TRAINING REPORT

OF

SIX MONTHS INDUSTRIAL TRAINING, UNDERTAKEN

AT

ThinkNEXT Technologies

Under the Guidance of: -

Name: Shahnawaz Khan

Designation:

Department:

Submitted By: -

Name: Nishant Sharma

University ID No.:1811981207

Acknowledgement

I would like to express my gratitude towards the trainer of ThinkNext Systems for providing me a great opportunity to complete a project on HealthCare.

My sincere thanks go to Mr. Shahnawaz Khan for his support and guidance for the completion of this project.

I am ensuring that this project was done by me and not copied from anywhere.

Nishant Sharma

Preface

This project deals with the Corporate Medicare Management. This project is very helpful to both Medicare staff as well as to the public. It is having mainly Administration and Client modules.

The growing quality demand in the hospital sector makes it necessary to exploit the whole potential of stored data efficiently, not only the clinical data, in order to improve diagnoses and treatments, but also on management, in order to minimize costs and improve the care given to the patients.

Contents

1.Objective

A hospital management app can help people find doctors, coordinate their appointments, and receive the treatment they need. It can also reduce or simplify administrative tasks for physicians and boost collaboration inside medical teams – a long-term advantage for both the hospital's reputation and treatment results.

Industry Application

The management of every professional space has entirely transformed with technology. Nobody uses telephones anymore to make appointments. In the medical industry, hospital management software now come with a scheduling module. It allows patients to book their own appointments online with 24/7 access to a patient portal.

Another feature of this module is automated alerts when routine preventive care appointments have been due. For instance, if somebody’s medicine prescription is very close to expiring, they would get an automated email from their doctor. It’s helpful in promoting the number one thing people must do to stay healthy: get consistent preventive care.

2. INTRODUCTION

This project deals with the Corporate Medicare Management. This project is very helpful to both Medicare staff as well as to the public. It is having mainly Administration and Client modules.

The growing quality demand in the hospital sector makes it necessary to exploit the whole potential of stored data efficiently, not only the clinical data, in order to improve diagnoses and treatments, but also on management, in order to minimize costs and improve the care given to the patients.

In this sense, Data Mining (DM) can contribute with important benefits to the health sector, as a fundamental tool to analyze the data gathered by hospital information systems (HIS) and obtain models and patterns which can improve patient assistance and a better use of resources and pharmaceutical expense.

Data Mining is the fundamental stage inside the process of extraction of useful and comprehensible knowledge, previously unknown, from large quantities of data stored in different formats, with the objective of improving the decisions of companies, organizations or institutions where the data have been gathered.

However, data mining and the overall process, known as Knowledge Discovery from Databases (KDD), is usually an expensive process, especially in the stages of business objectives elicitation, data mining objectives elicitation, and data preparation. This is especially the case each time data mining is applied to a hospital: many meetings have to been held with the direction of the hospital, area coordinators, computer scientists, etc., to establish the objectives, prepare the data, the mining views and for training the users to general DM tools.

2.1 PURPOSE

In Medicare management situations we are dealing with Data Mining objectives such as:

1. To optimize bed occupation.
2. To improve the use of operation theaters, avoiding the cancellation of operations.
3. To know how emergencies affect to the administration of the hospital departments or services (cancellation of operations, etc).
4. To optimize the allocation of human and material resources towards and shifts.
5. To detect the influence of certain diseases in the hospital's services.
6. To find clusters of patients.

2.2 SCOPE

2.2.1 Existing System Features

- Integration of Corporate Medicare centers is very difficult while it is having different branches.
- In most of the cases the database is similar from one hospital to another hospital. In those cases also we can't easily adapt a new technology in the new hospital.
- It is very difficult to analyze the usage percentage of hospital resources, Bed occupation Ratio, Administration, Laboratory information even in a single center. Then we can expect the complexity while integrating multi multi-specialty Medicare Centers.
- Room Reservations, Doctor Appointment Schedules, Operation Schedules, and Medicine indentation information is very difficult to maintain and share among the different Medicare Centers.
- Lack of generic and unique model we have to implement the same set of data model for every newly established Medicare Center.

2.2.2 Proposed System Features

In this project we are trying to implement which parts of a data-mining project for hospital management are equal or highly similar across different hospitals (at least in the same national healthcare system). This allows us to design several data mining modules, which can be portable across several hospitals, thus dramatically reducing the time to implement a data-mining program in a new hospital.

2.3 Structure of Automated Tool for Medicare

CRISP-DM (Standard Cross-Industry Process for Data Mining), is a consortium of companies (initially granted by the European Commission) which has defined and validated a data mining process that is applicable to several industry sectors. The following Figure 1 shows the different stages of this process:

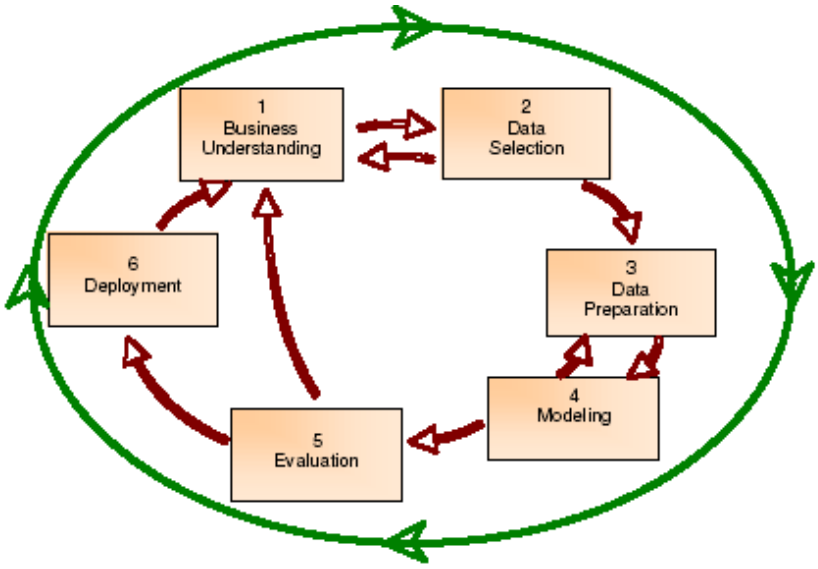


Figure 1: Life cycle of a data mining project.

The initial stage (Business Understanding) focuses on identifying the problems we are trying to solve through DM (i.e., the business objectives are defined). In our area of interest (healthcare), some hospital management objectives might be: to improve the use of hospital resources, to avoid bed occupation greater than 100% or to plan the schedule for using the

operating theatre more intensively. These objectives are defined by the people in charge of the hospital management, and then they have to be converted into data mining objectives.

For instance, some data mining objectives defined from the business objectives mentioned above are: to obtain a predictive model of hospital bed occupation, to predict the stay time of a patient depending on their disease, to establish models for estimating operations with higher cancellation or delay probability, etc. Objectives like these are of general interest for improving the management of any hospital independently of whether it is a general or a specialized health centre.

So these objectives could be included as an initial set of generic objectives in an automated data mining tool specially developed for this area. Something similar occurs with respect to the data that could be relevant for the hospital management: they are usually gathered for every centre. For instance, admission date, admission cause, discharge data, medical service assigned at the admission time, etc.

The main difference between hospitals is the format in which this information is stored in the DBMS. This fact makes it possible to (semi-)automate the rest of the life cycle stages. Hence, for stage 2, we only need to characterize the data load process from the particular HIS to the data warehouse (D.W.) for collecting all data needed for the data mining process. Likewise, regarding the data preparation stage, the same transformation processes (construction of new attributes, grouping continuous data in ranges, etc…) will be applicable for any HIS since all of them work with the same kind of data.

In general, stages 4 to 6 can also be done in an automated way since those generated models which are of interest for a hospital probably are also of interest for another one, and so on.

2.4 Data mining Tool for Hospital Management

Taken all of these considerations into account, we propose the following general scheme for an automated data mining tool for hospital management (Figure 2).

The tool is composed by several processes (modules) that correspond to the stages described in Figure 2. Thus, the load process corresponds to stages 2 and 3 (as we have discussed before).

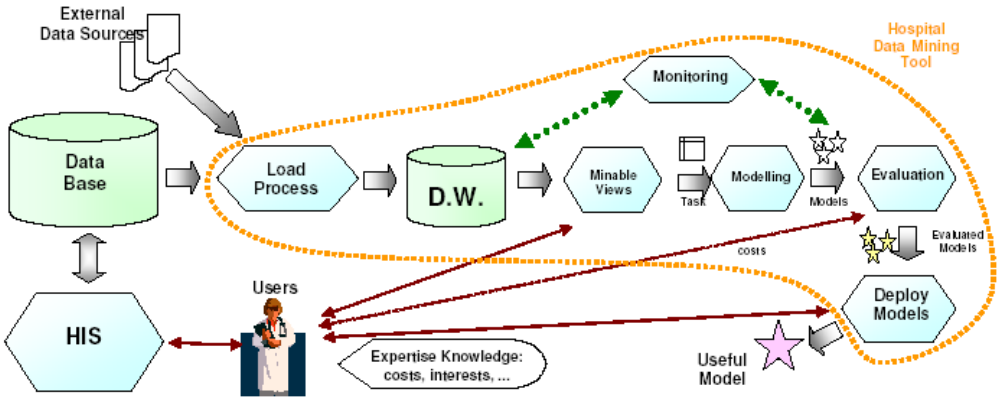


Figure 2: Data Mining Tool for Hospital Management.

The Minable View process integrates the business objectives in order to select from the D.W. the data to be used for constructing the models. Finally, the following processes (Modeling, Evaluation and Deploy Models) represent stages 4, 5 and 6, respectively.

2.5 Transforming Objectives into DM Objectives

Most of these objectives are related to emergency hospitalizations since it is a special service whose medical treatments and procedures cannot be usually delayed. Also, these objectives are interrelated. For example, if the bed occupation is closer 100%, it is necessary to cancel operations previously planned. If the operations are frequently cancelled, then the waiting lists are increased.

Now, the previous objectives have to be transformed into Data Mining objectives, such as:

1. To carry out global models about pressure emergencies by different time periods (daily, by shifts of work, by day of the week, etc).
2. To generate a model for predicting the number of daily hospitalizations coming from emergencies.
3. To obtain predictive models of global and partial use of beds by hospital service.
4. To construct models for estimating how the resources of a hospital are affected by a certain disease (for instance, influenza).
5. To carry out models to cluster patients (by age, by area, by pathology class, etc).

2.5.1Data Integration

For solving the data mining objectives, we need two kinds of information: internal (contained in the HIS) and external (not contained in the HIS). Internal information changes from one hospital to another, but for example, all of them collect general data from patients and their treatments. External data are not easy to obtain, because they are not gathered in any database.

In the area we're focusing on in this project, emergencies, we implemented the following integration:

- For internal data, our system gathers the personal patient details which are usually present in any hospital, sex, birthday date, country and living area. It is also fed by information about the patient workflow: admission date and time, reason of admission, discharge date and time, discharge code from emergencies, code of the medical service assigned at the admission time, initial diagnosis, final diagnosis, etc.

- For the external data, we gather the following data (different for each hospital, since this is geographically dependent): meteorological data (temperature, quantity of rain, wind speed, etc), lunar stage, character of the day (holiday, before holiday or after holiday, and also the festivals in the city, etc., important events, for example, football matches.

1.5.2 Data Preparation

One of the main problems to apply data mining for improving the management of a hospital is the bad quality of the source data. In many cases, the collected data contain missing or anomalous values. This can be due to a wide range of reasons: many patients do not have enough time (or they are not conscious) for filling the admission form patients do not have documents when they arrive at the hospital, illegible data, bad transcriptions, repetition of values, etc. Therefore, in these contexts, a thorough data preparation stage is very important for a successful data mining process. Some processes in the data extraction phase have been adapted to particular hospitals, but many other data cleansing/preparation processes (detection of missing or anomalous values, attribute transformation, feature creation, etc.) are the same across hospitals.

