. Facilities to arrange one-to-many and many-to-many relationships between classes are provided. In addition to managing associations between objects, Hibernate can also manage reflexive associations where an object has a one-to-many relationship with other instances of its own type. Hibernate supports the mapping of custom value types. This makes the following scenarios possible:

- [Overriding the default SQL type that Hibernate chooses when mapping a column to a property.](#)
- [Mapping Java Enum to columns as if they were regular properties.](#)
- [Mapping a single property to multiple columns.](#)

Definition: Objects in a front-end application follow OOP principles, while objects in the back-end follow database normalization principles, resulting in different representation requirements. This problem is called "object-relational impedance mismatch". Mapping is a way of resolving the impedance mismatch problem. Mapping tells the ORM tool which java class objects an application is needed to be store in which table of database. Hibernate provides transparent persistence for Plain Old Java Objects (POJOs). The only strict requirement for a persistent class is a no-argument constructor, not necessarily public. Proper behavior in some applications also requires special attention to the equals() and hashCode() methods. Collections of data objects are typically stored in Java collection objects such as Set and List. Java generics, introduced in Java 5, are supported. Hibernate can be configured to lazy load associated collections. Lazy loading is the default as of Hibernate 3. Related objects can be configured to cascade operations from one to the other. For example, a parent Album object can be configured to cascade its save and/or delete operation to its child Track objects. This can reduce development time and ensure referential integrity. A dirty checking feature

avoids unnecessary database write actions by performing SQL updates only on the modified fields of persistent objects.

Summary of technologies used

Front End/ GUI Tools: Java Play Framework, Bootstrap HTML framework

Programming Language: Java, Java EE

Backend: PostgreSQL, postgres java Connector, XML , hibernate, jaxws

Internet Technologies: HTML, JavaScript

Networking Technologies: TCP/IP

Operating Systems: Windows XP, Windows 7

Applications: ERP application, Database Management System.

Analysis of the Project:

Hospital Management System caters different types of employees of hospital. There will be IT-Users like receptionist, data entry operator, doctor, nurses and IT-Admin like hospital superintendent. Hospital Management System will provide a cost effective & efficient software solution for managing everything about Hospital. Different components of Hospital Management System area are depicted below.

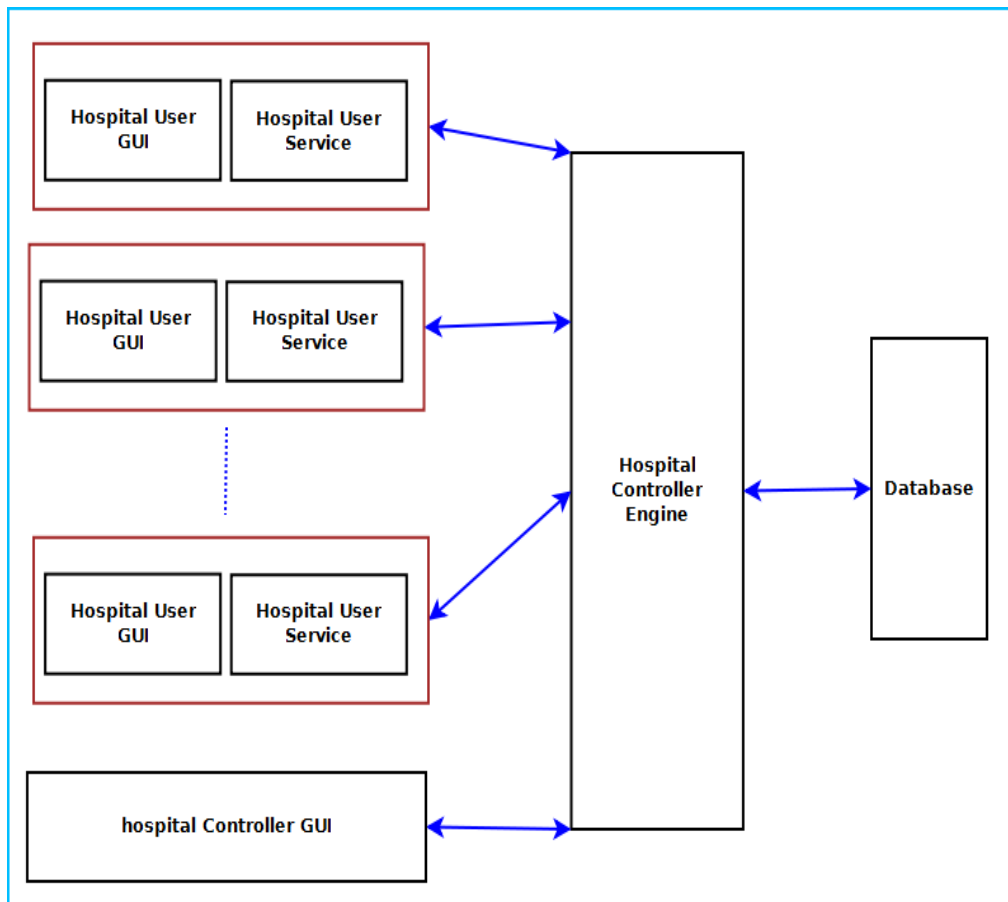


Figure: Components of Hospital Management System

Hospital Management System is divided into five main components. Such as:

- Hospital User GUI
- Hospital User Service
- Hospital Controller GUI
- Hospital Controller Engine
- Hospital Management System Database

Hospital User GUI & Service will be installed in IT-Users machine. Hospital controller GUI, engine & database will be installed in the IT-Admin's machine and user machines will be connected to the controller machine.

Hospital User GUI

Hospital User GUI will display User login, Patient information & billing information, information search, appointment interface and report management interface. It is divided into five modules. Such as:

- Login Interface
- Patient Info Interface
- Payment Interface

- Appointment Management Interface
- Resource Management Interface
- Report Management Interface

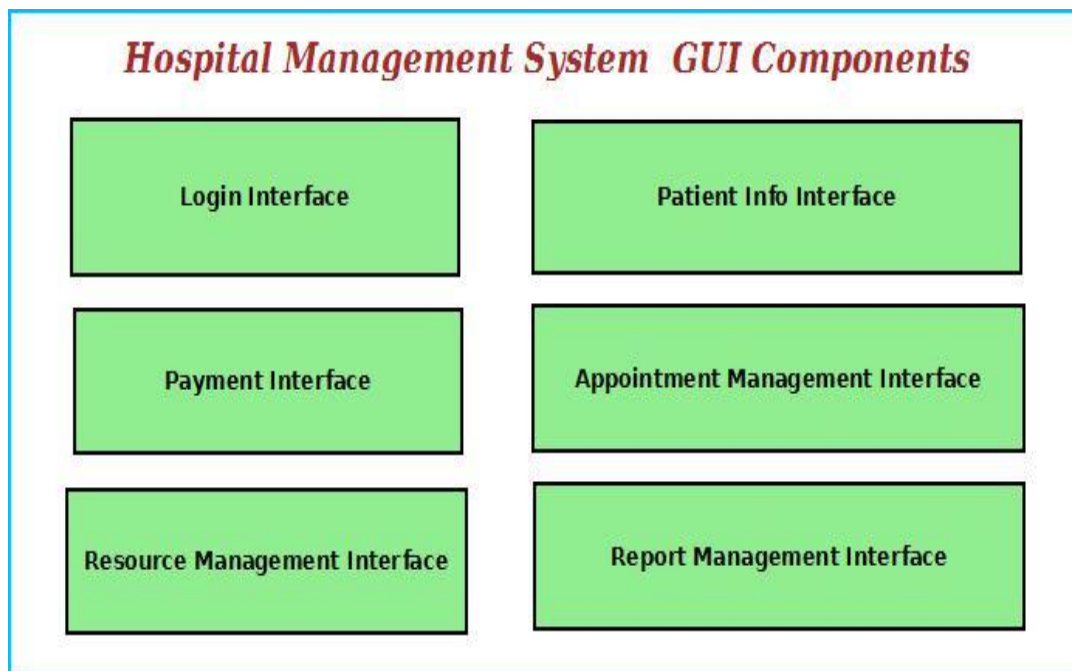


Figure: User GUI Components

Login Interface

Hospital User GUI can be used by variety of hospital employees. So user need to login using user id, password to access their registered account. It is mandatory to login to use the machine.

Patient Info Interface

This interface will allow IT-users to register new patients information, add new entries regarding patient current status. Patient Info Interface will have multiple forms to filled for adding required information.

It also allows printing in the centralized printer.

Appointment Management Interface

This interface will allow IT-user to search for appointment of a doctor of the Hospital and view fees information, time and location details.

Payment Interface

Payment interface enables bill generation for patients. IT-Users will inquire for services required by the patient. Then the Hospital Management System will search the fees and prepare the bill for printing. It will allow user to pay the bill by cash or credit/debit card.

Report Management Interface

This module will help to generate various reports about patients like diagnosis reports, release letter, medical certificates. IT-Users will use this interface to generate and print the reports on demand..

Resource Management Interface

This module will maintain information about available resources like operation theater, machines for use. Patients can book them prior to use. IT-Users will use this interface to book this.

Hospital User Service:

This is a windows service utility running on every user's machine to control users activity and access rights. This component is divided into five modules:

- Feature Usage Logger
- Usage Timer
- Access rights controller
- Connectivity controller
- Hospital Controller Engine Interactor

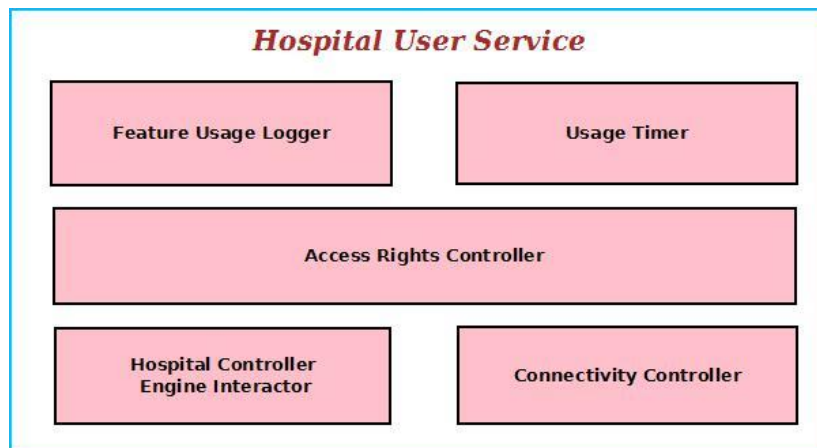


Figure: User Service Components

Feature Usage Logger

It logs the name of the Features and Resources used by the IT-user. It helps the Hospital Admin to realize the usage trend.

Usage Timer

This module logs the Users start time and runs a timer to display the elapsed time of the user. It also keeps the session information and remaining balance to avail next sessions.

Access rights controller

It controls the user's access rights like installing /uninstalling softwares, using external drives. User has to seek permission from the admin to perform certain actions. This module controls this kind of access right mechanism.

Connectivity controller

This module handles the internet connectivity and LAN connections in case of LAN Gaming. It tracks the user download and upload quantity and requests for billing if it exceeds the permitted amount.

Hospital Controller Engine Interactor

This module interacts with centralized Hospital controller engine to get certain information about user and perform actions accordingly.

Hospital Controller GUI

Hospital Controller GUI will display User information, usage & billing information, product information search, Product selection browser, security management interface, video surveillance window and Doctor account management browser. The Hospital can monitor and control whole Hospital from a single machine.

It is divided into eleven modules. Such as:

- Login Interface
- Access Right Interface
- Resource Management Interface
- Appointment Interface
- Billing Interface
- Medicine Stock & Price Info browser
- Patient Management Interface
- Report Interface
- Security Management Interface
- Video Surveillance Interface
- Controller Engine Interactor

Login Interface

This interface will allow the owner/admin to login and control other users. This will authenticate the admin and enable him to give access the admin features and approve user requests.

Access Right Interface

The users will ask for several access requests like installing new software, use external drive, and connect with other machine on LAN. The admin get this request as a popup and he can approve or reject them.

Billing Interface

This interface will display the usage of all the patients' available. It will also help generate bill for a patient's usage and purchase. This interface will allow viewing bulk purchase amounts, approvals for the higher management.

Medicine Stock & Price Info browser

This module will allow the IT-Admin to view the medicine stocks, price information.

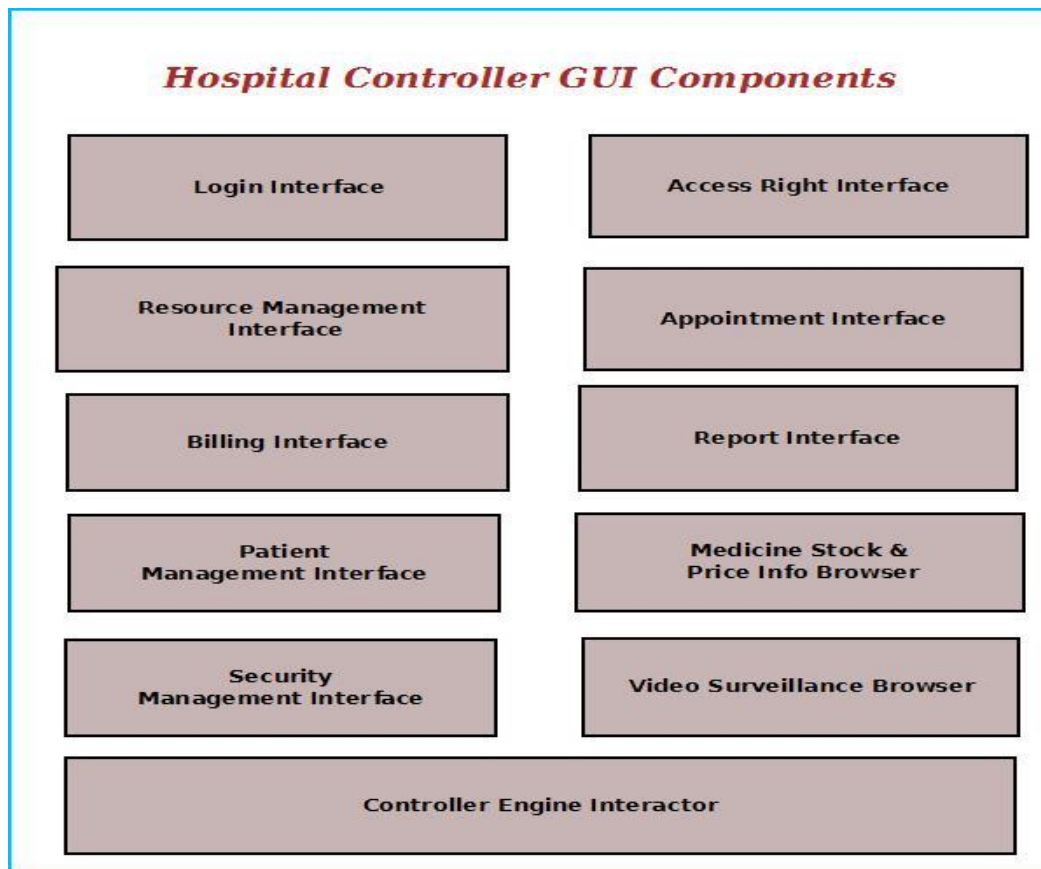


Figure: Controller GUI Components

Patient Management Interface

This interface will enable the admin to browse existing patients and add new patients. It will allow the admin to manage different aspects of patients of the Hospital.

Resource Management Interface

This interface will enable the admin to browse existing resources and approve resource allocation requests.

Appointment Management Interface

This interface will enable the admin to appointment of existing patients and resolve appointment conflicts.

Report Interface

This interface is for managing the report generation, centralized printer setup and to approve the print requests, prepare bill for the request.

Security Management Interface

Hospitals need to maintain the patient data, photo identity card number. This interface will allow managing security measures to taken about the patients. It will have a interface for entering the patient's security data, photo and scan the photo identity card of the user.

Video Surveillance Interface

This interface will display the video of specific areas of the Hospital and save the video for future reference. The admin can monitor from video surveillance window.

Controller Engine Interactor

This interface will interact with centralized controller engine and fetch the required data.

Hospital Controller Engine

Hospital Controller Engine controls the overall system. It provides logical and tactical solutions for managing the whole system. The Hospital Management System is divided into 12 divided modular components. Such as:

Engine controller

GUI Interactor

Database Controller

Information Controller

Employee Controller

Usage Handler

Access Controller

Connectivity Controller

Billing Handler

Search Engine

Patient Controller

Security Controller

Engine controller:

This controls the overall interaction between all the backend modules. It schedules the priority of the actions in case of overlapping.

GUI Interactor:

It interacts with the GUI and polls GUI calls. It exposes APIs and events for GUI to use. GUI Interactor helps Engine to maintain wrapper around the Engine modules so that the GUI can be ported to any other framework without much changes in Engine code.

Database Controller:

It controls the database interactions. It forms query to fetch information from the database. It also sends data to be saved in database for future use.

Information Controller:

This module processes patient and other information. It also archives the data to save space.

Patient Controller:

It keeps track of patient information. It has an algorithm for maintaining patient's treatment, disease history. Depending on those patient's care will be taken accordingly.

Security Controller:

It saves information for security measures. It can send it to the authority or police whenever required.

Usage Controller:

It controls the usage of the users depending on the balance available and informs the user about remaining time line.

Access Controller:

Access controller manages user login and features to be accessed by them. It will validate user's login and block access & inform management if any access violation attempts happen.