On the other hand, in many cases we will find attributes containing text, for instance, an initial description of the pathology of the patient. Since this kind of attributes cannot be directly dealt with classical learning methods, we could employ retrieval information techniques to transform the text attributes in one or more discrete attributes. For instance, we could transform the attribute with the initial description of the patient's pathology into a discrete attribute with a value for the most common pathologies (flu, traumatisms…), and a value "unclassified" for the rest of cases.

Part of this preparation stage is reused from hospital to hospital, through the automation of all these processes in a data preparation module.

We implemented scripts for extracting data from the different hospitals into the Data Warehouse (DW). These scripts must be slightly different from hospital to hospital. From the DW, since the data definition (multidimensional schema) is the same for every hospital, we used SQL scripts to generate the minable views, which are exactly the same. For instance, the minable view for the emergency pressure must integrate the number of admissions per day (or per shift) and calculate means for admission numbers of the previous week. Additionally, the number of nonworking days before and after must be computed in order to get the attributes for the minable view. All these complex SQL queries are highly time-consuming. With our approach, these complex queries are 100% portable from one hospital DW to another, and all this effort is reused.

From the minable views, the data is converted into a standard format (the arff format of WEKA) by means of Python scripts. In this way, using the command-line option in WEKA, we can generate, evaluate and export the models. Then the models are applied to new data. All this process is automated. Additionally, in some cases, the predictions can be integrated into the HIS.

2.5.3 Learning the Models

Once the data have been properly filtered, cleaned and transformed, we can proceed with the induction of the prediction models. For this purpose, we employ the suite WEKA and we make our modules work with it. This suite integrates many of the most known learning techniques, as well as, several pre-processing and post-processing tools. Additionally, WEKA has been released as open source, so, if it is required, we can adapt this software for our particular requirements

.

The key point for using WEKA is the proper construction of the minable view in such a way that could be directly used by the learning methods. A standard format (arff) has been defined as a data and model file repository in WEKA. So, the idea is to generate the data in this format, and in this way we can employ all the different leaning techniques integrated in this suite.

3. SYSTEM ANALYSIS

3.1 Requirement Analysis

A requirement is a feature that must be included in the system. Before the actual design and implementation start, getting to know the system to be implemented is of prime importance.

Main emphasis should be on**:**

- Inputs enter into the system.
- Standard Encryption of Input on submit

- The outputs expected from the system.
- The people involved in the working of the system.
- The volume of DATA (INPUT) and the amount of Information (OUTPUT) that will be involved with respect to the system itself, the following facts should be taking into consideration.

The Major process involved:

- The main points of the application.
- The processing rules fort he collected data.
- The exceptions that may be present.
- That checks that should be in place in order to avoid wrong entries.

3.1.1 Software Requirement Specification

OPERATING SYSTEM	:	WIN 98/2000/XP, UNIX/LINUX
DATA BASE	:	ORACLE
SOFTWARE	:	APACHE TOMCAT
FRONT END TOOL	:	DHTML
LANGUAGE	:	JAVA
SCRIPTING LANGUAGE	:	JAVA SCRIPT
WEB COMPONENTS	:	SERVLETS, JSP
DATA MINING TOOL	:	WEKA

3.1.2 Hardware Requirements Specification

PROCESSOR	:	Pentium-IV

PROCESSOR SPEED	:	2.4GHZ
MONITOR	:	COLOR MONITOR
HARD DISK	:	40GB
RAM	:	512MB
MOUSE	:	SCROLLING MOUSE
KEY BOARD	:	MM KEY BOARD

3.1.3 Communication protocols

- TCP/IP protocol should be installed.
- Any browser should be installed (Internet explorer 6.0 or Netscape navigator 8.0)
- HTTP 1.1 should be present on the system.
- Internet connection should be present in order to access the site.
- Internal modem or NIC card should be present.

3.2 Requirement study

The origin of most software systems is in the need of a client, who either wants to automate and existing manual system or desires a new software system. The software system itself is created by the developer finally the completed system will be used by the end user. Thus, there are three major parties interested in a new system: the client, the users, and the developer. The requirements for the system that will satisfy the need of the clients and the concerns of the user have to communicate to the developer.

The problem is that the client usually does not understand software or the software development process, and the developer often does not understand the clients problem and application area. This causes a communication gap between the parties involved in the development project. A basic purpose of software requirement specification is to bridge this communication gap. SRS is the medium through which the client and the user need are

accurately specified; indeed SRS forms the basis of software development. A good SRS should satisfy all the parties-something very hard to achieve and involves trade-offs and persuasion.

3.2.1 The Requirement Process:

The main reason of modeling generally focuses on the problem structure, not its external behaviors. Consequently, thing's like user interfaces are rarely modeled, whereas thy frequently from a major components of the SRS.

Similarly performance constraints, design constraints, standards compliance, recovery, etc. are specified clearly in the SRS because the designer must know about there to properly design the system.

To properly satisfy the basic goals, an SRS should have certain properties and should contain diff type of req. A good SRS is [iEE87, IEE94]: complete if everything the software is supposed to and responses to the software to all classes of input data are specified in the SRS.

Correctness and completeness go hand in hand an SRS in unambiguous if and only if every requirement stated has one and only one interpretation, requirements often written in natural language.

An SRS is verifiable if and only if every stated requirement is verifiable. A requirement is verifiable if there exists some cost-effective process that can check whether the final software meets those requirements. An SRS is consistent if there is no requirements that with another.

Writing and SRS is an iterative process. Even when requirement of system are specified they are later modified as the need of the client change. Hence an SRS should be easy to modify. An SRS is traceable if the origin of each of its requirements is clear and if it facilitates the referencing of each requirement in future development [EEE87].

One of the most common problems in requirement specification is when some of the requirements of the client are not specified. This necessitates addition and modifications to the requirements later in the development cycle, which are often expensive to incorporate.

3.2.2 Project Schedule Study phase:

In the study phase we do the preliminary investigation and determine the system requirements. We study the system and collect the data to draw the dataflow diagrams. We follow the methods like questions and observation to find the facts that are involved in the process. This is an important because if the specification study is not done properly then design phase etc will go wrongly.

3.2.3 Design Phase:

In this design phase we design the system making use of study phase and the data flow diagrams. We make use the general access for designing.

We consider the top down approach. In the design phase we determine the entities and their attributes and the relationships between the entities. We do both logical and physical design of the system.

3.2.4 Development Phase:

In the development phase we mostly do the coding part following the design of the system. We follow modular programming for development and after development and after developing each and every module we do the unit testing followed by the integration testing.

3.2.5 Implementation Phase:

The last phase of the project is the implementation phase. Quality assurance is the primary motive in this phase. The quality assurance is the review of software products and related documentation for completeness, correctness, reliability and maintainability. The philosophy behind the testing is it finds errors. The testing strategies are of tow types, the code testing and the specifications testing. In the code testing we are examining the logic of the program. On the

surface, code testing seems to be ideal methods for testing software, but no tall software errors are uncovered.

3.3 Feasibility Study

Feasibility is an important phase in software development process. It enables the developers to have an assessment of the product being developed. It refers to the feasibility study of product in terms of outcomes of the product, operational use and technical support required for implementation it.

Feasibility study should be performed on the basis of various criteria and parameters. The various feasibility studies are:

1. Economic Feasibility
2. Operational Feasibility
3. Technical Feasibility

3.3.1 Economic Feasibility

It refers to the benefits or outcomes we are deriving from the product as compared to the total cost we are spending for developing the product.

In the present system, the development of new product greatly enhances the accuracy of the system and reduces the delay in the processing of applications and generating the reports. The errors can be greatly reduced and at the same time providing the great level of security. Here we don't need additional equipment except memory of required capacity. No need for spending money on client for maintenance because the database used is web enabled database.

3.3.2 Operational Feasibility

It refers to the feasibility of the product to be operational. Some products may work very well at design and implementation but may fail in the real time environment. It includes the study of human required and their technical expertise.

In the present system, the entering the details, updating the details and reports generations are perfect and quick in operations.

3.3.3 Technical Feasibility

It refers to whether the software that is available in the market fully supports the present application .It studies the pros and cons of using particular software for the development and its feasibility. It also studies the additional time needed to be given to people to make the application work.

In the present system the user interface is user friendly and does not require much expertise and training .It just needs mouse click to do operations and to generate reports. The software that is used for developing is highly suitable for the present applications since the users require fast access to the web pages with a high degree of security. This is achieved through integration of web server and database server in the same environment.

4. LITERATURE SURVEY

Data mining research currently faces two great challenges: how to embrace data mining services with just-in-time and autonomous properties and how to mine distributed and privacy-protected data. To address these problems, the authors adopt the Business Process Execution Language for Web Services in a **service oriented distributed data mining (DDM)** platform to choreograph DDM component services and fulfill global data mining requirements. They also use the learning-from- illustrate how localized autonomy on privacy-policy enforcement plusa bidding process can help the service-oriented system self-organize.

Most data mining algorithms assume that data analysts will aggregate data extracted from production systems at a server for subsequent computationally intensive data-crunching processes. However, issues such as data privacy concerns (with respect to customer information stored in bank servers, for example) and limits on data transmission bandwidth (affecting terabytes of scientific data generated from remote lab instruments or supercomputers) demonstrate that aggregating data for centralized mining simply isn't possible in a growing number of cases.

Instead, it's become necessary to develop methodologies for mining distributed data that must remain private.1 In addition, being able to get the right information at the right time (with respect to real-time business intelligence, for example) is an important business strategy in today's highly dynamic market.

This real-time objective imposes additional requirements on distributed data mining (DDM), including providing on-demand and self-adaptive services so that companies can cope with heterogeneities in data sources, with respect to data privacy requirements, which aren't always known in advance. We can address these challenges in two ways: a distributed computing architecture can support seamless provision, integration, and coordination of just-in-time and autonomous data mining services and a privacy-conscious DDM methodology can work on top of this architecture.

The purpose of this project is to solve the problems of data mining research which is currently facing two great challenges i.e. how to embrace data mining services with just-in-time and autonomous properties and how to mine distributed and privacy-protected data.

5. SYSTEM DESIGN

System design is the process, which involves conceiving planning and carrying out the plan by generating the necessary reports and inputs. In other words design phase acts as bridge between the software requirement specification and implementation phase, which satisfies those

requirements. System design is the transformation of the analysis model into a system design model.

The design of the system is correct if a system built precisely according to the requirements of that system. Design should be clearly verifiable, complete and traceable. The goal is to divide the problem into manageably small modules that can be solving separately. The different modules have to cooperate and communicate together to solve the problem. The complete project is broken down into different identifiable modules. Each module can be understood separately. All the modules at last are combined to get the solution of the complete system.

5.1 Model View Controller Architecture

MVC Architecture defines that the controller is separated from the system model and View. It is composed of three different layers. Being a Web based architecture. The user Interface is completely separated from the entire Business Logic Layer.

The Database Layer and Business Logic Layer runs on the server Machine and the User Interface Layer will run under the Client Machine. For developing the User Interface we are having HTML and Java Script. For Business Login and Database Connectivity Servlets and JSP are used. In the Backed the servlets and Jsp's are connected to database through JDBC API. The web server plays a major role in connecting the client user interface and the servlets and JSP

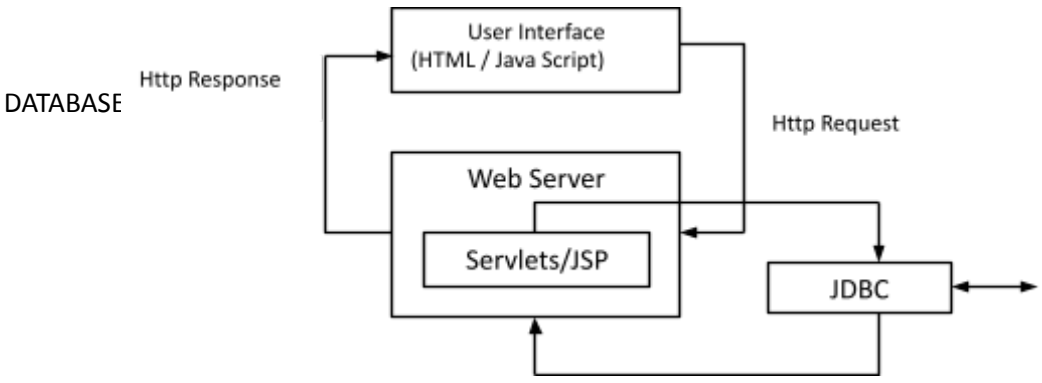


Figure 3: Block Diagram of Architecture

5.2 Unified Modeling Language Diagrams

- The unified modeling language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic semantic and pragmatic rules.
- A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagram, which is as follows.

Structural model view

- In this model the data and functionality are arrived from inside the system.
- This model view models the static structures.

Behavioral Model View

- It represents the dynamic of behavioral as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.

Implementation Model View

- In the structural and behavioral as parts of the system are represented as they are to be built.

Environmental Model View

- In this structural and behavioral aspects of the environment in which the system is to be implemented are represented.

UML is specifically constructed through two different domains they are

- UML Analysis modeling, this focuses on the user model and structural model views of the system.
- UML design modeling, which focuses on the behavioral modeling, implementation modeling and environmental model views.

5.3 Diagrams

5.3.1 Use case Diagrams

Use case Diagrams represent the functionality of the system from a user's point of view. Use cases are used during requirements elicitation and analysis to represent the functionality of the system. Use cases focus on the behavior of the system from external point of view.

Actors are external entities that interact with the system. Examples of actors include users like administrator, bank customer …etc., or another system like central database

5.3.2 Class Diagrams

Class diagrams are widely used to describe the types of objects in a system and their relationships. These model the class structure and contents using elements such as classes, packages and objects. Class diagrams describe three different perspectives when designing a system. Conceptual, specification and implementation.

5.3.3 Interaction Diagrams

Sequence diagrams and Collaboration diagrams both are called as interaction diagrams. These are two of the five diagrams used in the UML for modeling the dynamic aspects of the systems. An interaction diagrams shows an interaction, consisting of a set of objects and their relationships, including the messages that may be dispatched among them.

5.3.4 Sequence Diagrams

A sequence diagram shows, as parallel vertical lines ("life lines"), different processes or objects that live simultaneously and horizontal arrows, the messages exchanged

between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

5.3.5 Collaboration Diagrams

A collaboration diagram emphasizes the organization of the objects that participate in an interaction. There are two distinguish features from sequence diagram to represent, first, there is the path to indicate how one object is linked to other and second is sequence number, to indicate the time order of messages by prefix with number.

To illustrate the concept of autonomous DDM, we performed an experiment using GMM as the global model to demonstrate how to reach the optimal trade-off between the overall data mining quality and the local source's data granularity levels via self-organization. Instead of assuming that the privacy-control component is passive, we implemented the local data sources with the autonomous property to negotiate with the global broker service regarding which data abstraction level to present.

The global broker first requests a data abstraction with coarse granularity from each local source. Then, it actively requests more specific details from those sources on a need-to-know basis so that it can learn the global model in a cost-effective manner.

The global brokering service can send the local sources the global model learned up to a specific moment, for example, and the local sources can then return their bid values computed based on the local data likelihood (defined as the product of the probabilities of generating the data) gained per unit cost by advancing one more level of granularity at the local sources. The global service will ask for more data details from the source with the highest value returned. This protocol continues until the data likelihood stops improving significantly or the computational budget runs out.