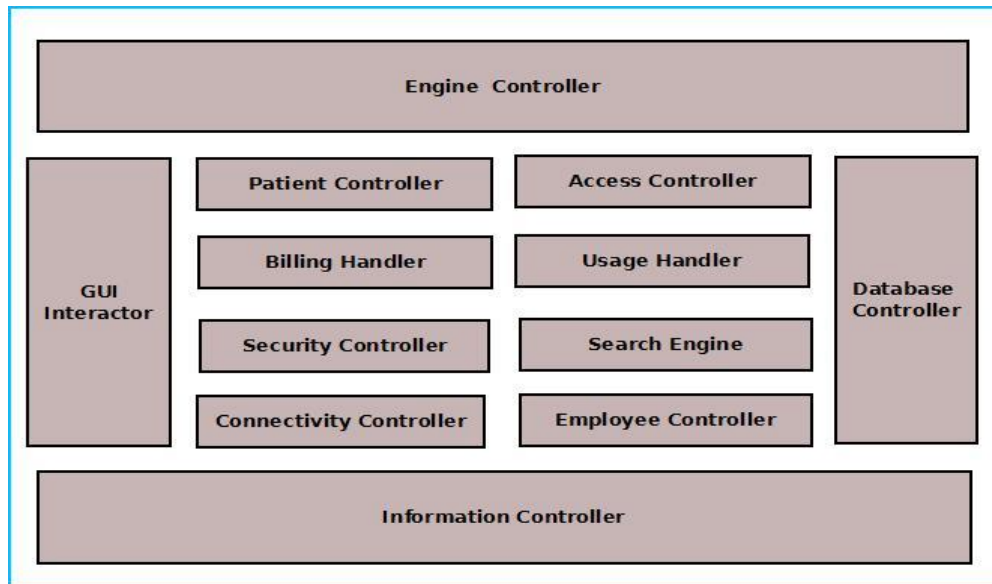


Figure: Components of Engine

Employee Controller:

This module will maintain the information about employees such as doctor, nurse, support stuff. It will save the information in database and retrieve them when required.

Search Engine:

Search engine enables rapid and efficient searching of data about doctors, products and suppliers. Search Engine prepares search indexes depending most accessed data. It improves the efficiency and performance of the software.

Billing Handler:

Billing Handler allows generating the bill for Patient's usage & purchases. It will get the pricing information from the database depending on user data and admin input about product type and quantity. It will add discounts, special offer and VAT amounts on the purchase amount and calculate the amount payable.

Connectivity Controller:

Web controller handles the interaction with Internet and LAN. It allows connecting to the internet depending users credit balance. It has the logic for connection handling between machines and controller.

Hospital Management System Database

Hospital Management System will maintain a centralized database for storing information. We will design a RDBMS to manage the database and engine interaction. It will have optimized design and archive older data to save space and increase performance.

Hardware & Software Requirements for the application to run

Hardware Requirements

Computer that has a 1.6GHz or faster processor

1 GB (32 Bit) or 2 GB (64 Bit) RAM

10 MB of available hard disk space

DVD-ROM Drive / USB Port

Webcam

GPS Trackers

NFC Id Cards

Hospital Cameras

Software Requirements

Windows XP (x86) with Service Pack 3 / Windows Vista (x86 & x64) with Service Pack 2 / Windows 7 (x86 & x64)

Microsoft .NET 4.0

DESIGN DOCUMENT:

Class Diagram

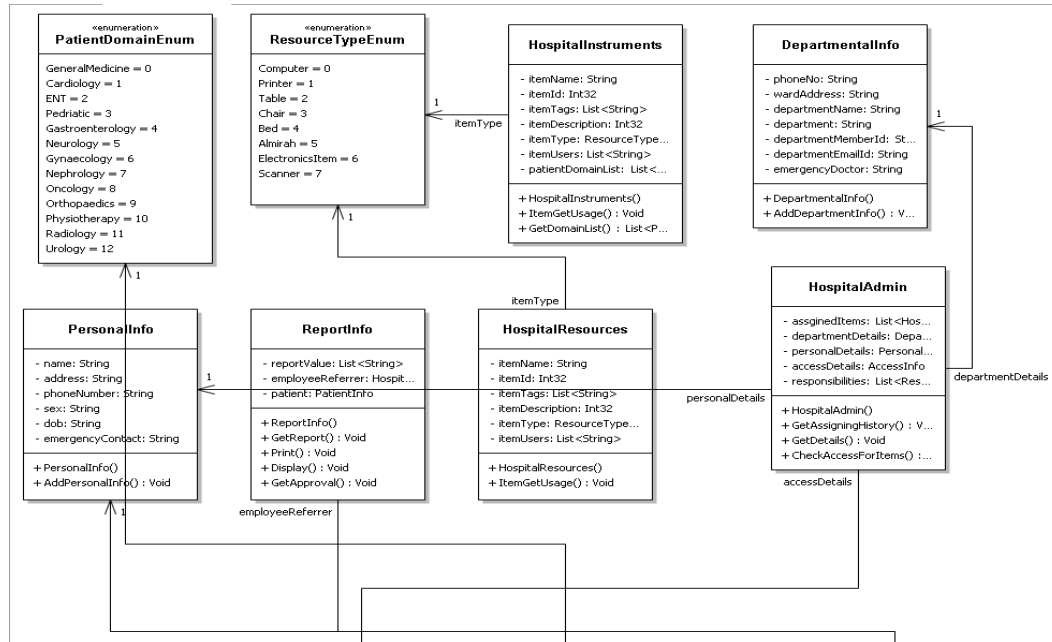


FIGURE 1:-CLASS DIAGRAM 1

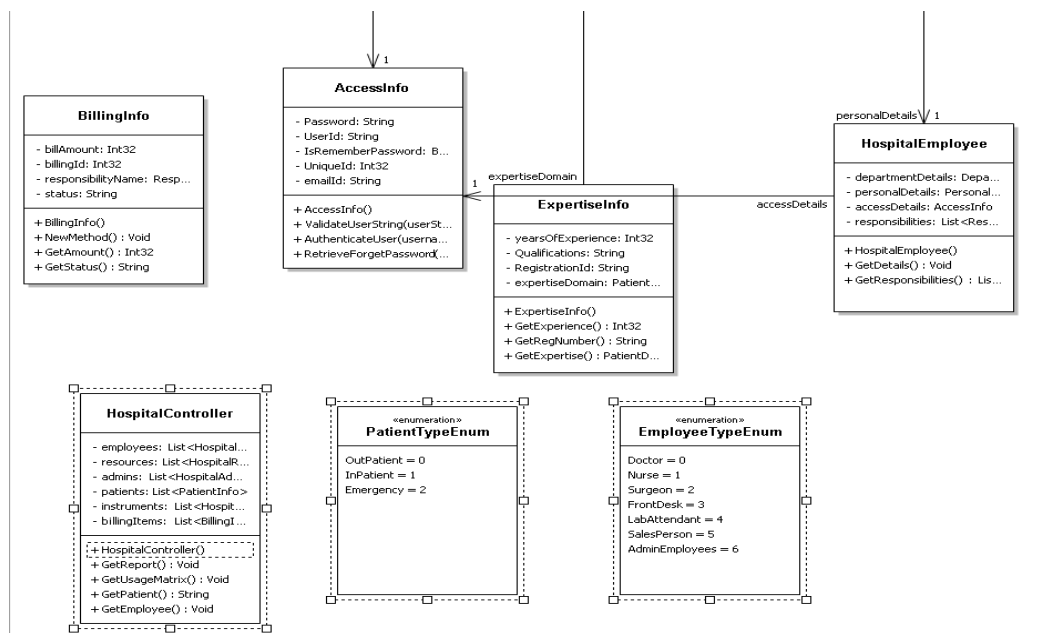


FIGURE 2:-CLASS DIAGRAM2

Data flow diagram

DFD-level 0

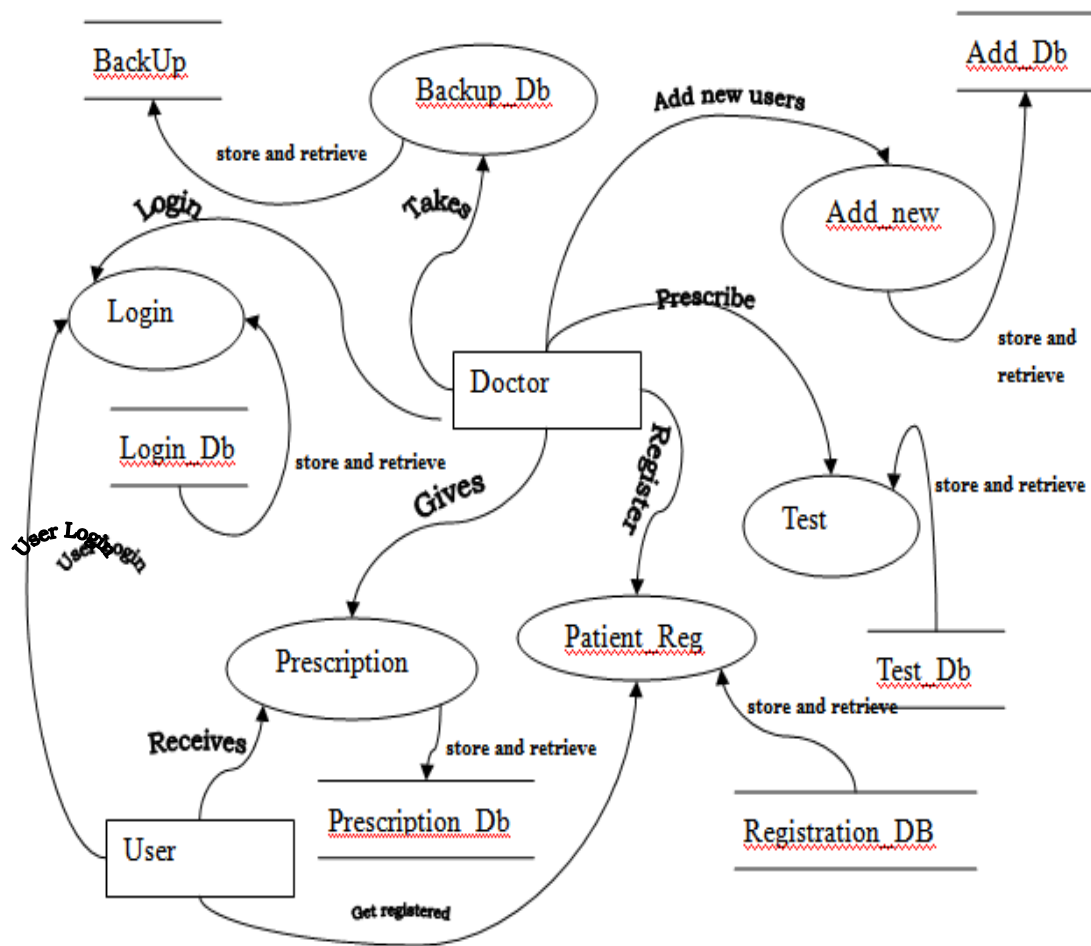


FIGURE 4 DFD-LEVEL 0

DFD level 1

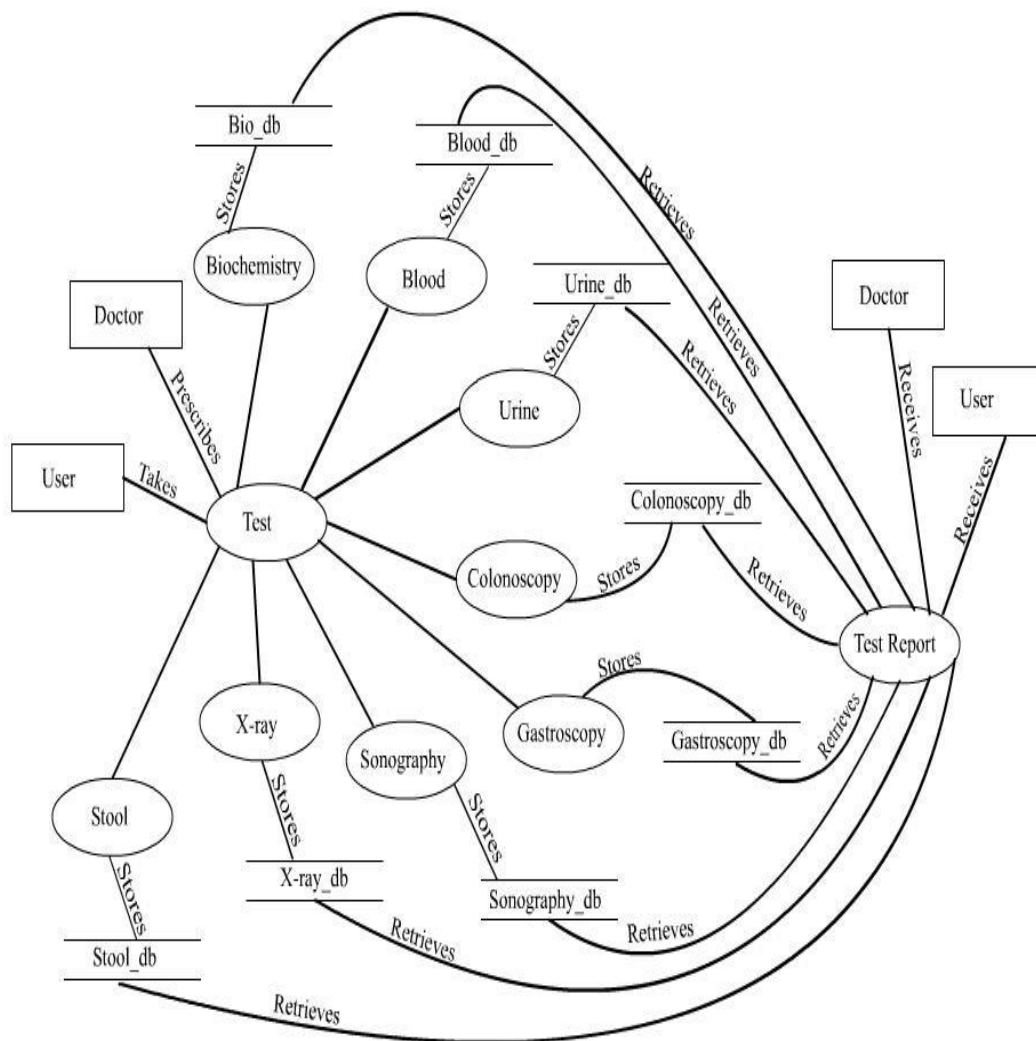


FIGURE 5 DFD LEVEL 1

DFD level 2

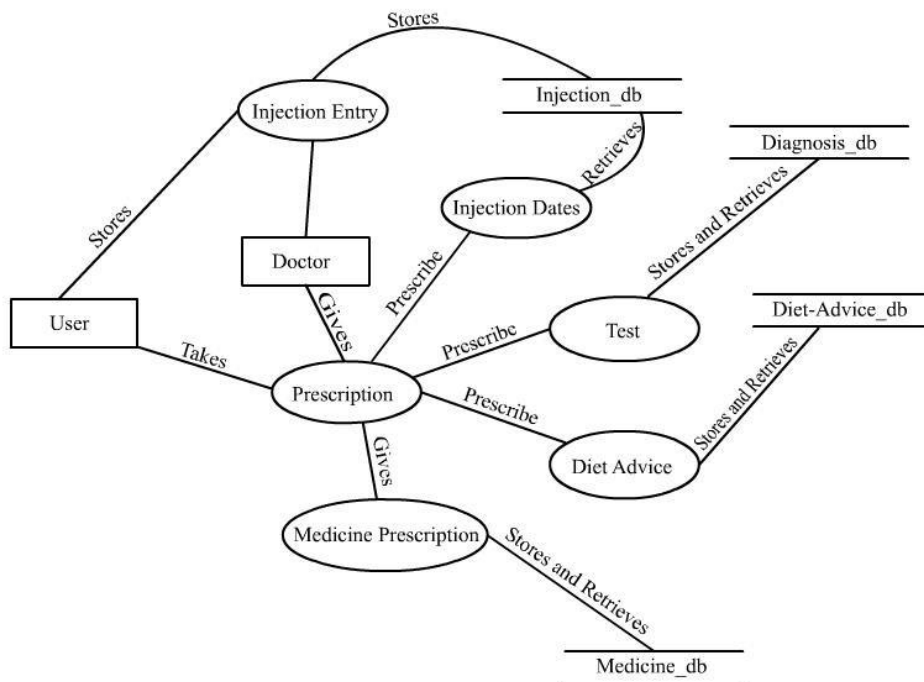


FIGURE 6 DFD LEVEL 2

DFD - level 3

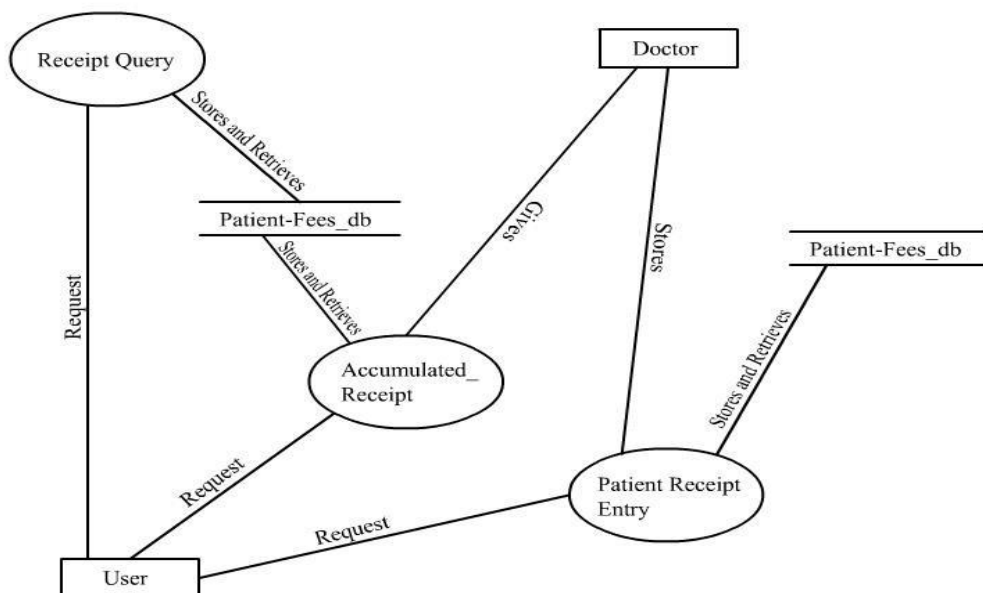
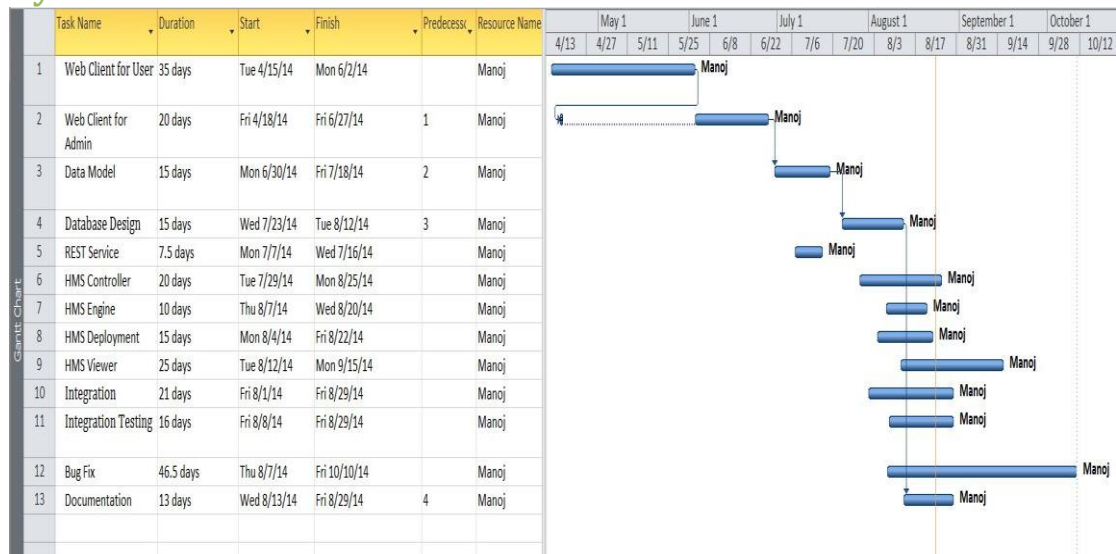
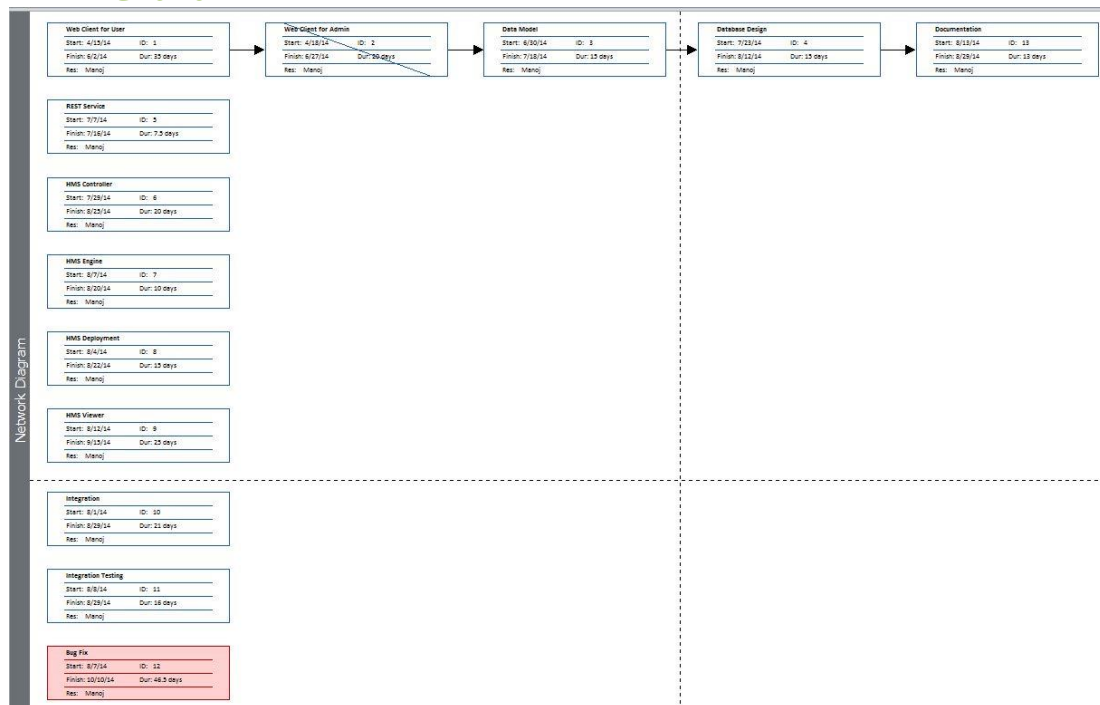


FIGURE 7 DFD LEVEL 3

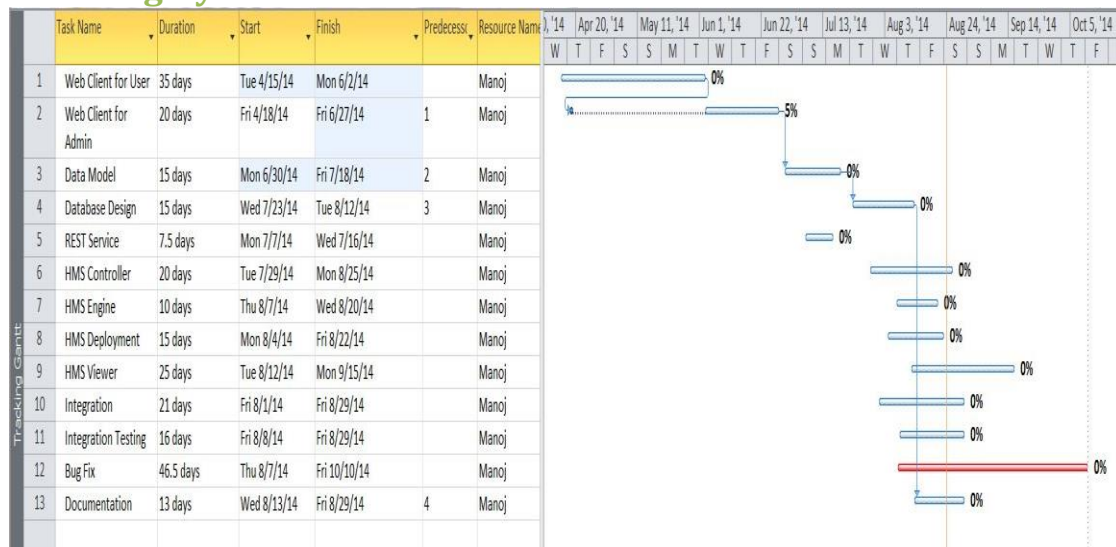
Gyantt Chart



PERT Chart



Tracking Gyantt



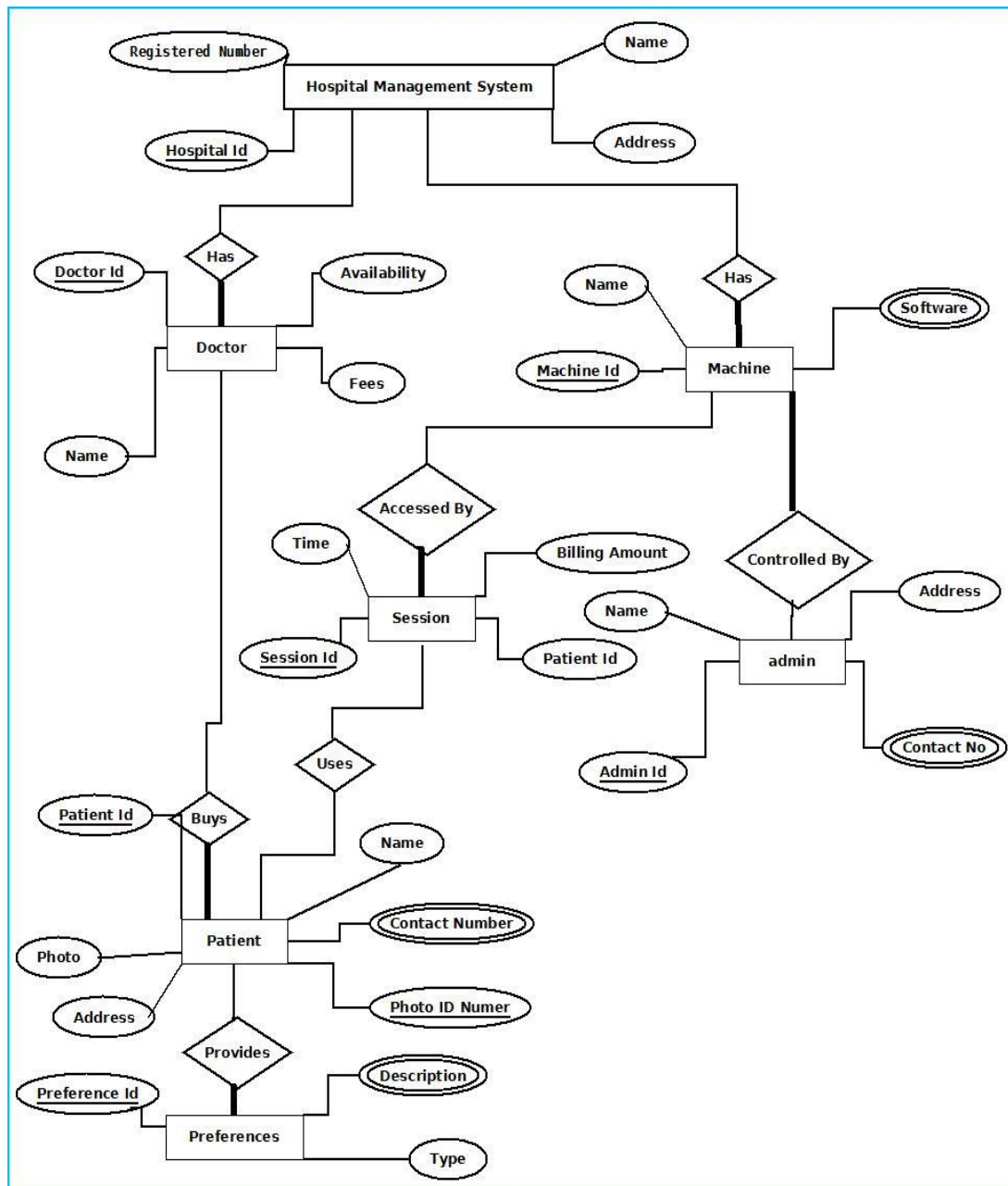
E-R Diagram:

We will design a RDBMS for File Management System. The entities and their attributes are listed below. Attributes in Bold letter is the unique key.

Entities	Attributes
Patient	User Id , Name, Address , Contact Number, password, Photo ID Num, Photo
Hospital Management System	Hospital Id , Name, Address, Registered no
Hospital Machine	Machine Id , Name, Software
Session	Session Id , User Id, Time, Billing amount
Doctor	Doctor Id , Name, address, contact number
User Preference	Preference Id , Type, Description
Admin	Admin id , name, price

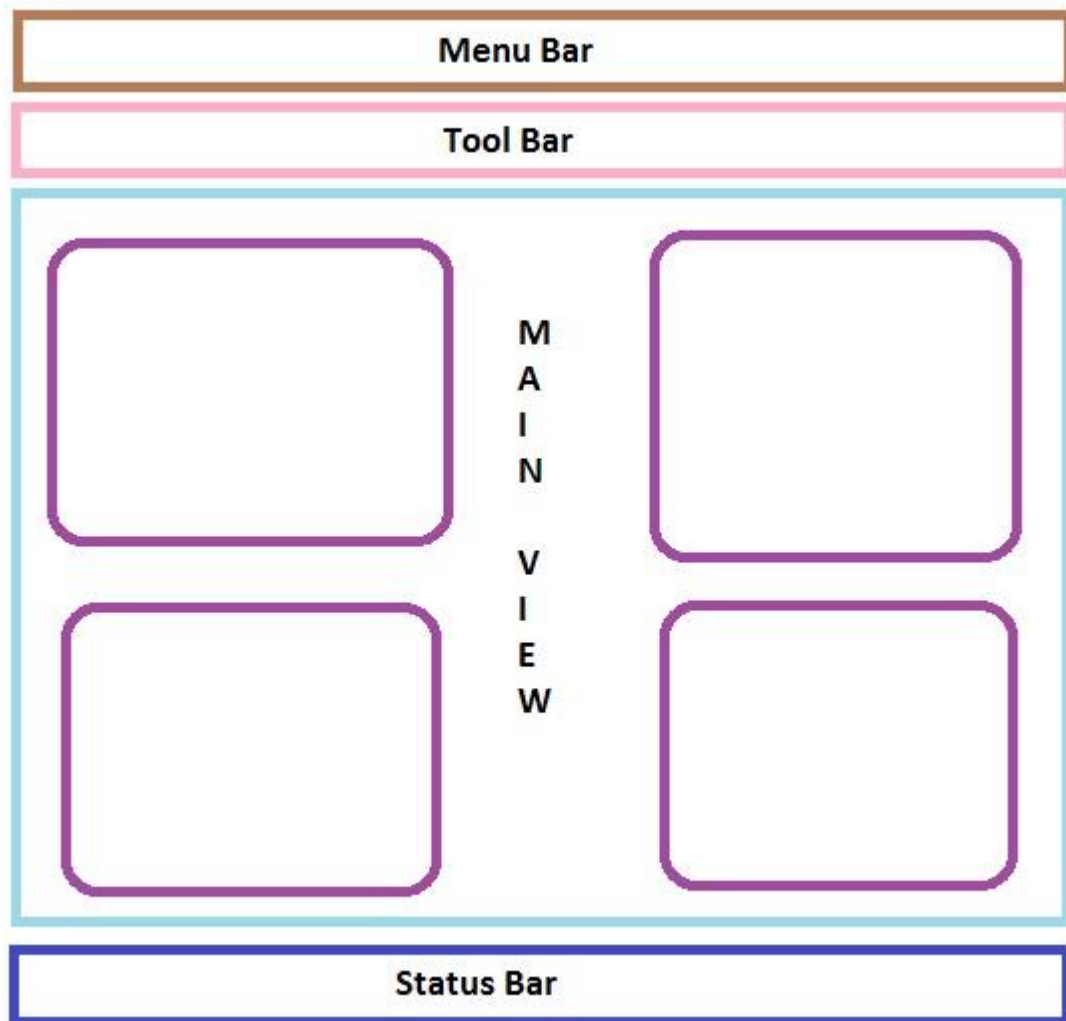
Relationship between Entities:

- ❖ Hospital Management System has Doctors → 1 : N
- ❖ Hospital Management System has Machine → 1 : N
- ❖ Patient System uses Session → 1 : 1
- ❖ Hospital Management System Cures patients → 1 : N
- ❖ Users provide Preferences → M : N



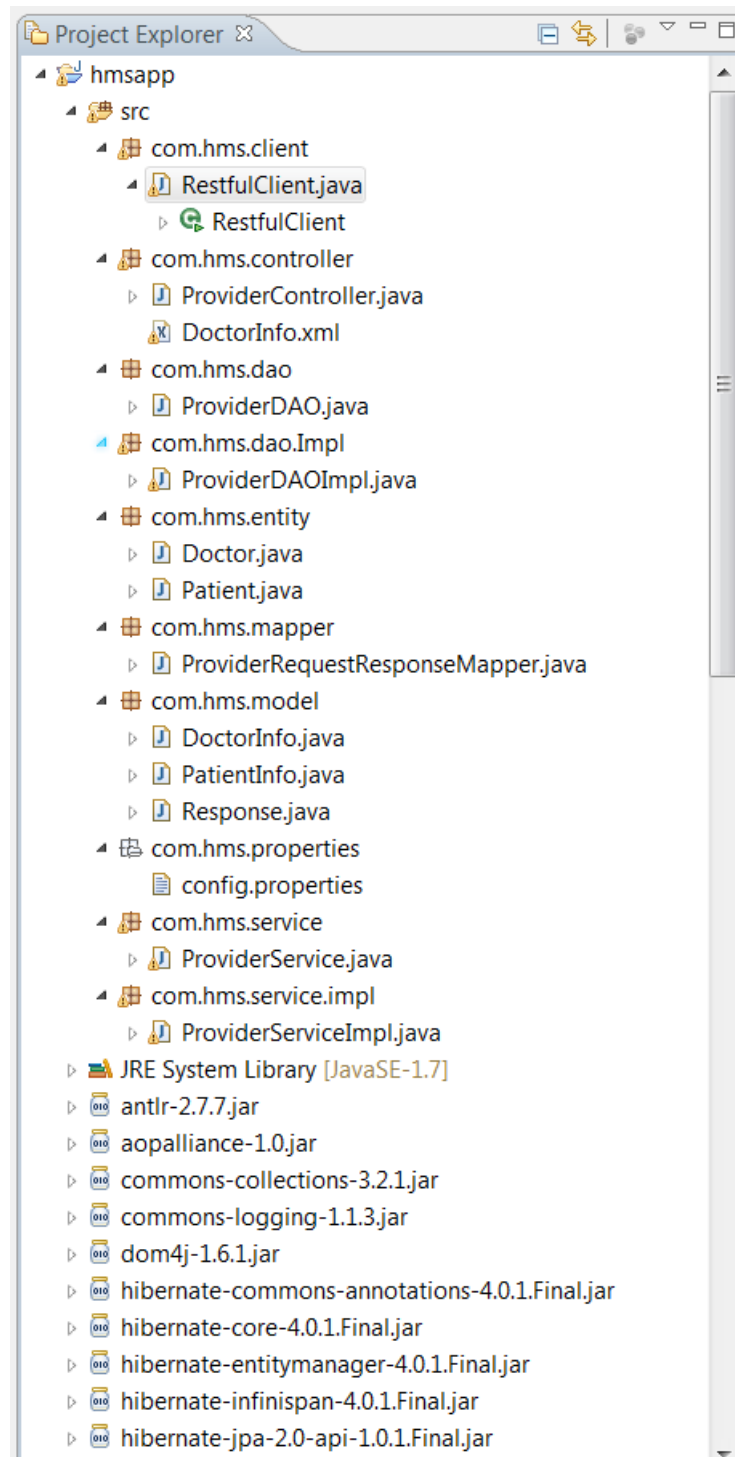
E-R Diagram of Hospital management system

The user interface:



PROGRAMME CODE:

Source Code Hierarchy



Source Code

ProviderDAO.java

```
package com.hms.dao;

import java.util.List;

import org.springframework.stereotype.Component;

import com.hms.entity.Doctor;
import com.hms.model.Response;

@Component
public interface ProviderDAO {

    public Response addDoctor(Doctor doc);

    //public Response updateDoctor(Doctor doc) ;

    public Response deleteDoctor(Doctor doc);

    public Doctor getDoctorByID(Doctor doc) ;

    public List<Doctor> getDoctorsByAttributes(Doctor doc);

}
```

ProviderDAOImpl.java

```
package com.hms.dao.impl;

import java.util.List;

import org.hibernate.Criteria;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Repository;
import org.springframework.transaction.annotation.Propagation;
import org.springframework.transaction.annotation.Transactional;

import com.hms.dao.ProviderDAO;
import com.hms.entity.Doctor;
import com.hms.model.Response;

@Repository
@Transactional
public class ProviderDAOImpl implements ProviderDAO {

    @Autowired
    private SessionFactory sessionFactory;
```



```

Session session = null;

@Override
@Transactional(readOnly=false,propagation=Propagation.REQUIRED)
public Response addDoctor(Doctor doc) {

    Response response =new Response();
    try {
        session = sessionFactory.getCurrentSession();
        session.saveOrUpdate(doc);

    } catch (Exception e) {

    }

    response.setCode("HMS001");
    response.setStatus("Successfully Added.");
    return response;

}

@Override
@Transactional(readOnly=false,propagation=Propagation.REQUIRED)
public Response deleteDoctor(Doctor doc) {

    Response response =new Response();
    try {
        session = sessionFactory.getCurrentSession();
        session.delete(doc);
        session.flush();
    } catch (Exception e) {

    }

    response.setCode("HMS002");
    response.setStatus("Record deleted succesfully.");
    return response;

}

/*
@Override
@Transactional(readOnly=false,propagation=Propagation.REQUIRED)
public Response updateDoctor(Doctor doc) {

    Response response =new Response();
    try {
        session = sessionFactory.getCurrentSession();

        Doctor docEntity =
        (Doctor)session.get(Doctor.class, doc.getIddoctors());

        session.saveOrUpdate(doc);

    } catch (Exception e) {

```

```

    }

    response.setCode("HMS002");
    response.setStatus("Successfully Updated.");
    return response;

}
*/

@SuppressWarnings("unchecked")
@Override
@Transactional(readOnly=true,propagation=Propagation.REQUIRED)
public List<Doctor> getDoctorsByAttributes(Doctor doc) {

    List<Doctor> doctorList = null;

    try {
        session = sessionFactory.getCurrentSession();
        Criteria c = session.createCriteria(Doctor.class);

        /*if(!isEmpty(doc.getContactNumber()))
            c.add(Restrictions.eq("name.firstname",
doc.getContactNumber())));
            if(!isEmpty(doc.getContactNumber()))
            c.add(Restrictions.eq("name.lastname", lastName));
            if(!isEmpty(doc.getContactNumber()))
            c.add(Restrictions.eq("demographic.sex", gender));
            if(!isEmpty(doc.getContactNumber()))
            c.add(Restrictions.eq("master.dob",
DateUtil.convertDateToSqlformat(dob)));
        */

        doctorList = (List<Doctor>)c.list();

    } catch (Exception e) {

    }

    return doctorList;
}

@SuppressWarnings("unchecked")
@Override
@Transactional(readOnly=true,propagation=Propagation.REQUIRED)
public Doctor getDoctorByID(Doctor doc) {

    Doctor docEntity=null;
    //List<Doctor> doctorList = null;

    try {
        session = sessionFactory.getCurrentSession();
        docEntity= (Doctor) session.load(Doctor.class,
doc.getIddoctors());

        System.out.println("inside getDoctorByID in DAO impl " +
docEntity.getName());
    } catch (Exception e) {

    }

}

```

```

        return docEntity;
    }

    public static boolean isNullOrEmpty(Object str) {
        return (str == null || str.toString().trim().isEmpty()) ? true : false;
    }
}

```

DoctorEntity.java

```

package com.hms.entity;

import java.io.Serializable;
import javax.persistence.*;

/**
 * The persistent class for the doctors database table.
 *
 */
@Entity
@Table(name="doctors")
public class Doctor implements Serializable {
    private static final long serialVersionUID = 1L;

    @Id
    @SequenceGenerator(name="DOCTORS_IDDOCTORS_GENERATOR",
sequenceName="SEQ_DOCTORS")
    @GeneratedValue(strategy=GenerationType.SEQUENCE,
generator="DOCTORS_IDDOCTORS_GENERATOR")
    private Integer iddoctors;

    private String address;

    @Column(name="contact_number")
    private String contactNumber;

    @Column(name="doctor_reg_number")
    private String doctorRegNumber;

    @Column(name="doctor_specialty")
    private String doctorSpecialty;

    private String name;

    public Doctor() {
    }

    public Integer getIddoctors() {
        return this.iddoctors;
    }

    public void setIddoctors(Integer iddoctors) {
        this.iddoctors = iddoctors;
    }

    public String getAddress() {

```

```

        return this.address;
    }

    public void setAddress(String address) {
        this.address = address;
    }

    public String getContactNumber() {
        return this.contactNumber;
    }

    public void setContactNumber(String contactNumber) {
        this.contactNumber = contactNumber;
    }

    public String getDoctorRegNumber() {
        return this.doctorRegNumber;
    }

    public void setDoctorRegNumber(String doctorRegNumber) {
        this.doctorRegNumber = doctorRegNumber;
    }

    public String getDoctorSpecialty() {
        return this.doctorSpecialty;
    }

    public void setDoctorSpecialty(String doctorSpecialty) {
        this.doctorSpecialty = doctorSpecialty;
    }

    public String getName() {
        return this.name;
    }

    public void setName(String name) {
        this.name = name;
    }
}

```

PatientEntity.java

```

package com.hms.entity;

import java.io.Serializable;
import javax.persistence.*;

/**
 * The persistent class for the patients database table.
 *
 */
@Entity
@Table(name="patients")
public class Patient implements Serializable {
    private static final long serialVersionUID = 1L;

```

```

        @Id
        @SequenceGenerator(name="PATIENTS_IDPATIENTS_GENERATOR",
sequenceName="SEQ_PATIENTS")
        @GeneratedValue(strategy=GenerationType.SEQUENCE,
generator="PATIENTS_IDPATIENTS_GENERATOR")
        private Integer idpatients;

        private String address;

        @Column(name="contact_number")
        private String contactNumber;

        @Column(name="govt_photo_id")
        private String govtPhotoId;

        private String name;

        @Column(name="photo_path")
        private String photoPath;

    public Patient() {
    }

    public Integer getIdpatients() {
        return this.idpatients;
    }

    public void setIdpatients(Integer idpatients) {
        this.idpatients = idpatients;
    }

    public String getAddress() {
        return this.address;
    }

    public void setAddress(String address) {
        this.address = address;
    }

    public String getContactNumber() {
        return this.contactNumber;
    }

    public void setContactNumber(String contactNumber) {
        this.contactNumber = contactNumber;
    }

    public String getGovtPhotoId() {
        return this.govtPhotoId;
    }

    public void setGovtPhotoId(String govtPhotoId) {
        this.govtPhotoId = govtPhotoId;
    }

    public String getName() {
        return this.name;
    }

    public void setName(String name) {
        this.name = name;
    }
}

```

```

        public String getPhotoPath() {
            return this.photoPath;
        }

        public void setPhotoPath(String photoPath) {
            this.photoPath = photoPath;
        }
    }
}

```

AccessInfo

```

package HospitalManagementSystem.HospitalUserGUI;

public class AccessInfo
{
    string Password;
    string UserId;
    bool IsRememberPassword;
    int UniqueId;
    string emailId;

    public void ValidateUserString(string userString)
    {
    }

    public void AuthenticateUser(string username, string password)
    {
    }

    public void RetrieveForgetPassword(string username, string
password)
    {
    }
}

```

BillingInfo

```

package HospitalManagementSystem.HmsDesign;

public class BillingInfo
{
    int billAmount;
    int billingId;
    Responsibility responsibilityName;
    string status;

    public void NewMethod()
    {
    }
}

```

```

        throw new NotImplementedException();
    }

    public int GetAmount()
    {
        throw new NotImplementedException();
    }

    public string GetStatus()
    {
        throw new NotImplementedException();
    }
}

```

DepartmentalInfo

```

package HospitalManagementSystem.HmsDesign;

public class DepartmentalInfo
{
    string phoneNo;
    string wardAddress;
    string departmentName;
    string department;
    string departmentMemberId;
    string departmentEmailId;
    string emergencyDoctor;

    public void AddDepartmentInfo()
    {
        throw new NotImplementedException();
    }
}

```

EmployeeTypeEnum.

```

package HospitalManagementSystem.HmsDesign;

public enum EmployeeTypeEnum
{
    Doctor,
    Nurse,
    Surgeon,
    FrontDesk,
    LabAttendant,
    SalesPerson,
    AdminEmployees
}

```

ExpertiseInfo

```
package HospitalManagementSystem.HmsDesign;

public class ExpertiseInfo
{
    int yearsOfExperience;
    string Qualifications;
    string RegistrationId;
    PatientDomainEnum expertiseDomain;

    public int GetExperience()
    {
        throw new NotImplementedException();
    }

    public string GetRegNumber()
    {
        throw new NotImplementedException();
    }

    public PatientDomainEnum GetExpertise()
    {
        throw new NotImplementedException();
    }
}
```

HospitalAdmin

```
package HospitalManagementSystem.HmsDesign;

public class HospitalAdmin
{
    List<HospitalInstruments> assignedItems;
    DepartmentalInfo departmentDetails;
    PersonalInfo personalDetails;
    AccessInfo accessDetails;
    List<Responsibility> responsibilities;

    public void GetAssigningHistory()
    {
        throw new NotImplementedException();
    }

    public void GetDetails()
    {
        throw new NotImplementedException();
    }
}
```



```

        public void CheckAccessForItems()
        {
            throw new NotImplementedException();
        }
    }
}

```

HospitalController

```

package HospitalManagementSystem.HmsDesign;

public class HospitalController
{
    List<HospitalEmployee> employees;
    List<HospitalResources> resources;
    List<HospitalAdmin> admins;
    List<PatientInfo> patients;
    List<HospitalInstruments> instruments;
    List<BillingInfo> billingItems;

    public void GetReport()
    {
        throw new NotImplementedException();
    }

    public void GetUsageMatrix()
    {
        throw new NotImplementedException();
    }

    public string GetPatient()
    {
        throw new NotImplementedException();
    }

    public void GetEmployee()
    {
        throw new NotImplementedException();
    }
}

```

HospitalEmployee

```

package HospitalManagementSystem.HmsDesign;

public class HospitalEmployee
{
    DepartmentalInfo departmentDetails;
    PersonalInfo personalDetails;
    AccessInfo accessDetails;
    List<Responsibility> responsibilities;
}

```

```

        public void GetDetails()
        {
            throw new NotImplementedException();
        }

        public List<Responsibility> GetResponsibilities()
        {
            throw new NotImplementedException();
        }
    }
}

```

HospitalInstruments

```

package HospitalManagementSystem.HmsDesign;

public class HospitalInstruments
{
    string itemName;
    int itemId;
    List<string> itemTags;
    int itemDescription;
    ResourceTypeEnum itemType;
    List<string> itemUsers;
    List<PatientDomainEnum> patientDomainList;

    public void ItemGetUsage()
    {
        throw new NotImplementedException();
    }

    public List<PatientDomainEnum> GetDomainList()
    {
        throw new NotImplementedException();
    }
}

```

HospitalResources

```

package HospitalManagementSystem.HmsDesign;

public class HospitalResources
{
    string itemName;
    int itemId;
    List<string> itemTags;
    int itemDescription;
    ResourceTypeEnum itemType;
    List<string> itemUsers;

    public void ItemGetUsage()
    {

```

```
        throw new NotImplementedException();
    }
}
```

MedicalInstruments

```
package HospitalManagementSystem.HmsDesign;

public enum MedicalInstruments
{
    ECG_machine,
    USG_machine,
    X_RAY_machine,
    CT_SCANNER,
    OXYGEN_SETUP
}
```

PatientDomainEnum

```
package HospitalManagementSystem.HmsDesign;

public enum PatientDomainEnum
{
    GeneralMedicine,
    Cardiology,
    ENT,
    Pedriatic,
    Gastroenterology,
    Neurology,
    Gynaecology,
    Nephrology,
    Oncology,
    Orthopaedics,
    Physiotherapy,
    Radiology,
    Urology
}
```

PatientInfo

```

package HospitalManagementSystem.HmsDesign;

    public class PatientInfo
    {
        List<MedicalInstruments> assignedItems;
        DepartmentalInfo deptDetails;
        PersonalInfo personalDetails;
        AccessInfo accessDetails;

        public void GetAssigningHistory()
        {
            throw new NotImplementedException();
        }

        public void GetDetails()
        {
            throw new NotImplementedException();
        }

        public void CheckAccessForItems()
        {
            throw new NotImplementedException();
        }
    }

```

PatientTypeEnum

```

package HospitalManagementSystem.HmsDesign;

    public enum PatientTypeEnum
    {
        OutPatient,
        InPatient,
        Emergency
    }

```

PersonalInfo

```

package HospitalManagementSystem.HmsDesign;

    public class PersonalInfo
    {
        string name;
        string address;
        string phoneNumber;
    }

```

```

        string sex;
        string dob;
        string emergencyContact;

        public void AddPersonalInfo()
        {
            throw new NotImplementedException();
        }
    }

```

ReportInfo

```

package HospitalManagementSystem.HmsDesign;

public class ReportInfo
{
    List<string> reportValue;
    HospitalEmployee employeeReferrer;
    PatientInfo patient;

    public void GetReport()
    {
        throw new NotImplementedException();
    }

    public void Print()
    {
        throw new NotImplementedException();
    }

    public void Display()
    {
        throw new NotImplementedException();
    }

    public void GetApproval()
    {
        throw new NotImplementedException();
    }
}

```

ResourceTypeEnum

```

package HospitalManagementSystem.HmsDesign;

public enum ResourceTypeEnum
{
    Computer,
    Printer,
    Table,
    Chair,

```

```

        Bed,
        Almirah,
        ElectronicsItem,
        Scanner
    }

```

Responsibility

```

package HospitalManagementSystem.HmsDesign;

public class Responsibility
{
    int description;
    int id;
    int relatedInstruments;
    int relatedResources;
    int relatedPatients;
    int releatedDoctor;

    public int GetStatus()
    {
        throw new NotImplementedException();
    }

    public string GetResponsible()
    {
        throw new NotImplementedException();
    }
}

```

VideoSurveillance.html

```

<Window x:Class="CyberCafeController.VideoSurveillance"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="VideoSurveillance" Height="307" Width="339">
    <Grid Height="270">
        <Grid.RowDefinitions>
            <RowDefinition Height="226*" />
            <RowDefinition Height="36*" />
        </Grid.RowDefinitions>
        <MediaElement x:Name="MediaEL" Grid.RowSpan="1" LoadedBehavior="Manual" />
        <StackPanel Orientation="Horizontal" Grid.Row="1" HorizontalAlignment="Center"
            Margin="8,0,8,5" Width="301">
            <Button x:Name="btnRec" Content="Record" Click="btnRec_Click"
                Width="50" Height="25"/>
            <Button x:Name="btnPlay" Content="Play" Click="btnPlay_Click"
                Width="50" Height="25"/>
        </StackPanel>
    </Grid>

```

```

        <Button x:Name="btnStop" Content="Stop" Click="btnStop_Click"
            Width="50" Height="25"/>
        <Button x:Name="btnMoveBackward" Content="Back" Click="btnMoveBackward_Click"
            Width="50" Height="25"/>
        <Button x:Name="btnMoveForward" Content="Forward" Click="btnMoveForward_Click"
            Width="50" Height="25"/>
        <Button x:Name="btnOpen" Content="Open" Click="btnOpen_Click"
            Width="50" Height="25"/>
    </StackPanel>
</Grid>
</Window>

```

VideoSurveillance.java

```

package HospitalController;
/// <summary>
/// Interaction logic for VideoSurveillance.html
/// </summary>
public partial class VideoSurveillance : Window
{
    public VideoSurveillance()
    {
        InitializeComponent();
        IsPlaying(false);
    }

    private void btnRec_Click(object sender, RoutedEventArgs e)
    {
    }

    #region IsPlaying(bool)
    private void IsPlaying(bool bValue)
    {
        btnStop.IsEnabled = bValue;
        btnMoveBackward.IsEnabled = bValue;
        btnMoveForward.IsEnabled = bValue;
        btnPlay.IsEnabled = bValue;
    }
    #endregion

    #region Play and Pause
    private void btnPlay_Click(object sender, RoutedEventArgs e)
    {
        IsPlaying(true);
        if (btnPlay.Content.ToString() == "Play")
        {
            MediaEL.Play();
            btnPlay.Content = "Pause";
        }
        else
        {
            MediaEL.Pause();
        }
    }
}