Use case Diagram for Admin

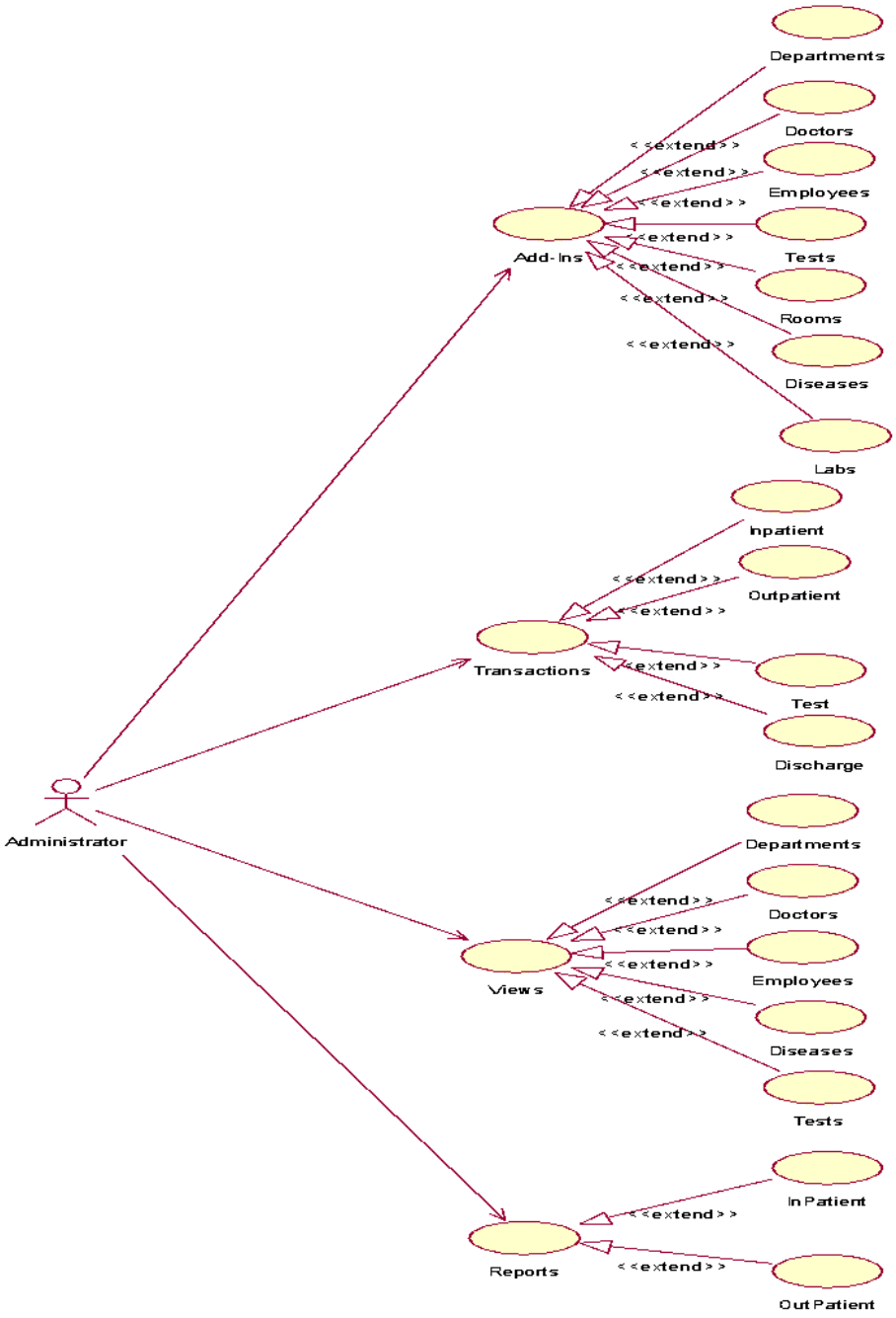


Figure 4: Use case for admin

Use case Diagram for Patient

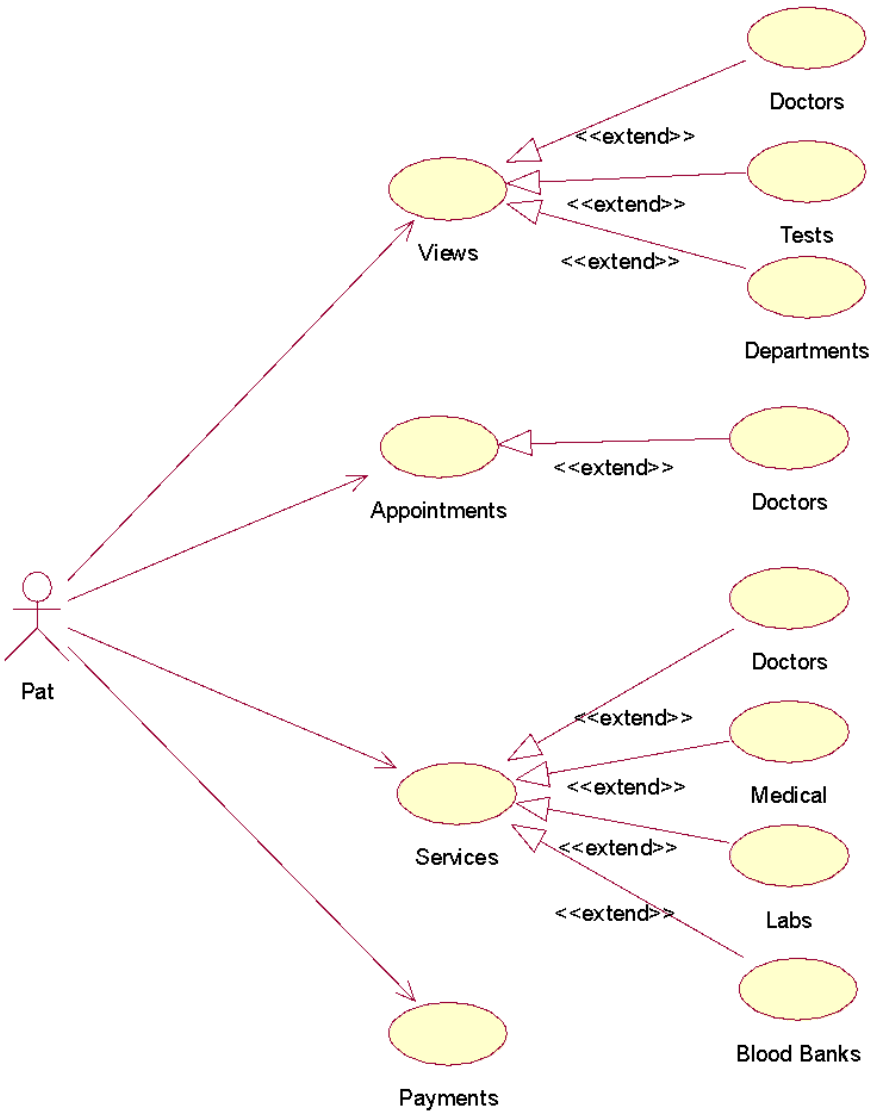


Figure 5: Use case for Patient

Use case Diagram for Patient

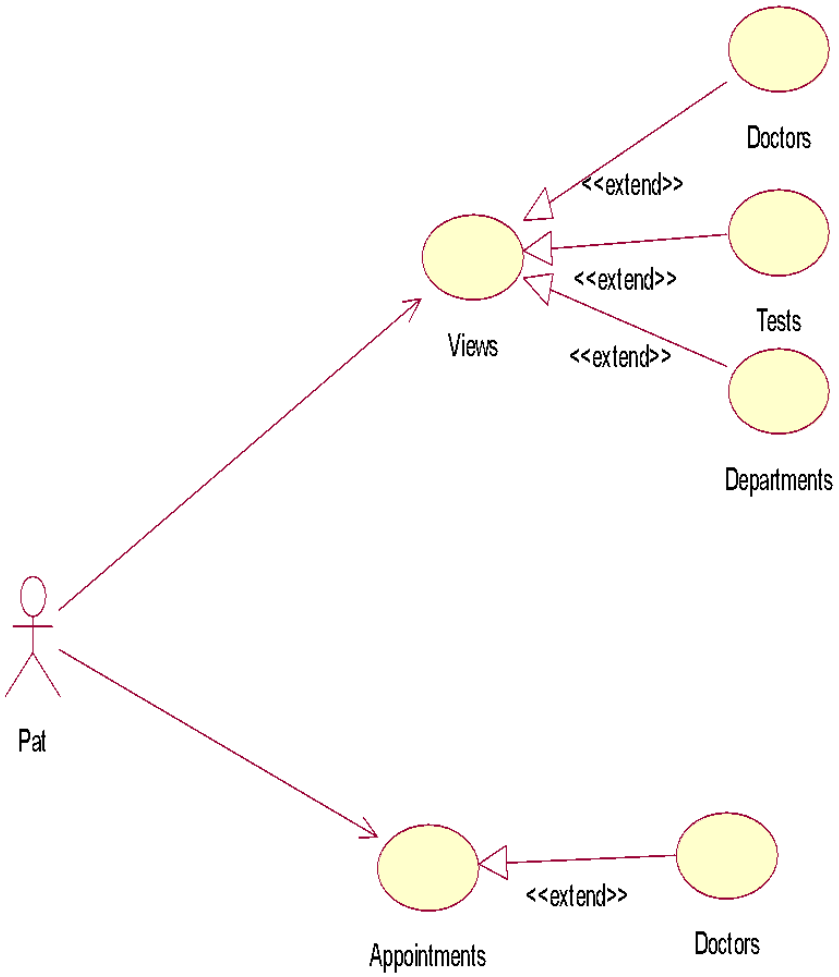


Figure 6: Use case for Public

Class Diagram for Medicare

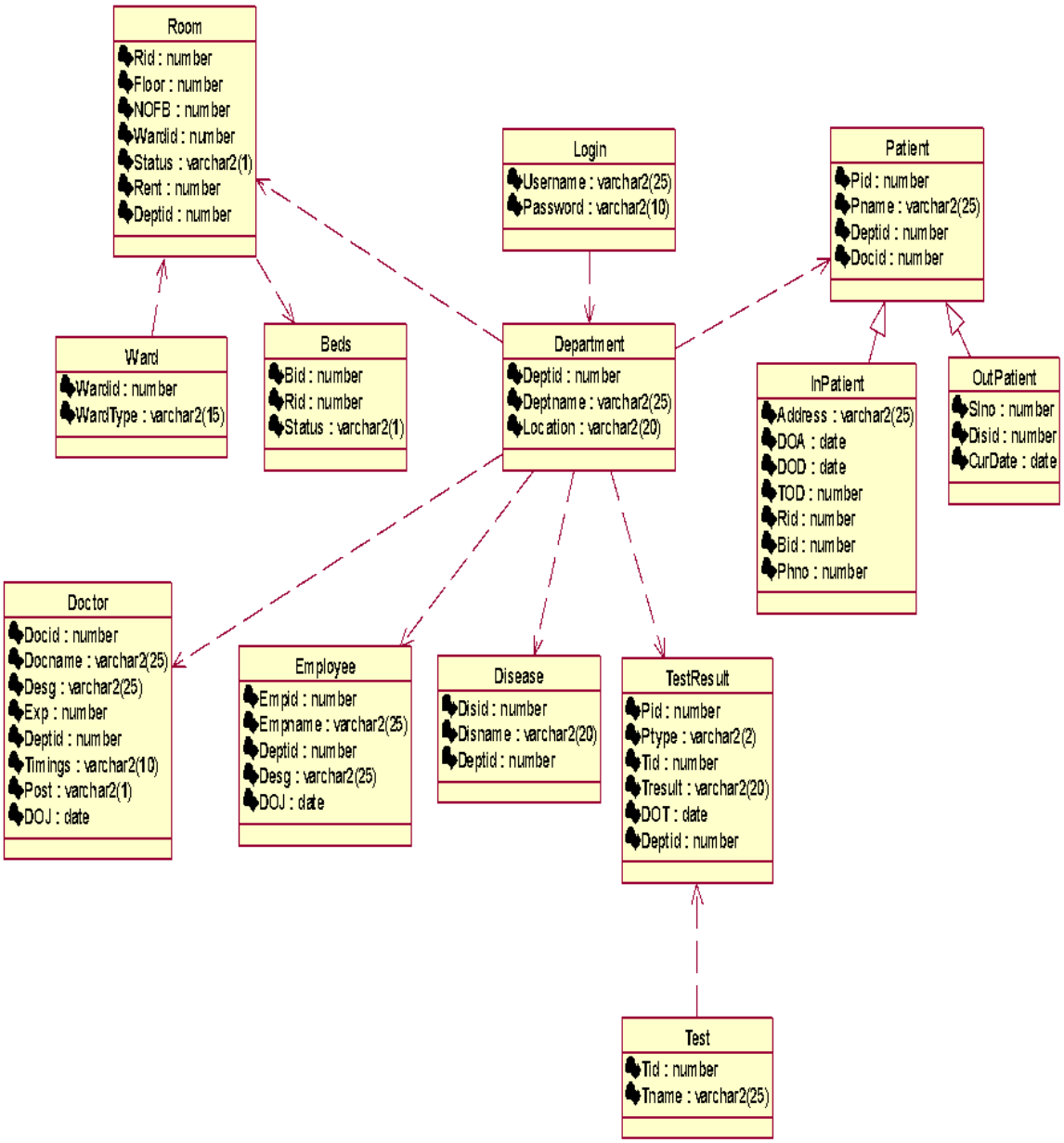


Figure 7: Class Diagram for Medicare

Sequence Diagram for Admin

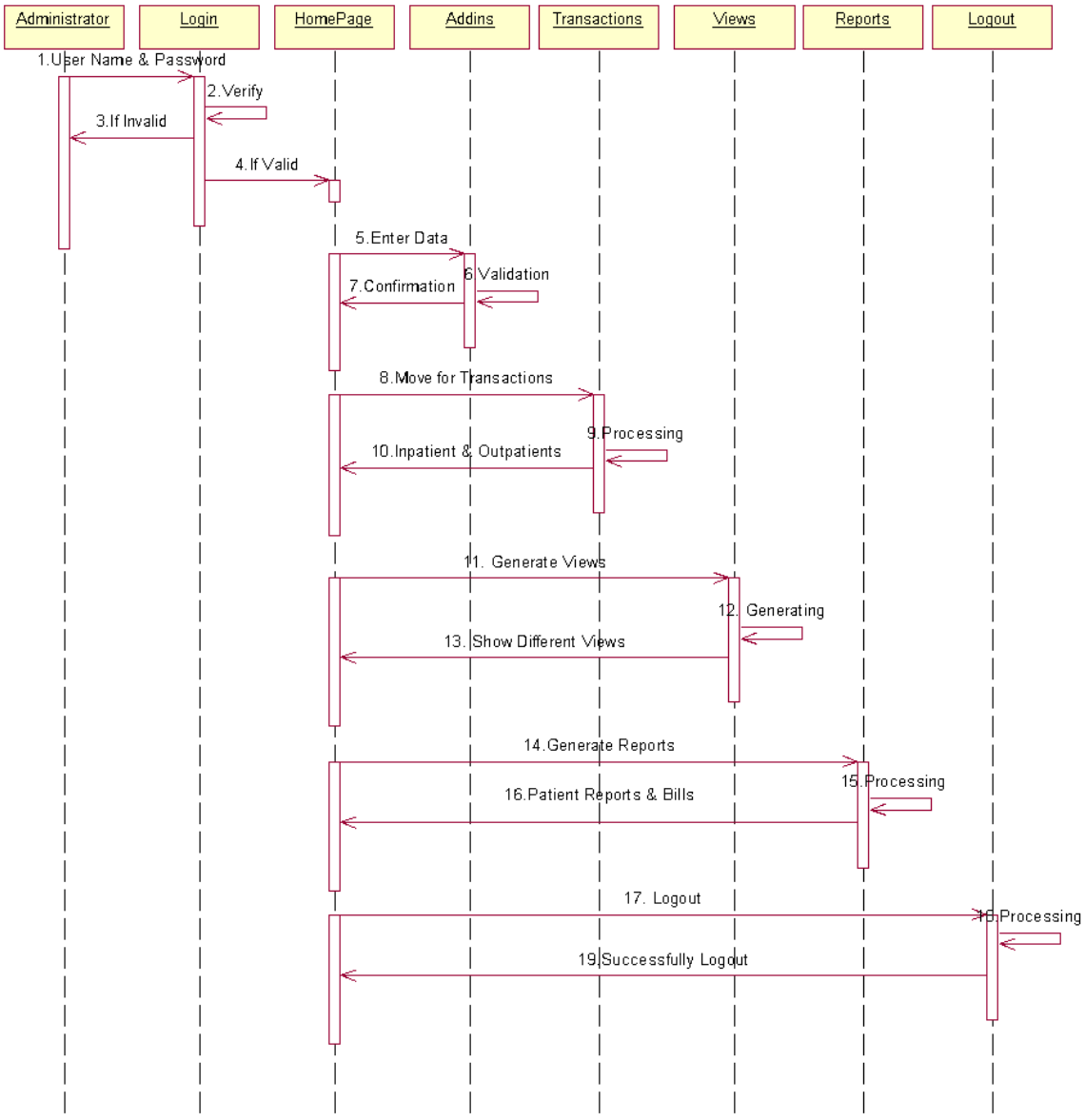


Figure: 8 Sequence Diagram for Admin

Sequence Diagram for patient

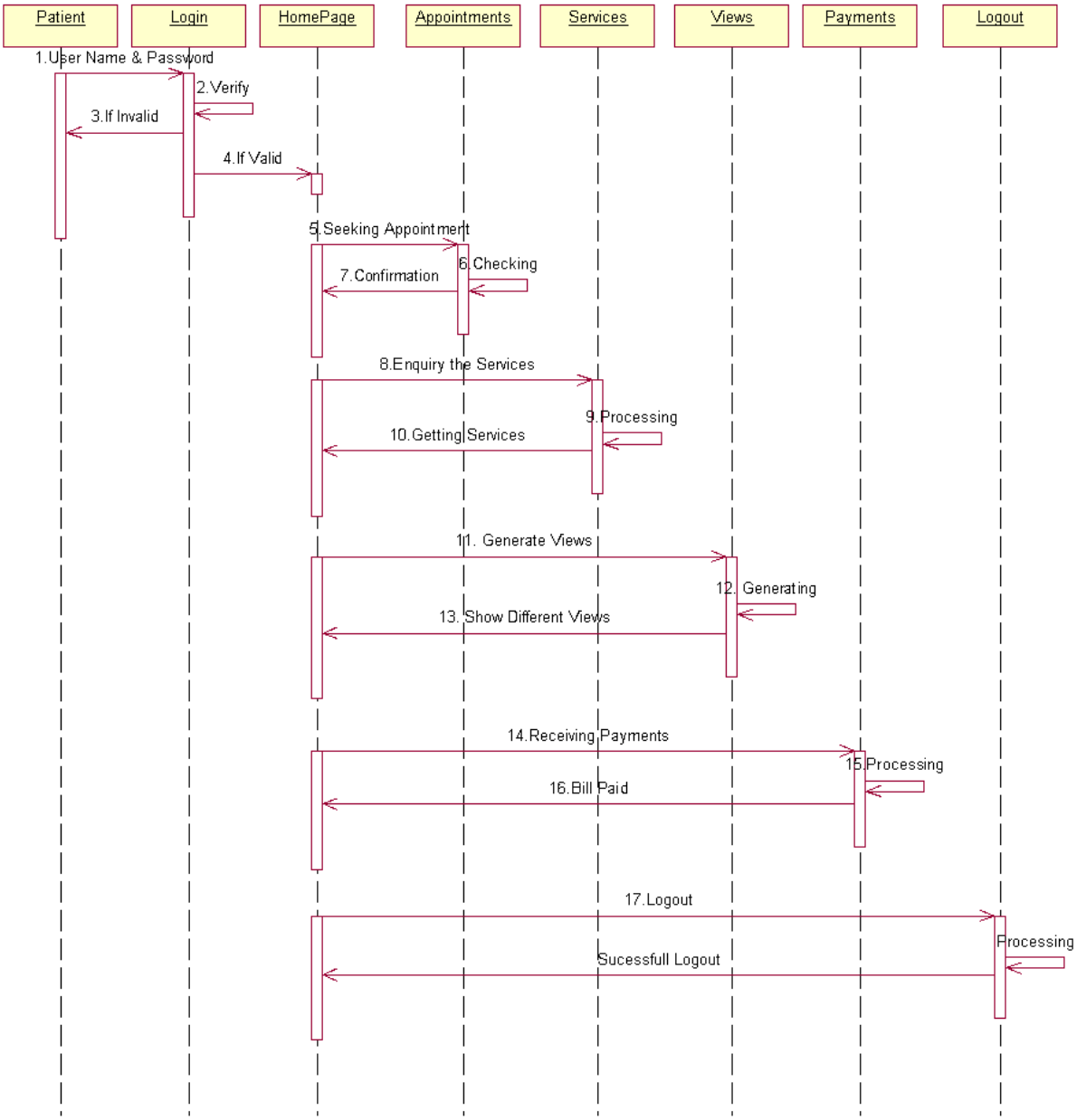


Figure: 9 Sequence Diagram for patient

Sequence Diagram for public

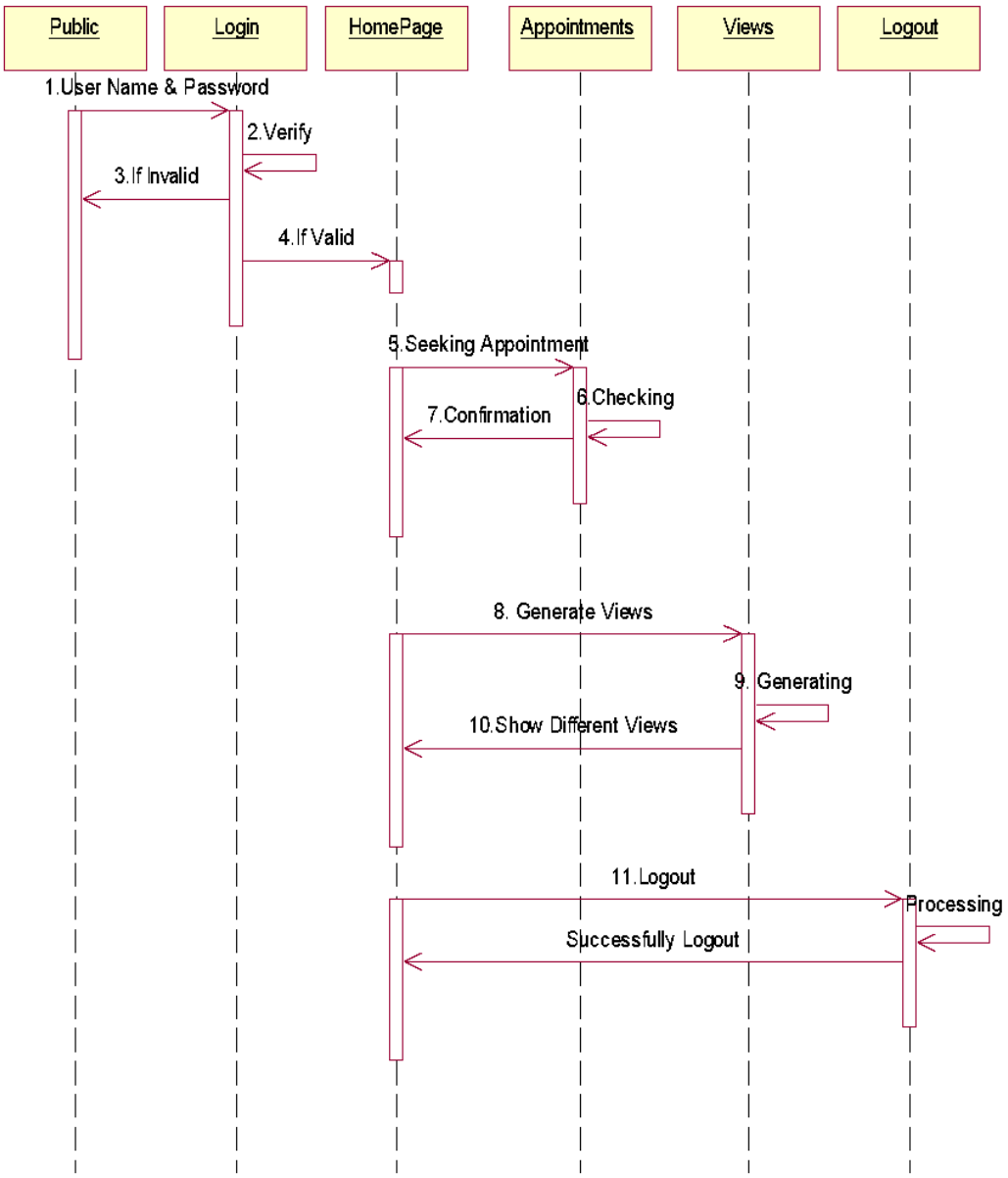


Figure: 10 Sequence Diagram for public

Collaboration Diagram for Admin

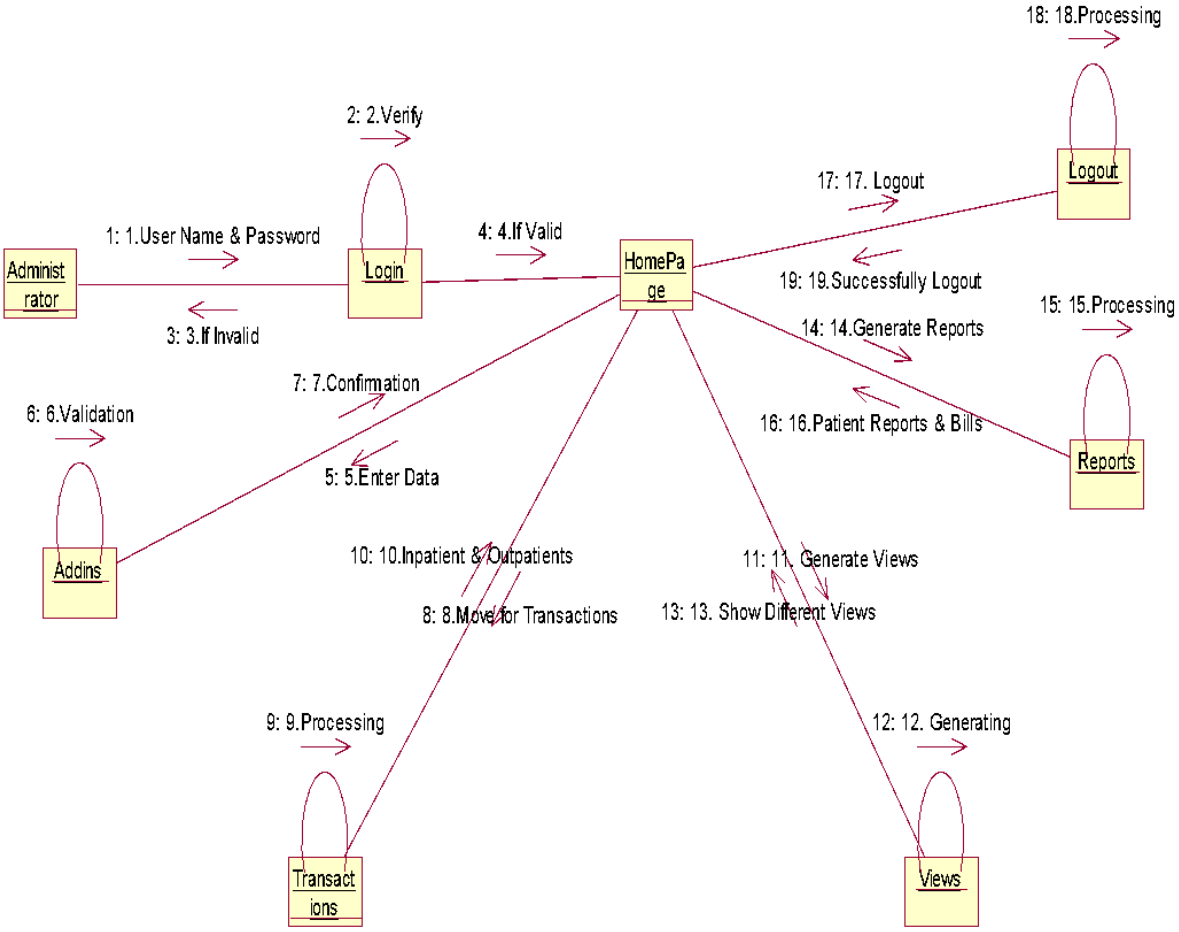


Figure: 11 Collaboration Diagram for Admin

Collaboration Diagram for Patient

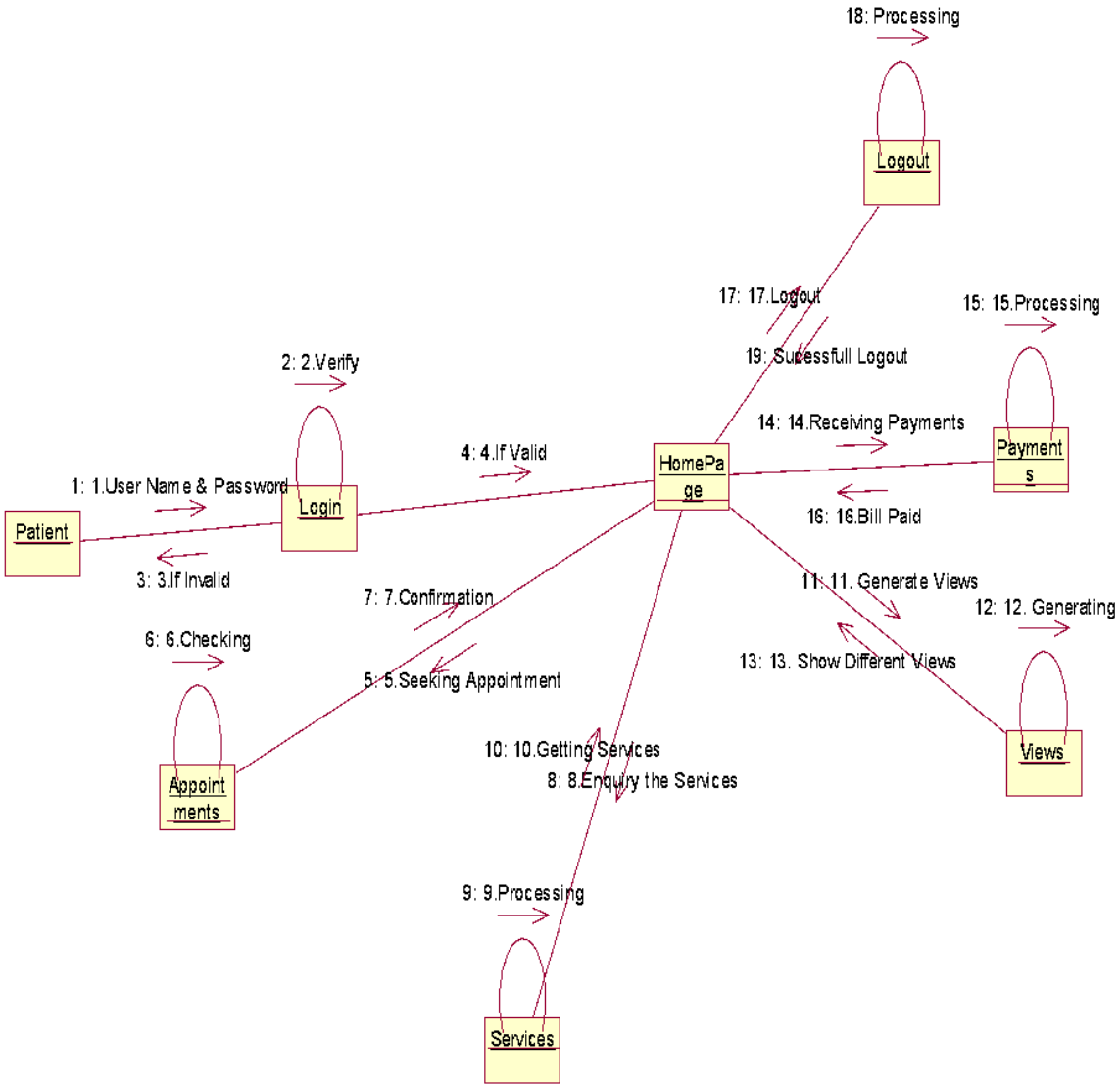


Figure: 12 Collaboration Diagram for Patient

Collaboration Diagram for public

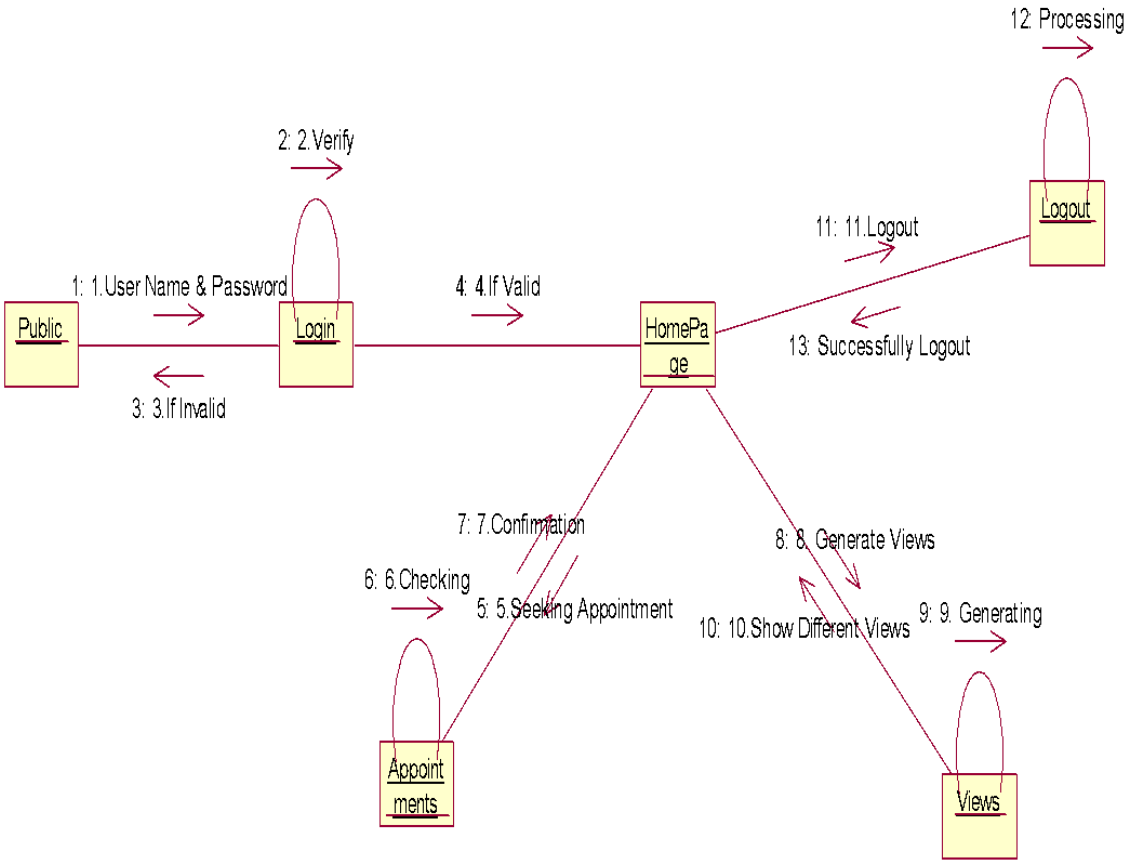


Figure: 13 Collaboration Diagram for public

State chart Diagram for Admin

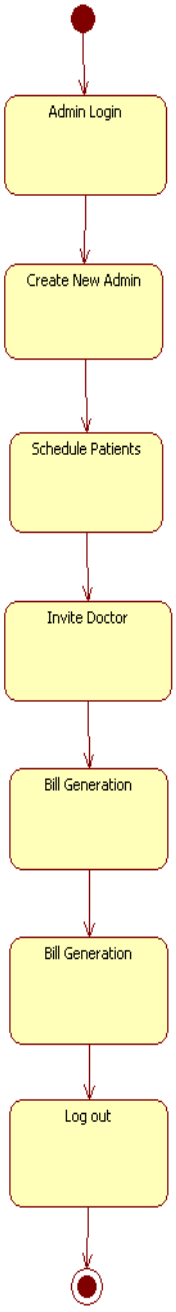


Figure 14: State chart Diagram for Admin

StateChart Diagram for Patient

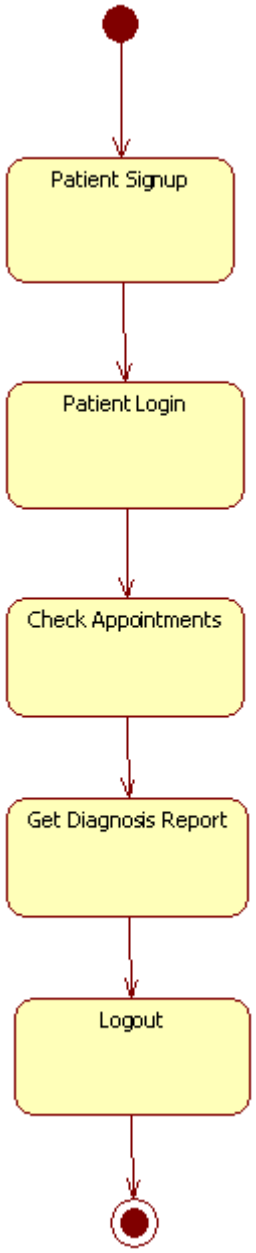


Figure 15 StateChart Diagram for Patient

State ChartDiagram for Doctor

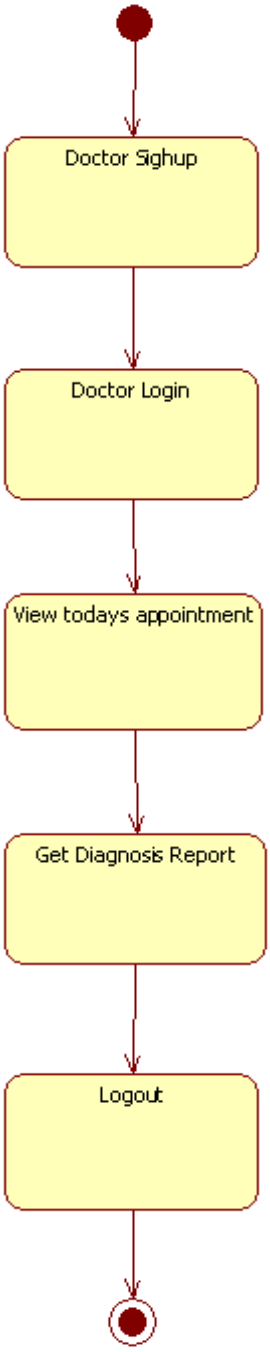


Figure 16 State ChartDiagram for Doctor

6. TECHNOLOGIES USED

6.1 JAVA

In my project, I have chosen **Java** language for developing the code.

About Java

Initially the language was called as **"Oak"** but it was renamed as **"Java"** in 1995. The primary motivation of this language was the need for a platform-independent (i.e., architecture neutral) language that could be used to create software to be embedded in various consumer electronic devices.

- Java is a programmer's language.
- Except for those constraints imposed by the internet environment, Java gives the programmer, full control.
- Finally, java is to internet programming where C was to system programming.

Importance of Java to the Internet

Java has had a profound effect on the Internet. This is because, Java expands the Universe of objects are transmitted between the Server and the Personal Computer. They are: Passive information and Dynamic active programs. The Dynamic, Self-executing programs cause serious problems in the areas of Security and probability. But, Java addresses those concerns and by doing so, has opened the door to an exciting new form of program called Applet.

Java can be used create two types of programs

Applications and Applets: An application is a program that runs on our Computer under the operating system of that computer. It is more or less like one creating using C or C++. Java's ability to create Applets makes it important. An Applet is an application designed to be transmitted over the Internet and executed by a Java-compatible web browser.

And applet is actually a tiny Java program, dynamically downloaded across the network, just like an image. But the difference is, it is intelligent program, not just a media file. It can react to the user input and dynamically change.

Features of java

- **Security**

Every time you that you download a "normal" program; you are risking a viral infection. Prior to Java, most users did not download executable programs frequently, and those who did scan them for viruses prior to execution. Most users still worried about malicious program exists that must be guarded against. This type of program can gather private information. Such as credit card numbers, bank account balances, and passwords. Java answers the both of these concerns by providing a "firewall" between a networked application and your computer.

- **Portability**

For programs to be dynamically downloaded to all the various types of platforms connected to the Internet, some means of generating portable executable code is needed as you will see the same mechanism that helps ensure security also helps create portability. Indeed, Java's solution to these two problems is both elegant and efficient.

- **The Byte code**

The key that allows the Java to solve the security and portability problems is that the output of Java compiler is Byte code. Byte code is a highly optimized set of instructions designed to be executed by the Java run-time system, which is called the Java Virtual Machine (JVM). That is, in its standard form, the JVM is an interpreter for byte code. Translating a java program into byte code helps makes it much easier to run a program in a wide variety of environments. The reason is, once the run-time package exists for a given system, any Java program can run on it.

Although Java was designed for interpretation, there is technically nothing about Java that prevents on-the-fly compilation of byte code into native code. Sun has just completed its Just In

Time (JIT) compiler for byte code. When the JIT compiler is a part of JVM, it compiles byte code into executable code in real time, on a piece –by-piece, all at once, because Java performs various run-time checks that can be done only at run Java Virtual Machine (JVM).