```

```

        btnPlay.Content = "Play";
    }
}
#endregion

#region Stop
private void btnStop_Click(object sender, RoutedEventArgs e)
{
    MediaEL.Stop();
    btnPlay.Content = "Play";
    IsPlaying(false);
    btnPlay.IsEnabled = true;
}
#endregion

#region Back and Forward
private void btnMoveForward_Click(object sender, RoutedEventArgs e)
{
    MediaEL.Position = MediaEL.Position + TimeSpan.FromSeconds(10);
}

private void btnMoveBackward_Click(object sender, RoutedEventArgs e)
{
    MediaEL.Position = MediaEL.Position - TimeSpan.FromSeconds(10);
}
#endregion

#region Open Media
private void btnOpen_Click(object sender, RoutedEventArgs e)
{
    Microsoft.Win32.OpenFileDialog ofd = new Microsoft.Win32.OpenFileDialog();
    ofd.Filter = "Video Files (*.wmv)|*.wmv";
    if (ofd.ShowDialog() == true)
    {
        MediaEL.Source = new Uri(ofd.FileName);
        btnPlay.IsEnabled = true;
    }
}
#endregion
}
}

```

HospitalStorage.java

```

package HospitalStorage;

public class HospitalStorage
{
    #region database interction data fetching

```



```

        MySql.Data.MySqlClient.MySqlConnection msqlConnection = null;

        private void fetchUserData()
        {
            msqlConnection = new
            MySql.Data.MySqlClient.MySqlConnection("server=localhost;user
            id=root;Password=technicise;database=sptdb;persist security info=False");
            try
            {
                //define the command reference
                MySql.Data.MySqlClient.MySqlCommand msqlCommand = new
                MySql.Data.MySqlClient.MySqlCommand();
                msqlCommand.Connection = msqlConnection;

                msqlConnection.Open();

                msqlCommand.CommandText = "Select * from Users;";
                MySql.Data.MySqlClient.MySqlDataReader msqlReader =
                msqlCommand.ExecuteReader();
                _UsersCollection.Clear();

                while (msqlReader.Read())
                {
                    UsersData UsersDataObject = new UsersData();

                    //UsersDataObject.serialNo = msqlReader.GetString("sl_no");
                    UsersDataObject.UserAdress =
                    msqlReader.GetString("address");
                    UsersDataObject.phoneNumber = msqlReader.GetString("ph_no");
                    UsersDataObject.UserName =
                    msqlReader.GetString("User_name");
                    UsersDataObject.UserVatNo = msqlReader.GetString("vat_no");
                    UsersDataObject.UserId = msqlReader.GetString("id");
                    UsersDataObject.UserTurnOver =
                    msqlReader.GetString("turn_over");
                    UsersDataObject.UserDue = msqlReader.GetString("due");
                    _UsersCollection.Add(UsersDataObject);

                }

            }
            catch (Exception er)
            {
                MessageBox.Show(er.Message);
            }
            finally
            {
                //always close the connection
                msqlConnection.Close();
            }
        }
    #endregion

    #region User table update

    private void UpdateUserTable()
    {
        //define the connection reference and initialize it
        msqlConnection = new
        MySql.Data.MySqlClient.MySqlConnection("server=localhost;user
        id=root;Password=technicise;database=sptdb;persist security info=False");
    }

```

```

try
{
    //define the command reference
    MySql.Data.MySqlClient.MySqlCommand msqlCommand = new
    MySql.Data.MySqlClient.MySqlCommand();

    //define the connection used by the command object
    msqlCommand.Connection = msqlConnection;

    double dbDue = 0.0;
    double dbTurnOver = 0.0;

    //open the connection
    if (msqlConnection.State != System.Data.ConnectionState.Open)
        msqlConnection.Open();
    string cusIdKey = UserInfoTb.SelectedValue.ToString();
    string cmdStr = "SELECT turn_over,due FROM Users WHERE id='" +
    cusIdKey + "'";
    msqlCommand.CommandText = cmdStr;

    MySql.Data.MySqlClient.MySqlDataReader msqlReader =
    msqlCommand.ExecuteReader();

    while (msqlReader.Read())
    {
        dbDue = double.Parse(msqlReader.GetString("due"));
        dbTurnOver =
    double.Parse(msqlReader.GetString("turn_over"));
    }

    if (msqlConnection.State == System.Data.ConnectionState.Open)
        msqlConnection.Close();

    //updating the value
    if (msqlConnection.State != System.Data.ConnectionState.Open)
        msqlConnection.Open();

    double totalBilledAmount = CalculateTotalAmount() +
    CalculateVAT();

    double newDueDouble = dbDue + totalBilledAmount -
    double.Parse(paymentAmountTB.Text);
    String newDue = newDueDouble.ToString();
    String newTurnOver = (dbTurnOver +
    totalBilledAmount).ToString();
    msqlCommand.CommandText = "UPDATE Users SET due='" + newDue +
    "', turn_over='" + newTurnOver + "' WHERE id='" + cusIdKey + "'";

    msqlCommand.ExecuteNonQuery();

}
catch (Exception er)
{
    MessageBox.Show(er.Message);
}
finally
{
    if (msqlConnection.State == System.Data.ConnectionState.Open)

```

```

        msqlConnection.Close();
    }
}

#endregion
}
}

```

AddNewDoctorWindow.html

```

<Window x:Class="doctors.AddNewDoctorWindow"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:sys="clr-namespace:System;assembly=microsoft.windows.common-user-core-6.0.2"
    Title="Add New Doctor" Height="223" Width="596">
    <DockPanel LastChildFill="True">
        <UniformGrid Rows="1" DockPanel.Dock="Bottom" Height="40">
            <Button Width="100" Margin="0 5" Name="cancelBtn"
                KeyboardNavigation.TabIndex="5" Click="cancelBtn_Click">Cancel</Button>
            <Button Width="100" Margin="0 5" Name="addBtn"
                KeyboardNavigation.TabIndex="4" Click="addBtn_Click">Add</Button>
        </UniformGrid>
        <UniformGrid Columns="4" Rows="2" DockPanel.Dock="Top">
            <Button Name="doctorNameBtn"
                KeyboardNavigation.IsTabStop="False">Name</Button>
            <Button Name="addressBtn"
                KeyboardNavigation.IsTabStop="False">Address</Button>
            <Button Name="phNumberBtn"
                KeyboardNavigation.IsTabStop="False">Phone Number</Button>
            <Button Name="vatBtn" KeyboardNavigation.IsTabStop="False">Reg
No</Button>
            <TextBox Name="NameDataTB" KeyboardNavigation.TabIndex="0"
                HorizontalContentAlignment="Center" VerticalContentAlignment="Center"
                PreviewTextInput="NameDataTB_PreviewTextInput"></TextBox>
            <TextBox Name="AddrDataTB" KeyboardNavigation.TabIndex="1"
                HorizontalContentAlignment="Center" VerticalContentAlignment="Center"
                PreviewTextInput="AddrDataTB_PreviewTextInput"></TextBox>
            <TextBox Name="phDataTB" KeyboardNavigation.TabIndex="2"
                VerticalContentAlignment="Center" HorizontalContentAlignment="Center"
                PreviewTextInput="phDataTB_PreviewTextInput"></TextBox>
            <TextBox Name="vatdataTB" KeyboardNavigation.TabIndex="3"
                VerticalContentAlignment="Center" HorizontalContentAlignment="Center"
                PreviewTextInput="vatdataTB_PreviewTextInput"></TextBox>
        </UniformGrid>
    </DockPanel>
</Window>

```

AddNewDoctorWindow.java

```

package doctors;

/// <summary>
/// Interaction logic for AddNewDoctorWindow.html
/// </summary>
public partial class AddNewDoctorWindow : Window
{
    public AddNewDoctorWindow()
    {
        InitializeComponent();
    }

    #region validation
    private static bool onlyNumeric(string text)
    {
        Regex regex = new Regex("[^0-9.-]+"); //regex that matches
disallowed text
        return !regex.IsMatch(text);
    }
    private static bool onlyAlphabet(string text)
    {
        Regex regex = new Regex("[^A-Z|^a-z|^ |^\\t]"); //regex that matches
disallowed text
        return !regex.IsMatch(text);
    }

    private static bool onlyAlphaNumeric(string text)
    {
        Regex regex = new Regex("[a-zA-Z0-9]*$"); //regex that matches
disallowed text
        return !regex.IsMatch(text);
    }

    private void NameDataTB_PreviewTextInput(object sender,
TextCompositionEventArgs e)
    {
        e.Handled = !onlyAlphabet(e.Text);
    }

    private void AddrDataTB_PreviewTextInput(object sender,
TextCompositionEventArgs e)
    {
        //e.Handled = !onlyAlphaNumeric(e.Text);
    }

    private void phDataTB_PreviewTextInput(object sender,
TextCompositionEventArgs e)
    {
        e.Handled = !onlyNumeric(e.Text);
    }

    private void vatdataTB_PreviewTextInput(object sender,
TextCompositionEventArgs e)
    {
        e.Handled = onlyAlphaNumeric(e.Text);
    }
    #endregion

    private void cancelBtn_Click(object sender, RoutedEventArgs e)
    {

```

```

    }

    private void addBtn_Click(object sender, RoutedEventArgs e)
    {
        }
    }
}

```

App.html

```

<Application x:Class="HospitalManagementSystem.App"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    StartupUri="HospitalControllerGUI.html">
    <Application.Resources>

    </Application.Resources>
</Application>

```

BillingWindow.html

```

<Window x:Class="HospitalManagementSystem.BillingWindow"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:sys="clr-namespace:System;assembly=microsoft.windows.common-base-1.0"
    Title="Billing Window" Height="857" Width="864">
    <DockPanel>
        <UniformGrid DockPanel.Dock="Top" Rows="1">
            <Label Content="Date:"></Label>
            <DatePicker Height="25" Name="datePicker" Background="#FFA39797"
                SelectedDate="{x:Static sys:DateTime.Now}" />
            <Label Content="Invoice Number:"></Label>
            <TextBlock Name="invoiceNumberTB"
                Background="#FFA39797"></TextBlock>
        </UniformGrid>
        <UniformGrid Rows="1" Height="30" DockPanel.Dock="Top">
            <Button Content="Patient" Name="PatientSelectBtn"></Button>
            <Button Content="Seller" ></Button>
        </UniformGrid>
        <UniformGrid Height="80" DockPanel.Dock="Top" Rows="1">
            <TextBox Name="PatientInfoTb" VerticalAlignment="Center"
                VerticalContentAlignment="Center" HorizontalContentAlignment="Center" Margin="3"
                Height="57">Patient 001</TextBox>
            <TextBlock Name="sellerInfoTb" Margin="3"
                TextWrapping="Wrap"></TextBlock>
        </UniformGrid>
        <UniformGrid Height="160" DockPanel.Dock="Bottom" Rows="2">
            <TextBlock Background="#FFE5E2E2" TextWrapping="Wrap">This invoice
                shows the actual price of the items , decribed in are true and

```

```

correct</TextBlock>
    <Label></Label>
    <Label HorizontalContentAlignment="Center"
VerticalContentAlignment="Center">Payment Amount</Label>
    <TextBox Name="paymentAmountTB" HorizontalContentAlignment="Center"
VerticalContentAlignment="Center">0.0</TextBox>

    </UniformGrid>
    <UniformGrid DockPanel.Dock="Bottom" Rows="1">
        <Label>Declaration</Label>
        <Label HorizontalContentAlignment="Right">Authorized
Signature</Label>
    </UniformGrid>
    <DockPanel DockPanel.Dock="Bottom" LastChildFill="True">
        <Label>Price in Words:</Label>
        <Label></Label>
    </DockPanel>

    <DockPanel DockPanel.Dock="Bottom" Height="40">
        <Button DockPanel.Dock="Left" Name="calculateTotalBtn"
Content="Total" Width="260" HorizontalContentAlignment="Right"
FontWeight="Bold"></Button>
        <Label DockPanel.Dock="Left" Name="totalNoOfItems"
Width="100"></Label>
        <Label DockPanel.Dock="Left" Name="itemUnit" Width="300"
HorizontalContentAlignment="Left"></Label>
        <Label DockPanel.Dock="Left" Name="totalAmountLabel"
Width="100"></Label>
    </DockPanel>
    <DockPanel DockPanel.Dock="Bottom" Height="40">
        <Button DockPanel.Dock="Left" Content="VAT" Name="calculateVATBtn"
Width="260" HorizontalContentAlignment="Right" FontWeight="Bold"></Button>
        <Label DockPanel.Dock="Left" Width="100"></Label>
        <Label DockPanel.Dock="Left" Width="300" HorizontalContentAlignment="Left"
></Label>
        <Label DockPanel.Dock="Left" Name="vatAmount" Width="100"></Label>
    </DockPanel>
    <ListView Name="billingItemListView" DockPanel.Dock="Bottom"
ItemsSource="{Binding billingCollection}">
        <ListView.View>
            <GridView>
                <GridViewColumn Width="50" Header="Sl No"
DisplayMemberBinding="{Binding serialNo}" />
                <GridViewColumn Width="260" Header="Description"
DisplayMemberBinding="{Binding description}" />
                <GridViewColumn Width="100" Header="Quantity"
DisplayMemberBinding="{Binding quantity}" />
                <GridViewColumn Width="100" Header="VAT"
DisplayMemberBinding="{Binding vat}" />
                <GridViewColumn Width="100" Header="Rate"
DisplayMemberBinding="{Binding rate}" />
                <GridViewColumn Width="180" Header="Amount"
DisplayMemberBinding="{Binding amount}" />
            </GridView>
        </ListView.View>
    </ListView>

    </DockPanel>
</Window>

```

BillingWindow.java

```
package HospitalManagementSystem;

/// <summary>
/// Interaction logic for BillingWindow.html
/// </summary>
public partial class BillingWindow : Window
{
    public BillingWindow()
    {
        InitializeComponent();
    }
}
```

DoctorsBrowser.html

```
<Window x:Class="HospitalManagementSystem.DoctorsBrowser"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="DoctorsBrowser" Height="418" Width="1008">
    <DockPanel LastChildFill="True">
        <UniformGrid Height="30" DockPanel.Dock="Top" Rows="1">
            <Button Content="Show doctors" Name="showPaymentsBtn"
Click="showPaymentsBtn_Click" Margin="10 0" ></Button>
            <Button Content="Add New" Name="addBtn" Click="addBtn_Click"
Margin="10 0" ></Button>
            <Button Content="Delete" Margin="10 0" Name="deleteBtn"
Click="deleteBtn_Click"></Button>
            <Button Content="Print" Margin="10 0" Name="printBtn"
IsEnabled="True" Click="printBtn_Click"></Button>
        </UniformGrid>
        <ListView Name="businessPersonListView" DockPanel.Dock="Bottom"
IsSynchronizedWithCurrentItem="True">
            <ListView.View>
                <GridView>
                    <!-- GridViewColumn Width="50" Header="Sl No"
DisplayMemberBinding="{Binding serialNo}" /-->
                    <GridViewColumn Width="150" Header="Id"
DisplayMemberBinding="{Binding doctorId}" />
                    <GridViewColumn Width="260" Header="Name"
DisplayMemberBinding="{Binding doctorName}" />
                    <GridViewColumn Width="160" Header="Address"
DisplayMemberBinding="{Binding doctorAdress}" />
                    <GridViewColumn Width="100" Header="Ph no."
DisplayMemberBinding="{Binding phoneNumber}" />
                    <GridViewColumn Width="100" Header="Manager"
DisplayMemberBinding="{Binding doctorVatNo}" />
                    <GridViewColumn Width="100" Header="Expertise"
DisplayMemberBinding="{Binding doctorTurnOver}" />
                    <GridViewColumn Width="100" Header="Duty Cycle"
DisplayMemberBinding="{Binding doctorDue}" />
                </GridView>
            </ListView.View>
        </ListView>
    </DockPanel>
</Window>
```

```
</DockPanel>
</Window>
```

DoctorsBrowser.java

```
package HospitalManagementSystem;

/// <summary>
/// Interaction logic for DoctorsBrowser.html
/// </summary>
public partial class DoctorsBrowser : Window
{
    public DoctorsBrowser()
    {
        InitializeComponent();
    }

    private void showPaymentsBtn_Click(object sender, RoutedEventArgs e)
    {

    }

    private void deleteBtn_Click(object sender, RoutedEventArgs e)
    {

    }

    private void printBtn_Click(object sender, RoutedEventArgs e)
    {

    }

    private void addBtn_Click(object sender, RoutedEventArgs e)
    {

    }
}
}
```

EmployeesBrowser.html

```
<Window x:Class="HospitalManagementSystem.EmployeesBrowser"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Employees Browser" Height="695" Width="986">
    <DockPanel LastChildFill="True">
        <UniformGrid Height="30" DockPanel.Dock="Top" Rows="1">
            <Button Content="Show employees" Name="showPaymentsBtn"
Click="showPaymentsBtn_Click" Margin="10 0" ></Button>
            <Button Content="Add New" Name="addBtn" Click="addBtn_Click"
Margin="10 0" ></Button>
            <Button Content="Delete" Margin="10 0" Name="deleteBtn"
Click="deleteBtn_Click"></Button>
            <Button Content="Print" Margin="10 0" Name="printBtn"
```



```

IsEnabled="True" Click="showPaymentsBtn_Click"></Button>
</UniformGrid>
<ListView Name="businessPersonListView" DockPanel.Dock="Bottom"
    IsSynchronizedWithCurrentItem="True">
    <ListView.View>
        <GridView>
            <!-- GridViewColumn Width="50" Header="Sl No"
DisplayMemberBinding="{Binding serialNo}" /-->
            <GridViewColumn Width="150" Header="Id"
DisplayMemberBinding="{Binding employeeId}" />
            <GridViewColumn Width="260" Header="Name"
DisplayMemberBinding="{Binding employeeName}" />
            <GridViewColumn Width="160" Header="Address"
DisplayMemberBinding="{Binding employeeAddress}" />
            <GridViewColumn Width="100" Header="Ph no."
DisplayMemberBinding="{Binding phoneNumber}" />
            <GridViewColumn Width="100" Header="Manager"
DisplayMemberBinding="{Binding employeeVatNo}" />
            <GridViewColumn Width="100" Header="Expertise"
DisplayMemberBinding="{Binding employeeTurnOver}" />
            <GridViewColumn Width="100" Header="Duty Cycle"
DisplayMemberBinding="{Binding employeeDue}" />
        </GridView>
    </ListView.View>
</ListView>
</DockPanel>
</Window>

```

EmployeesBrowser.java

```

package HospitalManagementSystem;

/// <summary>
/// Interaction logic for EmployeesBrowser.html
/// </summary>
public partial class EmployeesBrowser : Window
{
    public EmployeesBrowser()
    {
        InitializeComponent();
    }

    private void showPaymentsBtn_Click(object sender, RoutedEventArgs e)
    {

    }

    private void deleteBtn_Click(object sender, RoutedEventArgs e)
    {

    }

    private void addBtn_Click(object sender, RoutedEventArgs e)
    {

    }
}
}

```

HospitalControllerGUI.html

```
<Window x:Class="HospitalManagementSystem.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Hospital Controller" Height="372" Width="1066">
    <DockPanel>
        <Expander IsExpanded="True" Background="Gray" Header="Login"
Name="expander1" DockPanel.Dock="Left" Expanded="expander1_Expanded"
Height="486" Width="476">
            <UniformGrid Rows="6" HorizontalAlignment="Stretch"
VerticalAlignment="Stretch">
                <Label>Name</Label>
                <TextBox Width="200" Height="40"></TextBox>
                <Label>Password</Label>
                <PasswordBox Width="200" Height="40"></PasswordBox>
                <UniformGrid Columns="1">
                    <CheckBox>Remember Password</CheckBox>
                    <Label Hyperlink.Click="Label_Click">Forgot Password</Label>
                </UniformGrid>

                <Button Width="200" Height="40">Login</Button>
                <Button Click="Button_Click_6">Print Report</Button>
                <Button Click="Button_Click_7">Settings</Button>
            </UniformGrid>
        </Expander>
        <Expander Header="Features Access" IsExpanded="True"
DockPanel.Dock="Right">
            <UniformGrid Columns="2">
                <Button Click="Button_Click">Patients</Button>
                <Button Click="Button_Click_5">Doctors</Button>
                <Button Click="Button_Click_6">Billing</Button>
                <Button Click="Button_Click">Employees</Button>
                <Button Click="Button_Click_1">Resources</Button>
                <Button Click="Button_Click_2">Instruments</Button>
                <Button Click="Button_Click_3">Patient Registration</Button>
                <Button Click="Button_Click_4">Video Surveillance</Button>
            </UniformGrid>
        </Expander>
    </DockPanel>
</Window>
```

HospitalControllerGUI.java

```
package HospitalManagementSystem;

    /// <summary>
    /// Interaction logic for MainWindow.html
    /// </summary>
    public partial class MainWindow : Window
```

```

{
    public MainWindow()
    {
        InitializeComponent();
    }

    private void expander1_Expanded(object sender, RoutedEventArgs e)
    {

    }

    private void Button_Click(object sender, RoutedEventArgs e)
    {
        PatientsBrowser pb = new PatientsBrowser();
        pb.ShowDialog();
    }

    private void Button_Click_1(object sender, RoutedEventArgs e)
    {

    }

    private void Button_Click_2(object sender, RoutedEventArgs e)
    {
        InstrumentsBrowser ib = new InstrumentsBrowser();
        ib.ShowDialog();
    }

    private void Button_Click_3(object sender, RoutedEventArgs e)
    {
        PatientRegistration pr = new PatientRegistration();
        pr.ShowDialog();
    }

    private void Button_Click_4(object sender, RoutedEventArgs e)
    {

    }

    private void Button_Click_5(object sender, RoutedEventArgs e)
    {
        DoctorsBrowser db = new DoctorsBrowser();
        db.ShowDialog();
    }

    private void Button_Click_6(object sender, RoutedEventArgs e)
    {
        BillingWindow bw = new BillingWindow();
        bw.ShowDialog();
    }

    private void Label_Click(object sender, RoutedEventArgs e)
    {

    }

    private void Button_Click_7(object sender, RoutedEventArgs e)
    {

    }

    private void Button_Click_8(object sender, RoutedEventArgs e)

```

```

    {
        EmployeesBrowser eb = new EmployeesBrowser();
        eb.ShowDialog();
    }
}

```

InstrumentsBrowser.html

```

<Window x:Class="HospitalManagementSystem.InstrumentsBrowser"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Instruments Browser" Height="494" Width="1002">
    <DockPanel LastChildFill="True">
        <UniformGrid Height="30" DockPanel.Dock="Top" Rows="1">
            <Button Content="Show instruments" Name="showPaymentsBtn"
Click="showPaymentsBtn_Click" Margin="10 0" ></Button>
            <Button Content="Add New" Name="addBtn" Click="addBtn_Click"
Margin="10 0" ></Button>
            <Button Content="Delete" Margin="10 0" Name="deleteBtn"
Click="deleteBtn_Click_1"></Button>
            <Button Content="Print" Margin="10 0" Name="printBtn"
IsEnabled="True" Click="printBtn_Click_1"></Button>
        </UniformGrid>
        <ListView Name="businessPersonListView" DockPanel.Dock="Bottom"
IsSynchronizedWithCurrentItem="True">
            <ListView.View>
                <GridView>
                    <!-- GridViewColumn Width="50" Header="Sl No"
DisplayMemberBinding="{Binding serialNo}" /-->
                    <GridViewColumn Width="150" Header="Id"
DisplayMemberBinding="{Binding instrumentId}" />
                    <GridViewColumn Width="260" Header="Name"
DisplayMemberBinding="{Binding instrumentName}" />
                    <GridViewColumn Width="160" Header="Location"
DisplayMemberBinding="{Binding instrumentAddress}" />
                    <GridViewColumn Width="100" Header="Ph no."
DisplayMemberBinding="{Binding phoneNumber}" />
                    <GridViewColumn Width="100" Header="Admin"
DisplayMemberBinding="{Binding instrumentVatNo}" />
                    <GridViewColumn Width="100" Header="Usage"
DisplayMemberBinding="{Binding instrumentTurnOver}" />
                </GridView>
            </ListView.View>
        </ListView>
    </DockPanel>
</Window>

```

InstrumentsBrowser.java

```

package HospitalManagementSystem;

/// <summary>
/// Interaction logic for InstrumentsBrowser.html
/// </summary>
public partial class InstrumentsBrowser : Window

```

```

{
    public InstrumentsBrowser()
    {
        InitializeComponent();
    }

    private void showPaymentsBtn_Click(object sender, RoutedEventArgs e)
    {

    }

    private void deleteBtn_Click(object sender, RoutedEventArgs e)
    {

    }

    private void printBtn_Click(object sender, RoutedEventArgs e)
    {

    }

    private void addBtn_Click(object sender, RoutedEventArgs e)
    {

    }

    private void deleteBtn_Click_1(object sender, RoutedEventArgs e)
    {

    }

    private void printBtn_Click_1(object sender, RoutedEventArgs e)
    {

    }
}
}

```

PatientRegistration.html

```

<Window x:Class="HospitalManagementSystem.PatientRegistration"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Patient Registration" Height="303" Width="660">
    <DockPanel LastChildFill="True">
        <UniformGrid Rows="1" DockPanel.Dock="Bottom" Height="40">
            <Button Width="100" Margin="0 5" Name="cancelBtn"
                KeyboardNavigation.TabIndex="5" Click="cancelBtn_Click">Cancel</Button>
            <Button Width="100" Margin="0 5" Name="addBtn"
                KeyboardNavigation.TabIndex="4" Click="addBtn_Click">Add</Button>
        </UniformGrid>
        <UniformGrid Columns="4" Rows="2" DockPanel.Dock="Top">
            <Button Name="doctorNameBtn"
                KeyboardNavigation.IsTabStop="False">Name</Button>
            <Button Name="addressBtn"
                KeyboardNavigation.IsTabStop="False">Address</Button>
            <Button Name="phNumberBtn"

```

```

KeyboardNavigation.IsTabStop="False">Phone Number</Button>
    <Button Name="vatBtn"
KeyboardNavigation.IsTabStop="False">Problems</Button>
    <TextBox Name="NameDataTB" KeyboardNavigation.TabIndex="0"
HorizontalContentAlignment="Center" VerticalContentAlignment="Center"
></TextBox>
    <TextBox Name="AddrDataTB" KeyboardNavigation.TabIndex="1"
HorizontalContentAlignment="Center" VerticalContentAlignment="Center"
></TextBox>
    <TextBox Name="phDataTB" KeyboardNavigation.TabIndex="2"
VerticalContentAlignment="Center" HorizontalContentAlignment="Center"
></TextBox>
    <TextBox Name="vatdataTB" KeyboardNavigation.TabIndex="3"
VerticalContentAlignment="Center" HorizontalContentAlignment="Center"
></TextBox>
    </UniformGrid>
</DockPanel>

</Window>

```

PatientRegistration.java

```

package HospitalManagementSystem;

/// <summary>
/// Interaction logic for PatientRegistration.html
/// </summary>
public partial class PatientRegistration : Window
{
    public PatientRegistration()
    {
        InitializeComponent();
    }

    private void cancelBtn_Click(object sender, RoutedEventArgs e)
    {
    }

    private void addBtn_Click(object sender, RoutedEventArgs e)
    {
    }
}

```

PatientsBrowser.html

```

<Window x:Class="HospitalManagementSystem.PatientsBrowser"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

```

```

        Title="Patients Browser" Height="635" Width="1020">
        <DockPanel LastChildFill="True">
            <UniformGrid Height="30" DockPanel.Dock="Top" Rows="1">
                <Button Content="Show patients" Name="showPaymentsBtn"
Click="showPaymentsBtn_Click" Margin="10 0" ></Button>
                <Button Content="Add New" Name="addBtn" Click="addBtn_Click"
Margin="10 0" ></Button>
                <Button Content="Delete" Margin="10 0" Name="deleteBtn"
Click="deleteBtn_Click"></Button>
                <Button Content="Print" Margin="10 0" Name="printBtn"
IsEnabled="True" Click="printBtn_Click_1"></Button>
            </UniformGrid>
            <ListView Name="businessPersonListView" DockPanel.Dock="Bottom"
IsSynchronizedWithCurrentItem="True">
                <ListView.View>
                    <GridView>
                        <!-- GridViewColumn Width="50" Header="Sl No"
DisplayMemberBinding="{Binding serialNo}" /-->
                        <GridViewColumn Width="150" Header="Id"
DisplayMemberBinding="{Binding patientId}" />
                        <GridViewColumn Width="260" Header="Name"
DisplayMemberBinding="{Binding patientName}" />
                        <GridViewColumn Width="160" Header="Address"
DisplayMemberBinding="{Binding patientAdress}" />
                        <GridViewColumn Width="100" Header="Ph no."
DisplayMemberBinding="{Binding phoneNumber}" />
                        <GridViewColumn Width="100" Header="Doctor"
DisplayMemberBinding="{Binding patientVatNo}" />
                        <GridViewColumn Width="100" Header="Problems"
DisplayMemberBinding="{Binding patientTurnOver}" />
                        <GridViewColumn Width="100" Header="Bill Amount"
DisplayMemberBinding="{Binding patientDue}" />
                    </GridView>
                </ListView.View>
            </ListView>
        </DockPanel>
    </Window>

```

PatientsBrowser.java

```

package HospitalManagementSystem;

/// <summary>
/// Interaction logic for PatientsBrowser.html
/// </summary>
public partial class PatientsBrowser : Window
{
    public PatientsBrowser()
    {
        InitializeComponent();
    }

    private void printBtn_Click(object sender, RoutedEventArgs e)
    {
    }
}

```

```

        private void showPaymentsBtn_Click(object sender, RoutedEventArgs e)
        {

        }

        private void deleteBtn_Click(object sender, RoutedEventArgs e)
        {

        }

        private void printBtn_Click_1(object sender, RoutedEventArgs e)
        {

        }

        private void addBtn_Click(object sender, RoutedEventArgs e)
        {

        }
    }
}

```

RestfulClient.java

```

package com.hms.client;

import javax.ws.rs.core.MediaType;

import org.apache.log4j.chainsaw.Main;

import com.hms.model.DoctorInfo;
import com.hms.model.Response;
import com.sun.jersey.api.client.Client;
import com.sun.jersey.api.client.ClientResponse;
import com.sun.jersey.api.client.WebResource;

public class RestfulClient {

    public void invokeClient(){

        DoctorInfo doc= new DoctorInfo();
        doc.setIddoctors(300);
        doc.setAddress("Govind Address");
        doc.setContactNumber("9611125680");
        doc.setName("Govind Reddy");
        try{
            Client client = Client.create();
            WebResource webResource =
client.resource("http://localhost:8080/hmsapp/hms/findDoctorByID");
            ClientResponse response =
webResource.accept(MediaType.APPLICATION_XML,
MediaType.APPLICATION_JSON).post(ClientResponse.class, doc);

```



```

        DoctorInfo doc1 = response.getEntity(DoctorInfo.class);

        System.out.println("Name" + doc1.getName());

    }catch (Exception e) {
        // TODO: handle exception
        e.printStackTrace();
    }

    }

    public static void main(String[] args) {
        RestfulClient client= new RestfulClient();
        try {
            client.findDoctorByID();
            //client.addOrUpdateDoctor();
        } catch (Exception e) {
            // TODO: handle exception
            e.printStackTrace();
        }

    }

    private void addOrUpdateDoctor() {

        DoctorInfo doc= new DoctorInfo();
        //doc.setIddoctors(300);
        doc.setAddress("Govind Address");
        doc.setContactNumber("9611125680");
        doc.setName("Govind Reddy");
        try{
            Client client = Client.create();
            WebResource webResource =
client.resource("http://localhost:8080/hmsapp/hms/addOrUpdateDoctor");
            ClientResponse response =
webResource.accept(MediaType.APPLICATION_XML,
MediaType.APPLICATION_JSON).post(ClientResponse.class, doc);

            Response serviceResponse =
response.getEntity(Response.class);
            if(serviceResponse != null){

                System.out.println(serviceResponse.getStatus());

            }
        }catch (Exception e) {
            // TODO: handle exception
            e.printStackTrace();
        }

    }

}

```

```

private void findDoctorByID() {

    DoctorInfo doc= new DoctorInfo();
    doc.setIddoctors(300);
    doc.setAddress("Govind Address");
    doc.setContactNumber("9611125680");
    doc.setName("Govind Reddy");
    try{
        Client client = Client.create();
        WebResource webResource =
client.resource("http://localhost:8080/hmsapp/hms/findDoctorByID");
        ClientResponse response =
webResource.accept(MediaType.APPLICATION_XML,
MediaType.APPLICATION_JSON).post(ClientResponse.class, doc);

        DoctorInfo doc1 = response.getEntity(DoctorInfo.class);

        System.out.println("Name" + doc1.getName());
        /*Response serviceResponse =
response.getEntity(Response.class);
        if(serviceResponse != null){

            System.out.println(serviceResponse.getDoctorInfoList().size());

        }*/
    }catch (Exception e) {
        // TODO: handle exception
        e.printStackTrace();
    }

}

}
}

```

ProviderController.java

```

package com.hms.controller;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.bind.annotation.RestController;

import com.hms.model.DoctorInfo;
import com.hms.model.Response;
import com.hms.service.ProviderService;

@RestController
@RequestMapping(value = "/hms")
public class ProviderController {

    @Autowired

```

```

        ProviderService providerService;

        @RequestMapping(value="/addOrUpdateDoctor", method = RequestMethod.POST
,headers="Accept=application/json,application/xml")
        public @ResponseBody Response addDoctor(@RequestBody DoctorInfo doctorInfo){
            System.out.println("inside addOrUpdateDoctor method ");
            Response response=null;
            try {

                response = providerService.addDoctor(doctorInfo);

            } catch (Exception e) {

                e.printStackTrace();
            }
            return response;
        }

        @RequestMapping(value="/deleteDoctor", method = RequestMethod.POST
,headers="Accept=application/json,application/xml")
        public @ResponseBody Response deleteDoctor(@RequestBody DoctorInfo doctorInfo){
            System.out.println("inside deleteDoctor method ");
            Response response=null;
            try {

                response = providerService.deleteDoctor(doctorInfo);

            } catch (Exception e) {

                e.printStackTrace();
            }
            return response;
        }

        @RequestMapping(value = "/findDoctorByAttributes" , method =
RequestMethod.POST,headers="Accept=application/json,application/xml")
        public @ResponseBody Response findDoctorstByAttributes(@RequestBody DoctorInfo
doctorInfo){
            System.out.println("inside findDoctorByAttributes method ");
            Response response=null;
            try {

                response = providerService.getDoctorsByAttributes(doctorInfo);
            } catch (Exception e) {
                e.printStackTrace();
            }
            return response;
        }

        @RequestMapping(value = "/findDoctorByID" , method =
RequestMethod.POST,headers="Accept=application/json,application/xml")
        public @ResponseBody DoctorInfo findDoctorByID(@RequestBody DoctorInfo doctorInfo){
            System.out.println("inside findDoctorByID method ");
            DoctorInfo response=null;
            try {

```

```

        response = providerService.getDoctorByID(doctorInfo);
    } catch (Exception e) {
        e.printStackTrace();
    }
    return response;
}
}