Beyond the language, there is the Java virtual machine. The java virtual machine is an important element of the java technology. The virtual machine can be embedded within a web browser or an operating system. Once a piece of Java code is loaded onto a machine, it is verified. As part of the loading process, a class loader is invoked and does byte code verification makes sure that the code that's has been generated by the compiler will not corrupt the machine that it's loaded on. Byte code verification takes place at the end of the compilation process to make sure that is all accurate and correct. So byte code verification is integral to the compiling and executing of Java code

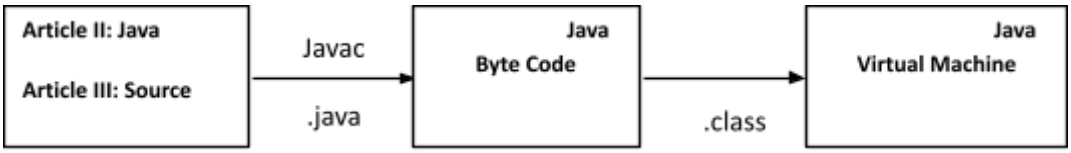


Figure: 15 Development process of java programming

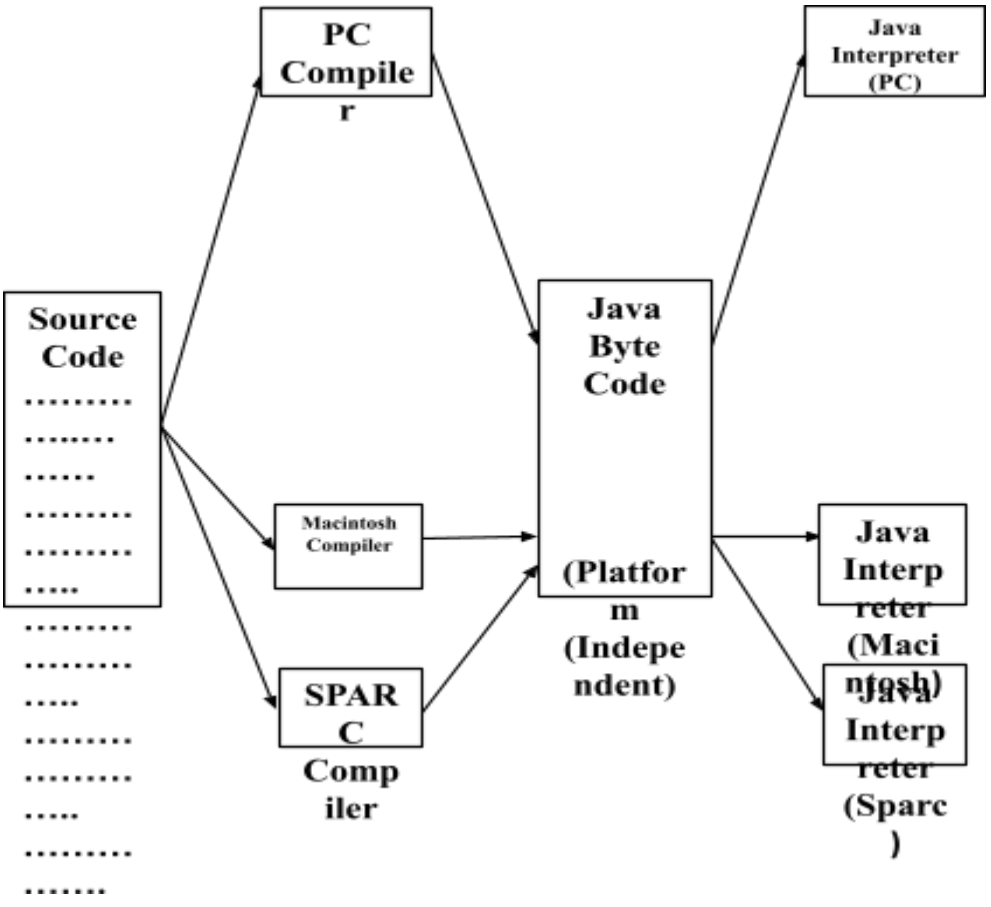
- ## Java Architecture

Java architecture provides a portable, robust, high performing environment for development. Java provides portability by compiling the byte codes for the Java Virtual Machine, which is then interpreted on each platform by the run-time environment. Java is a dynamic system, able to load code when needed from a machine in the same room or across the planet.

- **Compilation of code:**

When you compile the code, the Java compiler creates machine code (called byte code) for a hypothetical machine called Java Virtual Machine (JVM). The JVM is supposed to execute the byte code. The JVM is created for overcoming the issue of portability. The code is written and compiled for on machine and interpreted on all machines. This machine is called Java Virtual Machine.

- **Compiling and interpreting Java Source Code**



During run-time the Java interpreter tricks the byte code file into thinking that it is running on a Java Virtual Machine. In reality this could be a Inte3l Pentium Windows 95 or sun SARC station running Solaris or Apple Macintosh running system and all could receive code from any computer through Internet and run the Applets.

- **Simple**

Java was designed to be easy for the Professional programmer to learn and to se effectively. If you are an experienced C++ programmer, learning Java will be even easier. Because Java inherits the C/C++ syntax and many of the object oriented features of C++. Most of the confusing concepts from C++ are either left out of Java or implemented in a cleaner, more approachable manner. In java there are a small number of clearly defined ways to accomplish a given task.

- **Object-Oriented**

Java was not designed to be source code compatible with any other language. This allowed the Java team the freedom to design with a blank slate. One outcome of this was a clean usable, pragmatic approach to objects. The object model in java is simple and easy to extend, while simple types, such as integers, are kept as high-performance non-objects.

- **Robust**

The multi-platform environment of the Web places extraordinary demands on a program, because the program must execute reliably in a variety of systems. The ability to create robust programs was given a high priority in the design of Java. Java is strictly typed language; it checks your code at compile time and runtime.

Java virtually eliminates the problems of memory management and deallocation, which is completely automatic. In a well-written Java program, all run time errors –can and should –be managed by your program.

6.2 SERVLETS

Introduction

The Java Web server is javasofts own web Server. The java web server is just a part of a larger framework, intended to provide you not just with a web server, but also with tools. To build customized network servers of ay Internet or Intranet client/server system. Servlets are to a web server, how applets are to browser.

About servlets

Servlets provide a Java –based solution used to address the problems currently associated with doing server-side programming, including inextensible scripting solutions, platform-specific APIs, and incomplete interfaces.

Servlets are objects that conform to a specific interface that can be plugged into a Java-based server. Servlets are to the server-side what applets are to the client-side-object byte codes that can be dynamically loaded off the net. They differ from applet that they are faceless objects (without graphics or a GUI component). They serve as platform independent, dynamically loadable, pluggable helper byte code objects on the server side that can be used to dynamically extend server-side functionality.

For example, an HTTP Servlets can be used to generate dynamic HTML content. When you use servlets to do dynamic content you get the following advantages:

- They’re faster and cleaner than CGI scripts.
- They use a standard API (the Servlets API).
- They provide all the advantages of Java (run on a variety of servers without needing to be rewritten).

6.3 JAVA SCRIPT

JavaScript is a script-based programming language which was developed by Netscape Communication Corporation. JavaScript was originally called Live Script and renamed as JavaScript to indicate its relationship with Java.

JavaScript supports the development of both client and server components of Web-based applications. On the client side, it can be used to write programs that are executed by a Web browser within the context of a Web page. On the server side, it can be used to write web server programs that can process information submitted by a Web browser and then updates the browser's accordingly.

Even though **JavaScript** supports both client and server Web programming, we prefer JavaScript at Client side programming since most of the browsers supports it.

JavaScript is almost as easy to learn as HTML, and JavaScript statements can be included in HTML documents by enclosing the statements between a pair of scripting tags <SCRIPT>...</SCRIPT>.

<SCRIPT LANGUAGE="JavaScript">

JavaScript statements

</SCRIPT>

Here are a few things we can do with JavaScript:

- Validate the contents of a form and make calculations.
- Add scrolling or changing messages to the Browser's status line.
- Animate images or rotate images that change when we move the mouse over them.
- Detect the browser in use and display different content for different browsers.
- Detect installed plug-ins and notify the user if a plug-in is required.

We can do much more with JavaScript, including creating entire application.

JavaScript vs. Java

JavaScript and Java are entirely different languages. A few of the most glaring differences are:

- Java applets are generally displayed in a box within the web document; JavaScript can affect any part of the Web document itself.
- While JavaScript is best suited to simple applications and adding interactive features to Web pages. Java can be used for incredibly complex applications. There are many other differences but the important thing to remember is that JavaScript and Java are separate languages. They are both useful for different things; in fact they can be used together to combine their advantages.

Advantages

- JavaScript can be used for Server-side and Client-side scripting. It is more flexible than VBScript.
- JavaScript is the default scripting languages at Client-side since all the browses supports it.

5.4 JAVA SERVER PAGES (JSP)

A JSP page is a text-based document that describes how to process a request to create a response. The description intermixes template data with some dynamic actions and leverages on the Java Platform.

The Java Server Pages specification includes:

- Standard directives
- Standard actions
- Script language declarations, script lets and expressions

- A portable tag extension mechanism.

Directives and Actions

There may be two types of elements in a JSP page: **directives** or **actions.** Directives provide global information that is conceptually valid independent of any specific request received by the JSP page. For example, a directive can be used to indicate the scripting language to use in a JSP page. Actions may, and often will, depend on the details of the specific request received by the JSP page. If a JSP container uses a compiler or translator, the directives can be seen as providing information for the compilation/translation phase, while actions are information for the subsequent request processing phase.

An action may create some objects and may make them available to the scripting elements through some scripting-specific variables.

Directive elements: Syntax:

<%@ directive ... %>

Action elements: tag:<mytag attr1=”attribute value” ...>

body

</mytag>

or an empty tag

<mytab attr1=”attribute value” .../>

An element type abstracts some functionality by defining a specialized (sub)language that allows more natural expression of the tasks desired, can be read and written more easily by tools and also can even contribute specialized yet portable tool support to create them. The JSP specification provides a Tag Extension mechanism that enables the addition of new actions, thus allowing the JSP page “language” to be easily extended in a portable fashion. A typical example would be elements to support embedded database queries. Tag libraries can be used by JSP page

authoring tools and can be distributed along with JSP pages to any JSP container like Web and Application servers.

The Tag Extension mechanism can be used from JSP pages written using any valid scripting language, although the mechanism itself only assumes a Java run time environment. Custom actions provide access to the attribute values and to their body; they can be nested and their bodies can include scripting elements.

Execution

A JSP page is executed in a JSP container, which is installed on a Web server, or on a Web enabled application server. The JSP container delivers requests from a client to a JSP page and responses from the JSP page to the client. All JSP containers must support HTTP as a protocol for requests and responses, but a container may also support additional request/response protocols. The default request and response objects are of type HttpServletRequest and HttpServletResponse, respectively.

Compilation

JSP pages may be compiled into its JSP page implementation class plus some deployment information. This enables the use of JSP page authoring tools and JSP tag libraries to author a Servlet. This have several benefits:

Removal of the start-up lag that occurs when a JSP page delivered as source receives the first request & Reduction of the footprint needed to run a JSP container, as the java compiler is not needed.

Objects and Scopes

JSP page can create and/or access some Java objects when processing a request. The JSP specification indicates that some objects are created implicitly, perhaps as a result of a directive other objects are created explicitly through actions; objects can also be created directly using scripting code, although this is less common. The created objects have a scope attribute defining where there is a reference to the object and when that reference is removed.

6.5 HTML

Hypertext Markup Language (HTML), the languages of the World Wide Web (WWW), allows users to produces Web pages that include text, graphics and pointer to other Web pages (Hyperlinks).

HTML is not a programming language but it is an application of ISO Standard 8879, **SGML (Standard Generalized Markup Language),** but specialized to hypertext and adapted to the Web. The idea behind Hypertext is that instead of reading text in rigid linear structure, we can easily jump from one point to another point. We can navigate through the information based on our interest and preferences.

A markup language is simply a series of elements, each delimited with special characters that define how text or other items enclosed within the elements should be displayed. Hyperlinks are underlined or emphasized works that load to other documents or some portions of the same documents.

HTML can be used to display any type of document on the host computer, which can be geographically at a different location. It is a versatile language and can be used on any platform or desktop.

HTML provides tags (special codes) to make the document look attractive. HTML tags are not case-sensitive. Using graphics, fonts, different sizes, color, etc., can enhance the presentation of the document. Anything that is not tag is part of the document itself.