```

ProviderRequestResponseMapper.java

```

/**
 *
 */
package com.hms.mapper;

import java.util.ArrayList;
import java.util.List;

import org.springframework.stereotype.Component;

import com.hms.entity.Doctor;
import com.hms.model.DoctorInfo;

@Component
public class ProviderRequestResponseMapper {

    public Doctor getDoctorEntity(DoctorInfo doctorInfo){
        Doctor doc =new Doctor();
        doc.setIddoctors(doctorInfo.getIddoctors());
        doc.setAddress(doctorInfo.getAddress());
        doc.setContactNumber(doctorInfo.getContactNumber());
        doc.setDoctorRegNumber(doctorInfo.getDoctorRegNumber());
        doc.setDoctorSpecialty(doctorInfo.getDoctorSpecialty());
        doc.setName(doctorInfo.getName());
        return doc;
    }

    public List<DoctorInfo> getDoctorInfoList(List<Doctor> doctorList) {
        // TODO Auto-generated method stub

        List<DoctorInfo> doctorInfoList= new ArrayList<DoctorInfo>();

        for (int i = 0; i < doctorList.size(); i++) {
            Doctor doc= doctorList.get(i);

            doctorInfoList.add(getDoctorInfoFromEntity(doc));
        }
        return doctorInfoList;
    }

    public DoctorInfo getDoctorInfoFromEntity(Doctor doc) {
        DoctorInfo doctorInfo = new DoctorInfo();
        doctorInfo.setAddress(doc.getAddress());
        doctorInfo.setContactNumber(doc.getContactNumber());
    }
}

```

```

        doctorInfo.setDoctorRegNumber(doc.getDoctorRegNumber());
        doctorInfo.setDoctorSpecialty(doc.getDoctorSpecialty());
        doctorInfo.setName(doc.getName());
        doctorInfo.setIddoctors(doc.getIddoctors());
        return doctorInfo;
    }
}

```

DoctorInfoModel.java

```

package com.hms.model;

import java.io.Serializable;
import javax.xml.bind.annotation.XmlRootElement;
import org.springframework.stereotype.Component;

@XmlRootElement (name = "DoctorInfo")
@Component
public class DoctorInfo implements Serializable{

    /**
     *
     */
    private static final long serialVersionUID = 1L;

    private String address;

    private Integer iddoctors;

    private String contactNumber;

    private String doctorRegNumber;

    private String doctorSpecialty;

    private String name;

    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }

    public String getContactNumber() {
        return contactNumber;
    }
}

```

```

    public void setContactNumber(String contactNumber) {
        this.contactNumber = contactNumber;
    }

    public String getDoctorRegNumber() {
        return doctorRegNumber;
    }

    public void setDoctorRegNumber(String doctorRegNumber) {
        this.doctorRegNumber = doctorRegNumber;
    }

    public String getDoctorSpecialty() {
        return doctorSpecialty;
    }

    public void setDoctorSpecialty(String doctorSpecialty) {
        this.doctorSpecialty = doctorSpecialty;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public Integer getIddoctors() {
        return iddoctors;
    }

    public void setIddoctors(Integer iddoctors) {
        this.iddoctors = iddoctors;
    }
}

```

PatientInfoModel.java

```

package com.hms.model;

import java.io.Serializable;

import javax.xml.bind.annotation.XmlRootElement;

import org.springframework.stereotype.Component;

```

```

@XmlRootElement (name = "PatientInfo")
@Component
public class PatientInfo implements Serializable{

    /**
     *
     */
    private static final long serialVersionUID = 1L;

    private String address;

    private String contactNumber;

    private String govtPhotoId;

    private String name;

    private String photoPath;

    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }

    public String getContactNumber() {
        return contactNumber;
    }

    public void setContactNumber(String contactNumber) {
        this.contactNumber = contactNumber;
    }

    public String getGovtPhotoId() {
        return govtPhotoId;
    }

    public void setGovtPhotoId(String govtPhotoId) {
        this.govtPhotoId = govtPhotoId;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getPhotoPath() {
        return photoPath;
    }

    public void setPhotoPath(String photoPath) {
        this.photoPath = photoPath;
    }

}

```

ResponseModel.java

```
/**
 *
 */
package com.hms.model;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlRootElement;

import org.springframework.stereotype.Component;

/**
 * @author govind
 *
 */
@XmlRootElement
@Component
@XmlAccessorType(XmlAccessType.PROPERTY)
public class Response implements Serializable{

    public Response(){

    }

    /**
     *
     */
    private static final long serialVersionUID = 1L;

    private String code;

    private String status;

    private List<PatientInfo> patientInfoList = new ArrayList<PatientInfo>();
    private List<DoctorInfo> doctorInfoList = new ArrayList<DoctorInfo>();

    public String getCode() {
        return code;
    }
    public void setCode(String code) {
        this.code = code;
    }
}
```



```

    public String getStatus() {
        return status;
    }
    public void setStatus(String status) {
        this.status = status;
    }
    public List<PatientInfo> getPatientInfoList() {
        return patientInfoList;
    }
    public void setPatientInfoList(List<PatientInfo> patientInfoList) {
        this.patientInfoList = patientInfoList;
    }
    public List<DoctorInfo> getDoctorInfoList() {
        return doctorInfoList;
    }
    public void setDoctorInfoList(List<DoctorInfo> doctorInfoList) {
        this.doctorInfoList = doctorInfoList;
    }
}

```

ProviderService.java

```

/**
 *
 */
package com.hms.service;

import org.springframework.stereotype.Component;

import com.hms.entity.Doctor;
import com.hms.model.DoctorInfo;
import com.hms.model.Response;

@Component
public interface ProviderService {

    public Response addDoctor(DoctorInfo doctorInfo);

    public Response deleteDoctor(DoctorInfo doctorInfo);

    public Response getDoctorsByAttributes(DoctorInfo doctorInfo);

    public DoctorInfo getDoctorByID(DoctorInfo doctorInfo);

}

```

ProviderServiceImpl.java

```

/**
 *
 */
package com.hms.service.impl;

```

```

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Repository;

import com.hms.dao.ProviderDAO;
import com.hms.entity.Doctor;
import com.hms.mapper.ProviderRequestResponseMapper;
import com.hms.model.DoctorInfo;
import com.hms.model.Response;
import com.hms.service.ProviderService;

/**
 * @author govind
 *
 */
@Repository
public class ProviderServiceImpl implements ProviderService {

    @Autowired
    ProviderDAO providerDAO;
    @Autowired

    ProviderRequestResponseMapper providerRequestResponseMapper;

    /* (non-Javadoc)
     * @see com.hms.service.ProviderService#addDoctor(com.hms.model.DoctorInfo)
     */
    @Override
    public Response addDoctor(DoctorInfo doctorInfo) {
        // TODO Auto-generated method stub
        return
providerDAO.addDoctor(providerRequestResponseMapper.getDoctorEntity(doctorInfo));
    }

    /* (non-Javadoc)
     * @see com.hms.service.ProviderService#getDoctorsByAttributes(com.hms.model.DoctorInfo)
     */
    @Override
    public Response getDoctorsByAttributes(DoctorInfo doctorInfo) {
        // TODO Auto-generated method stub
        Response response= new Response();
        List<Doctor> doctorList=
providerDAO.getDoctorsByAttributes(providerRequestResponseMapper.getDoctorEntity(doctorInfo));
        if(doctorList.size()>0){
            List<DoctorInfo> doctorInfoList=
providerRequestResponseMapper.getDoctorInfoList(doctorList);
            response.setDoctorInfoList(doctorInfoList);
        }

        return response;
    }

    @Override
    public Response deleteDoctor(DoctorInfo doctorInfo) {
        return
providerDAO.deleteDoctor(providerRequestResponseMapper.getDoctorEntity(doctorInfo));
    }

    @Override
    public DoctorInfo getDoctorByID(DoctorInfo doctorInfo) {

```

```
        // TODO Auto-generated method stub
        Doctor doc =
        providerDAO.getDoctorByID(providerRequestResponseMapper.getDoctorEntity(doctorInfo));
        return providerRequestResponseMapper.getDoctorInfoFromEntity(doc);
    }
}
```

TESTING

UNIT TESTCASES:

Test case Id	Description	Prerequisite	Steps to produce
HMS_UT_001	Load a Prescription	Valid prescription and a software	Launch the software double click on left navigation pan
HMS_UT_002	Login as a doctor	Valid user and a software	Launch the software double click on left navigation pan
HMS_UT_003	Display Image	Valid user with Image and ID and a software	Launch the software double click on left navigation pan
HMS_UT_004	Display Video	Valid prescription with video and a software	Launch the software click on video tab
HMS_UT_005	Video Play	Valid prescription with video and a software	Launch the software click on video tab
HMS_UT_006	Video Pause	Valid prescription and a software	Launch the software click once
HMS_UT_007	Video Stop	Valid prescription and a software	Launch the software click on stop button
HMS_UT_008	Video Fast Forward	Valid prescription and a software	Launch the software click on fast forward button
HMS_UT_009	Video Rewind	Valid prescription and a software	Launch the software click on back button
HMS_UT_010	Video Open	Valid prescription and a software	Launch the software click on open button
HMS_UT_011	Display Online Prescription	Valid prescription and a software	Launch the software.click on online button (Online prescription Navigation pan)
HMS_UT_013	Go to Online prescription Link	Valid prescription and a software	Launch the software click on buy button left navigation pan
HMS_UT_014	Sound On/Off	Valid prescription and a software	Launch the software click on sound on/off button
HMS_UT_015	Local prescription Display	Valid prescription and a software	Launch the software click on open button to browse the saved prescriptions.
HMS_UT_016	Menu items functionality	Valid prescription and a software	Launch the software click on menubar
HMS_UT_017	Toolbar item functionality	Valid prescription and a software	Launch the software click on toolbar buttons
HMS_UT_018	Register as a new user	Valid ID and a software	Launch the software click on toolbar and register buttons(create an account)
HMS_UT_019	Login as a User	Valid ID and a software	Launch the software click on login buttons
HMS_UT_020	Register as a new User	Valid ID and a software	Launch the software click on toolbar and register

			buttons(create an account) at admin section
HMS_UT_021	Membership card issuing	Valid ID and a software	Launch the software click on toolbar and register buttons(create an account)and click to issue membership.
HMS_UT_022	Check Data Usages	Valid ID and a software	Launch the software click on Data Usage toolbar
HMS_UT_023	Print report from Local site	Valid ID and a software	Launch the software click on print button
HMS_UT_024	Print report from Central site	Valid ID and a software	Launch the software.click on print button from central machine
HMS_UT_025	Check availability information	Valid ID and a software	Launch the software click on navigation menu
HMS_UT_026	Search a item	Valid ID and a software	Launch the software click on search
HMS_UT_027	Arrangement of searching items	Valid ID and a software	Launch the software click on search

INTEGRATION TESTCASES:

Test case Id	Description	Prerequisite	Steps to produce
HMS_IT_001	Completely Run a prescription online	Installed software and a valid and prescription	Launch the software double click on left navigation pan
HMS_IT_002	Run a Saved Prescription	Installed software and a valid and prescription	Launch the software double click on left navigation pan and click on open button to browse the saved prescription
HMS_IT_003	Run a downloaded Prescription	Installed software and a valid and prescription and an internet connection	Launch the software Download a prescription from online store.

HMS_IT_004	Check the installer	Installed software	Launch the software double click on icon
HMS_IT_005	Track resource usage	Installed software and a ID ,prescription and an internet connection	Launch the software double click on icon
HMS_IT_006	Check membership	Installed software And logged in as a valid user	Launch the software Click on login button
HMS_IT_007	Check admin authentication	Installed software And logged in as a valid admin	Launch the software Click on login button
HMS_IT_008	Local machine connectivity with central machine	Installed software With a configured LAN connection	Launch the software Click the connect button
HMS_IT_009	Check data usage	Installed software With a configured LAN connection	Launch the software Click the connect button and Data usage button
HMS_IT_010	Search an user or prescription	Installed software And logged in as a valid user	Launch the software Click on search menu

SYSTEM TESTCASES:

Test case Id	Description	Prerequisite	Steps to produce
HMS_ST_001	Install the software	HMS installer,.net frame work	Double click on the setup file
HMS_ST_002	Launch the software from desktop shortcut	HMS installed ,.net frame work	Double click on the HMS icon
HMS_ST_003	Launch the software from program menus	HMS installed,.net frame work	Navigate from start and click on HMS icon
HMS_ST_004	Run a prescription	HMS installed,.net frame work	Launch the software double click on left navigation pan and click on open button to browse the saved prescription
HMS_ST_005	Close the Software	HMS installed,.net	Press alt+f4 or click on close button

		frame work , launched software	
HMS_ST_006	Uninstall the Software	HMS installed,.net frame work	Open control panel and uninstall the program.
HMS_ST_007	Download a prescription	HMS installed,.net frame work,valid prescription	Launch the software Download a prescription from online store.
HMS_ST_008	Run a Video report	HMS installed,.net frame work, valid prescription with video	Launch the software click on open button
HMS_ST_009	Run a Image Prescription	HMS installed,.net frame work, valid prescription with image	Launch the software click on open button
HMS_ST_010	Run a search	HMS installed,.net frame work, valid prescription with image video text	Launch the software click on open button

UNIT TEST RESULT:

UNIT TESTID	Expected Result	Actual Result	Comment	Status
HMS_UT_001	Load a prescription	Load a prescription	NA	PASS
HMS_UT_002	Logged in as a user	Logged in successfully	NA	PASS
HMS_UT_003	Display Image	Display Image	NA	PASS
HMS_UT_004	Display Video	Display Video	NA	PASS
HMS_UT_005	Video Play	Video Play	NA	PASS
HMS_UT_006	Video Pause	Video Pause	NA	PASS
HMS_UT_007	Video Stop	Video Stop	NA	PASS
HMS_UT_008	Video Fast Forward	Video Fast Forward	NA	PASS
HMS_UT_009	Video Rewind	Video Rewind	NA	PASS

HMS_UT_010	Video Open	Video Open	NA	PASS
HMS_UT_011	Display Online Prescription or prescription	Display Online Prescription or prescription	NA	PASS
HMS_UT_013	Go to Online prescription Link	Go to Online prescription Link	NA	PASS
HMS_UT_014	Sound On/Off	Sound On/Off	NA	PASS
HMS_UT_015	Local prescription Display	Local prescription Display	NA	PASS
HMS_UT_016	Menu items functionality working properly	Menu items functionality working properly	NA	PASS
HMS_UT_017	Toolbar item functionality	Toolbar item functionality working properly	NA	PASS
HMS_UT_018	Registered as a new user	Registered as a new user	NA	PASS
HMS_UT_019	Logged in as a employee	Logged in as a employee	NA	PASS
HMS_UT_020	Registered as a new employee	Registered as a new employee	NA	PASS
HMS_UT_021	Membership card issued	Membership card issued	NA	PASS
HMS_UT_022	Display Data Usages	Display Data Usages	NA	PASS
HMS_UT_023	Print report from Local site	Print report from Local site	NA	PASS
HMS_UT_024	Print report from Central site	Print report from Central site	NA	PASS
HMS_UT_025	Show availability information	Show availability information	NA	PASS
HMS_UT_026	Found a item	Found a item	NA	PASS
HMS_UT_027	Arranged in an ascending order	Arranged in an ascending order	NA	PASS

INTEGRATION TEST RESULT:

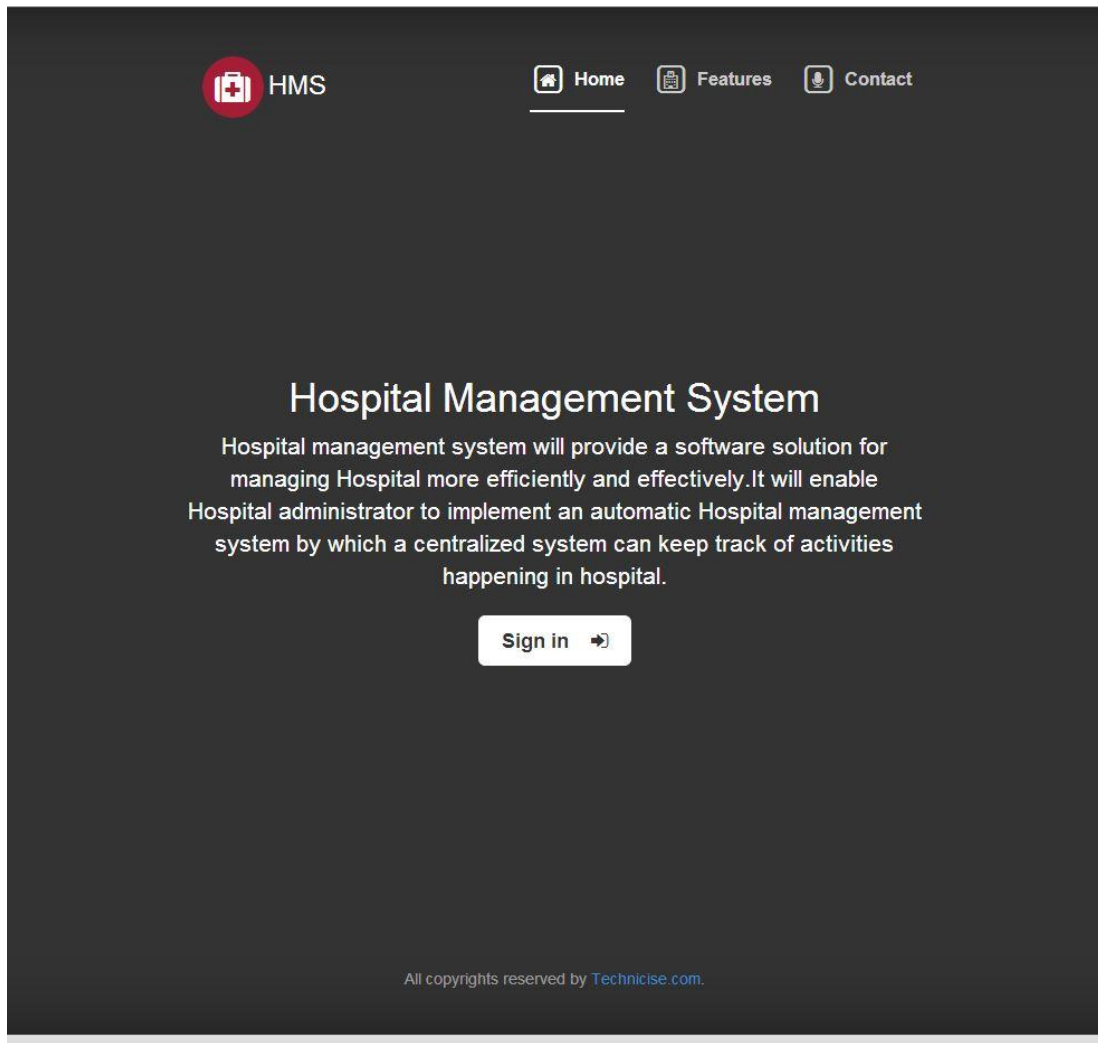
Test case Id	Expected Result	Actual Result	Comment	Status
HMS_IT_001	Completely Run a prescription online	Completely Run a Prescription	NA	PASS

HMS_IT_002	Run a Saved Prescription	Run a Saved Prescription	NA	PASS
HMS_IT_003	Run a downloaded Prescription	Run a downloaded Prescription	NA	PASS
HMS_IT_004	Check the installer	Check the installer	NA	PASS
HMS_IT_005	Track resource usage	Track resource usage	NA	PASS
HMS_IT_006	Validated membership	Validated membership	NA	PASS
HMS_IT_007	admin authenticated successfully	admin authenticated successfully	NA	PASS
HMS_IT_008	Local machine connected with central machine	Local machine connected with central machine	NA	PASS
HMS_IT_009	Show data usage	Show data usage	NA	PASS
HMS_IT_010	Found a valid user or prescription	Found a valid user or prescription	NA	PASS

SYSTEM TEST RESULT:


Test case Id	Expected Result	Actual Result	Comment	Status
HMS_ST_001	Install the software	Install the software	NA	PASS
HMS_ST_002	Launch the software from desktop shortcut	Launch the software from desktop shortcut	NA	PASS
HMS_ST_003	Launch the software from program menus	Launch the software from program menus	NA	PASS
HMS_ST_004	Run a prescription	Run a prescription	NA	PASS
HMS_ST_005	Uninstall the Software	Uninstall the Software	NA	PASS
HMS_ST_006	Download a Prescription	Download a Prescription	NA	PASS
HMS_ST_007	Run a Video prescription	Run a Video prescription	NA	PASS
HMS_ST_008	Run a Image Prescription	Run a Image Prescription	NA	PASS
HMS_ST_010	Run a search	Run a search	NA	PASS


GUI Screenshots:





 [Home](#)

 [Features](#)

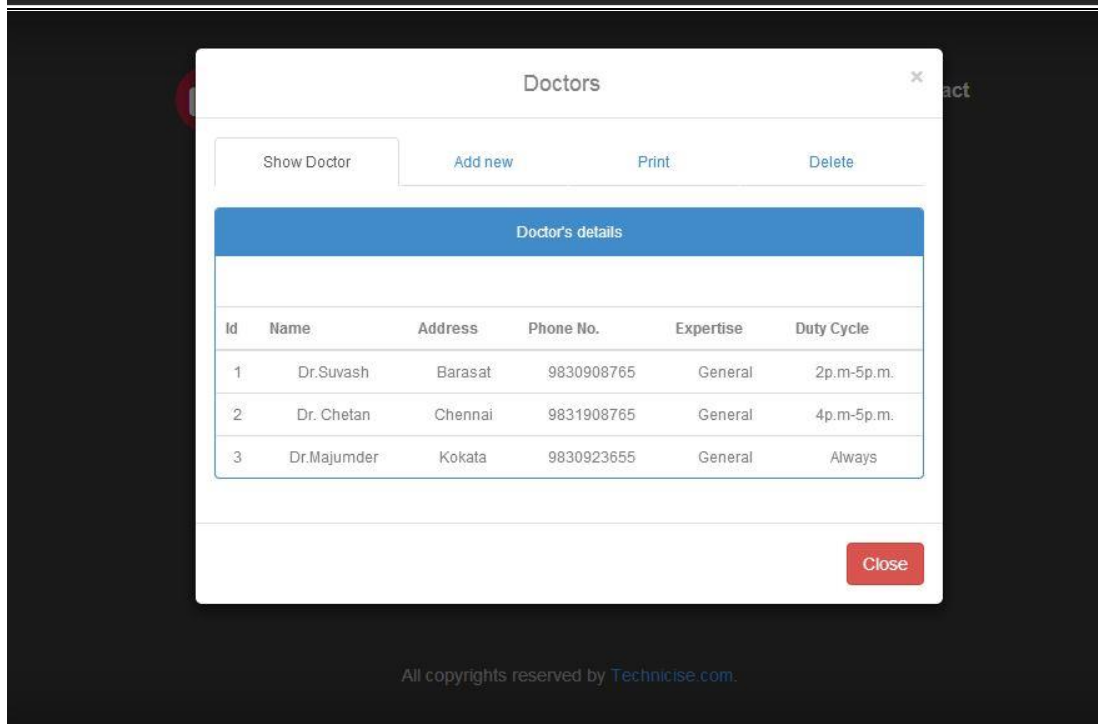
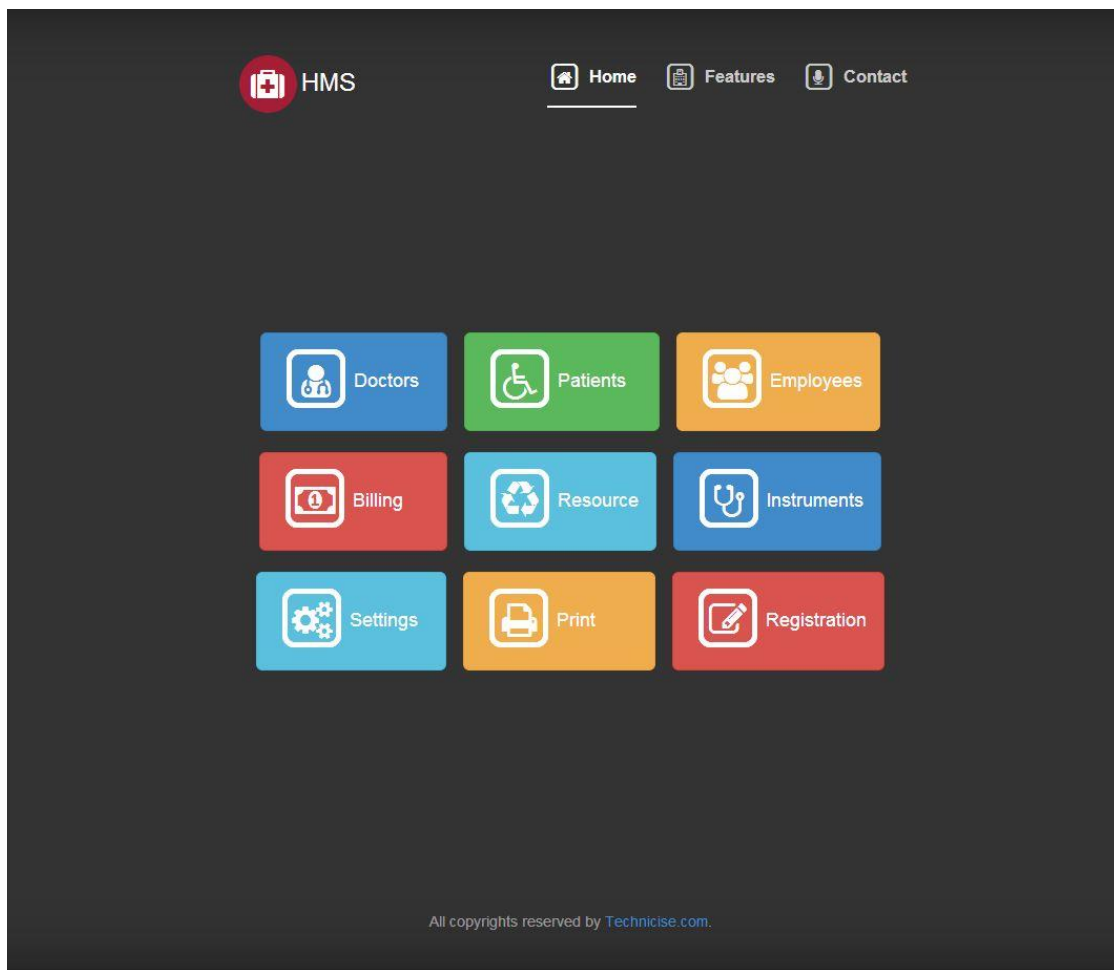
 [Contact](#)

Please sign in

☐ Remember me

[Sign in](#)

All copyrights reserved by [Technicise.com](#).



Doctors

Show Doctor

Add new

Print

Delete

Doctor's details

Doctor's Name

DoctorName

Address

Address

Phone number

Ex:9932457689

Expertise

Expertise

Duty cycle

Duty cycle

Save and continue

Close

All copyrights reserved by [Technicise.com](#).

patients

Show patient

Add new


Print

Delete

patient's details

	Id	Name	Address	Phone No.	Department	Admission Period
<input type="checkbox"/>	1	Mr. Soumen Paul	Barasat	9831908765	Cardiac Surgery	2p.m-5p.m.
<input checked="" type="checkbox"/>	2	Mr. Amiyo Biswas	Chennai	9831908765	Malnutrition	4p.m-5p.m.
<input type="checkbox"/>	3	Amtabha Majumder	Kolkata	9830923655	Kidney failure	Always
<input type="checkbox"/>	3	Rita Roy	Kolkata	9831113655	Gastric	Always

Close


HMS

patients

Show patient

Add new

Print

Delete

patient's details

patient's Name

Address

Phone number


Department

Admission Period

Save and continue

Close

All copyrights reserved by [Technicise.com](#).


HMS

employees

Show employee

Add new

Print


Delete

employee's details

Id	Name	Address	Phone No.	Expertise	Duty Cycle
1	Partha Dey	Barasat	9830128765	Front Desk	2p.m-5p.m.
2	Puspita Roy	Chennai	9831908765	Technician	4p.m-5p.m.
3	Maitri Halder	kolkata	9830923655	Nurse	Always
4	Bimal Jana	kolkata	9830923655	Male Nurse	Always
5	Amit Paul	kolkata	9830923655	Security	Always

Close

All copyrights reserved by [Technicise.com](#).

HMS

employees

Show employee

Add new

Print

Delete

employee's details

employee's Name

employeeName

Address

Address

Phone number

Ex:9932457689

Expertise

Expertise

Duty cycle

Duty cycle

Save and continue

Close

All copyrights reserved by [Technicise.com](#).

Invoice for purchase



Invoice for purchase #33221

Billing Details

Somnath Saha:
1111 Army Navy
Colony
Shyambazar
Kolkata
22 203

Payment Information

Card Name:
Visa
Card Number:
***** 332
Exp Date:
09/2020

Order Preferences

Gift: No
Express
Delivery: Yes
Insurance: No
Coupon: No

Shipping Address

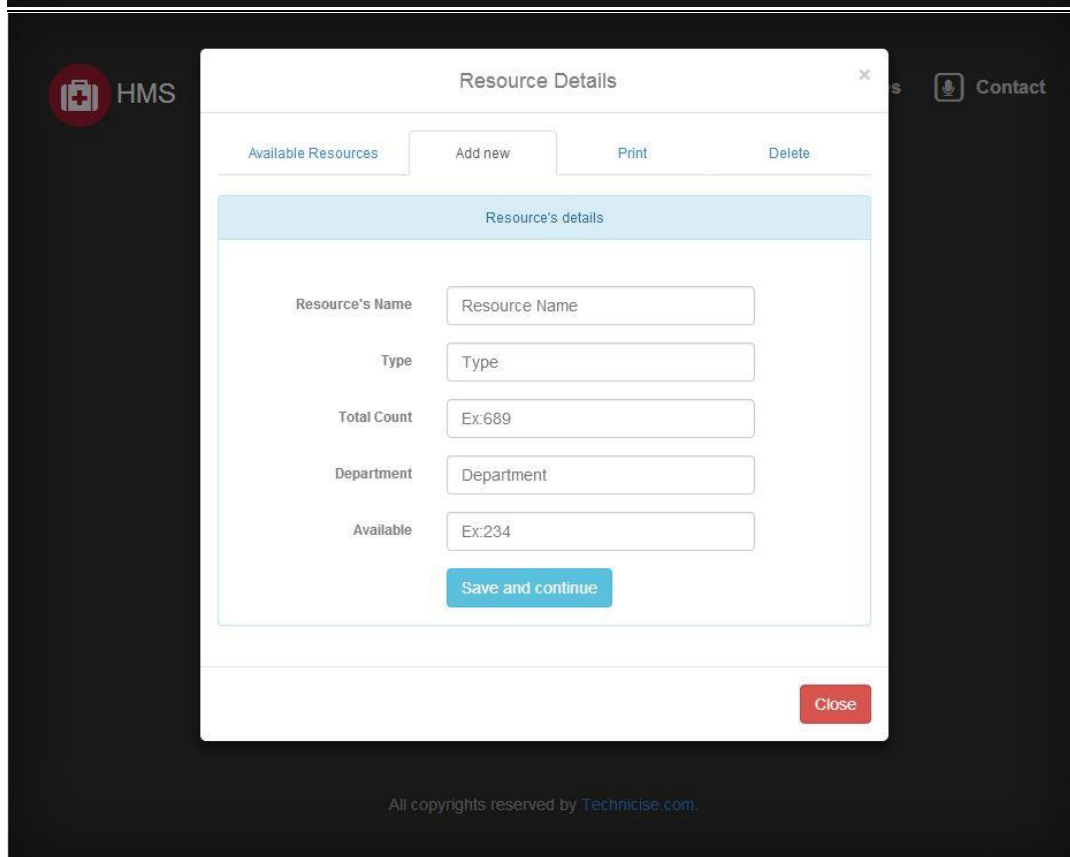
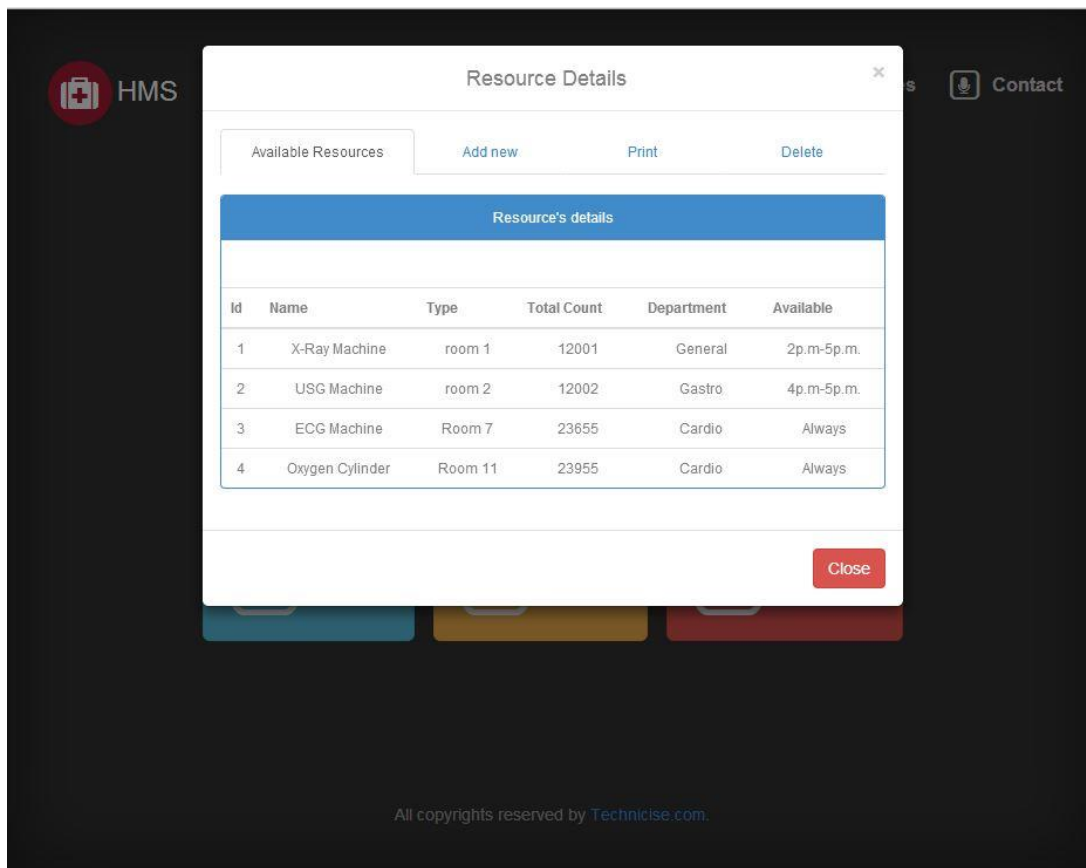
Somnath Saha:
1111 Army Navy
Colony
Shyambazar
Kolkata
22 203

Order summary

Item Name	Item Price	Item Quantity	Total
Bed rent	Rs.900	1	Rs.900
Medicine	Rs.30.00	1	Rs.30.00
Others	Rs.7	4	Rs.28
Subtotal			Rs.958.00
Shipping			Rs.20
Total			Rs.978.00



Close



Organization

Technicise.com
Bamangachi Station Road,
P.O: Bamangachi, P.S: Barasat,
Dist: North 24 Pgs
West Bengal- 743706

Future Scope and Further enhancement of the Project:

- To extend it for multiple organization with a centralized database.
- To support mobile apps for Android , iOS.

BIBLOGRAPHY:

<https://www.playframework.com/>

<http://www.oracle.com/technetwork/java/javaee/overview/index.html>

<http://hibernate.org/>

<http://www.postgresql.org/>

<http://jbossas.jboss.org/>

<http://www.oracle.com/technetwork/java/javaee/tech/persistence-jsp-140049.html>

<http://www.google.co.in>

<http://en.wikipedia.org>

<http://msdn.microsoft.com/en-us/>

<http://www.microsoft.com/en-us/default.aspx>

<http://www.codeplex.com/>

<http://stackoverflow.com/>

<http://www.codeguru.com/>

<http://www.w3schools.com>

----- Thank You -----