Basic HTML Tags

<! --- -->	specifies comments
<A>……</A>	Creates hypertext links
<B>……. </B>	Formats text as bold
<BIG>……</BIG>	Formats text in large font
<BODY>……. </BODY>	Contains all tags and text in the HTML document
<CENTER>…. </CENTER>	Creates text

<DD>……..</DD>	Definitions of a term
<DL>……</DL>	Creates definition list
<FONT>……. </FONT>	Formats text with a particular font
<FORM>…..</FORM>	Encloses a fill-out form
<FRAME>…. </FRAME>	Defines a particular frame in a set of frames
<H#>……..</H#>	Creates headings of different levels
<HEAD>……..</HEAD>	Contains tags that specify information about a document
<HR>……..</HR>	Creates a horizontal rule
<HTML>……</HTML>	Contains all other HTML tags
<META>……</META>	Provides meta-information about a document
<SCRIPT>……</SCRIPT>	Contains client-side or server-side script
<TABLE>……</TABLE>	Creates a table
<TD>……</TD>	Indicates table data in a table
<TR>……</TR>	Designates a table row
<TH>……</TH>	Creates a heading in a table.

Advantages

- A HTML document is small and hence easy to send over the net. It is small because it does not include formatted information.
- HTML is platform independent and HTML tags are not case-sensitive.

5.6 Java Database Connectivity

What is JDBC

JDBC is a Java API for executing SQL statements. (As a point of interest, JDBC is a trademarked name and is not an acronym; nevertheless, JDBC is often thought of as standing for Java Database Connectivity. It consists of a set of classes and interfaces written in the Java programming language. JDBC provides a standard API for tool/database developers and makes it possible to write database applications using a pure Java API.

Using JDBC, it is easy to send SQL statement to virtually any relational database. One can write a single program using the JDBC API, and the program will be able to send SQL statements to appropriate database. The combinations of java and JDBC lets a programmer write it once and run it anywhere.

What Does JDBC Do

Simply put, JDBC makes it possible to do three things:

- Establish a Connection with a database
- Send SQL statements
- Process the results

JDBC versus ODBC and other APIs

At this point, Microsoft‘s ODBC (open database Connectivity) API is that probably the most widely used programming interface for accessing relational databases. It offers the ability to connect to almost all databases on almost on almost all platforms.

So why not just use ODBC form java? The answer is that you can use ODBC from java, but this is best done with the help of JDBC in the form of the JDBC-ODBC.

Bridge, which we will cover shortly. The question now becomes “why do you need JDBC? “ There are several answers to this question:

- ODBC is not appropriate for direct use form java because it uses a c Interface. Calls from java to native c code have a number of drawbacks in the security, implementation, robustness, and automatic portability of applications.
- A literal translation of the ODBC C API would not be desirable. For example, java has no pointers, and ODBC makes copious use of them including the notoriously error-prone generic pointer "void *". You can think of JDBC as ODBC translated into an object-oriented interface that is natural for java programmers.
- ODBC is hard to learn. It mixes simple and advanced features together, and it has complex options even for simple queries. JDBC, on the others hand, was designed to keep simple things simple while allowing more advanced capabilities where required.
- A java API like JDBC is needed in order to enable a "pure java" solution. When ODBC is used, the ODBC drivers manage and drivers must be manually installed on every client machine. When the JDBC driver is written completely in java, however, JDBC code is automatically installable, portable and secure on all java platforms form network computers to mainframes.

6.7 ORACLE

Oracle is a relational database. The language used to access relation database is Structured Query language, SQL is flexible, efficient language, with features designed to manipulate and examine relational data.

SQL is fourth generation language. This means that the language describes what should be done, but not how to do it. Fourth generation languages are fairly simple and have fewer commands, 4 GL's also insulate the user from underlying data structures and algorithms.

Why to choose oracle

Oracle is popular relational database. Software is available on a wide number of platforms. Oracle provides complete control organizing the data storage to obtain good performance using indexing, clustering.

Introduction to SQL

STRUCTURED QUERY LANGUAGE is a tool of communication b/w a user and RDBMS. SQL is a simple and powerful language in the sense that most of the operations in RDBMS can be done using SQL. It is a 4'GL. The important feature is that it is not procedural, advanced of using non-procedural languages are:

- Reduction of code
- Simplicity in writing code.
- Ease of maintenance.

SQL is made of 3 sub languages:

DDL: Data Definition Language consists of commands to create the objects, such as tables, views, indexes etc.

DML: Data Manipulation Language is used to querying, inserting deleting and updating of information stored in the database.

DCL: Data Control Language is used to control data and access to the database in the multi-user environment for security purpose. The DCL is used for giving access to different users.

ORACLE Utilities

Oracle utilities enhance RDBMS support of data entry, maintenance and retrieval. They are Export, import and SQL Loader.

EXPORT

Export is the only utility provided for oracle. Export writes data from an oracle database into oracle binary format files. The export makes a copy of data structure in an operating system file.

To export

- A user must have DBA privileges to Oracle database.
- A user must own tables to be exported.
- Before running there must be enough storage space on disk of tape to write exported files. If there is not enough space, export will terminate with a write failure error.
- The information that is to be exported depends upon the mode chosen.
- Export modes are tables, user, and full database.

- Export can be of three types incremental, cumulative and complete.

IMPORT

Import reads files created by export and places data in the database. To import user should have connection or privileges to an oracle database and access to an export file.

Only a DBA can import a file exported by him/her. A table is imported in to its original table space using the original storage parameters. If the storage space does not exist then the system uses default space of the current user. If an error occurs the import prints an error message and skips to the next table and continues the processing.

SQL * Loader

It is a tool for loading data from external files into oracle database. There are two mechanisms used to load data. Two types of inputs are to be provided to
SQL * Loader.

7. DATA DICTIONARY

7.1 Data base design

Introduction

A database model is a collection of logical constructs used to represent the data in data structures and data relationships within the database. Basically, the database models may be grouped into two categories: conceptual model and implementation models. The conceptual model focuses on the logical nature of that data presentation. Therefore the conceptual model is concerned with what is representing in the database and the implementation model is concerned with how it is represented.

- **Conceptual Model**

The conceptual model represents data present in the entities as well the relations present is the entities. All the strong entities and weak entities are identified here and it mainly focuses on the logical nature of that data presentation.

- **General Access Model**

General access model is used to convert the ER model into the relation model. Here we can identify the references to the other entities and the nature of the each attribute. But, A network model's record can have more than one parent.

- **Relation Model**

The relational model is represented as tables. The columns of each table are attributes that define the data or value domain for entities in that column. The rows of each table are tuples representing individual data objects being stored. A relational table should have only one primary key. A Primary key is a combination of one or more attributes whose value unambiguously locates each row in the table. Database normalization is a design technique by which relational database tables are structured in such a way as to make them invulnerable to certain types of logical inconsistencies and anomalies.

Tables can be normalized to varying degrees: relational database theory defines "normal forms" of successively higher degrees of stringency, so, for example, a table in third normal form is less open to logical inconsistencies and anomalies than a tale that is only in second normal form. Although the normal forms are after defined (informally) in terms of the characteristics of tables, rigorous definitions of the normal forms are concerned with the characteristics of mathematical constructs known as relations. Whenever information is represented relationally—that is, roughly speaking, as values within rows beneath fixed column headings—it makes sense to ask to what extent the representation is normalized.

7.2 Normal forms

The normal forms (abbrev. NF) of relational database theory provide criteria for determining a table's degree of vulnerability to logical inconsistencies and anomalies. The higher the normal form applicable to a table, the vulnerable it is to such inconsistencies and anomalies. Each table has a "highest normal form" (HNF): by definition, a table always meets the requirements of its HNF; also by definition, a table fails to meet the requirements of any normal form higher than its HNF.

First Normal Form

The criteria for first normal form (1NF) are:

- A table must be guaranteed not to have any duplicate records; therefore it must have at least one candidate key.
- There must be no repeating groups, i.e. no attributes which occur a different number of times on different records. For example, suppose that an employee can have multiple skills: a possible representation of employees' skills is {Employee ID, Skill1, Skill2, Skill3…….}, where {Employee ID} is the unique identifier for a record. This representation would not be in 1NF.
- Note that all relations are in 1NF. The question of whether a given representation is in 1NF is equivalent to the question of whether it is a relation.

Second Normal Form

The criteria for second normal form (2NF) are:

- The table must be in 1NF.
- None of the non-prime attributes of the table are functionally dependent on a part (proper subset) of a candidate key; in other words, all functional dependencies of non-prime attributes on candidate keys are full functional dependencies. For example, consider a "Department Members" table whose attributes are Department ID, Employee ID, and Employee Date of Birth; and suppose that an employee works in one or more departments. The combination of Department ID and Employee ID and Employee ID

uniquely identifies records within the table. Given that Employee Date of Birth depends on only one of those attributes – namely, Employee ID – the table is not in 2NF.

Third Normal Form

The criteria for third normal form (3NF) are:

- The table must be in 2NF.
- There are no non-trivial functional dependencies between non-prime attribute is only indirectly dependent (transitively dependent) on a candidate key, by virtue of being functionally dependent on another nonprime attribute. For example, consider a "Departments" table whose attributes are Department ID, Department Name, Manager ID, and Manager Hire Date; and suppose that each manager can manage one or more departments. {Department ID} is a candidate key. Although Manager Hire Date is functionally dependent on {Department ID}, it is also functionally dependent on the non-prime attribute Manager ID. This means the table is not in 3NF.

Boyce-Codd Normal Form

The criteria for Boyce-Codd Normal Form (BCNF) are:

- The table must be in 3NF.
- Every non-trivial functional dependency must be a dependency on a super key.

7.3Functional Dependency

Dependencies in my project are elaborated below

- First we have to take the initial table which has following fields
- Dist_name, Div_name, Mandal_name, Village_name, FPS_add, Deler_name,Licence_no, Date_Of_Issue, Ren_date, CSG_id, Rationcard_no, cardholder_name,card_type.

- In the above table we have redundant data because If we take the FPS's in one district say East-Godavari then we should again and again type that district name, resulting redundancy.
- Same situation occurs in the case of Div_name, Mandal_name, Village_name then we have to enter FPS details for each and every card holder.

Apply first normal form then we get

Dist_id	Dist_name

Div_id	Div_name
Mand_id	**Mand_name**

Vill_id	Vill_name

FPS_add	Deler_name	Lic_no	DOI	Ren_Date	CSG_id

Rationcard_no	Card_name	Card_type

If we are not apply the first normal form then we have anomalies (update, insert, delete)

- **Update anomaly**: if we make any update by using filed like dist_name, Div_name etc.. then all the fields which have the same dist_name, Div_name etc.. can also updated.
- **Delete anomaly**: if we delete any row by using repetition data field then we can loss the necessary data.
- **Insertion anomaly**: without FPS details we cannot insert card holder details.
- Applying second normal :
- Div_name is dependend on dist_id and div_id because according to second normal form, a relation scheme R is in 2Nf if every non attribute A in R is fully functionally depended in the primary key of R i.e., divi_name is partially depended on div_id and fully depeneded on dist_id and div_id.

Dist_id	Div_id	Div_name

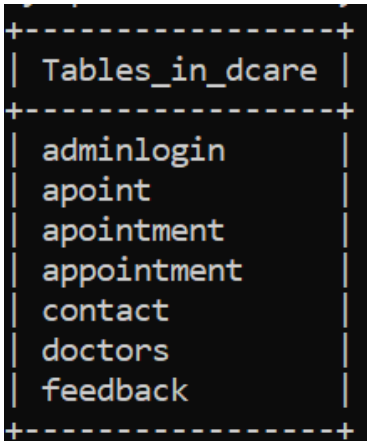
Dist_id	Div_id	Mand_id	Mand_name

Dist_id	Div_id	Mand_id	Vill_id	Vill_name

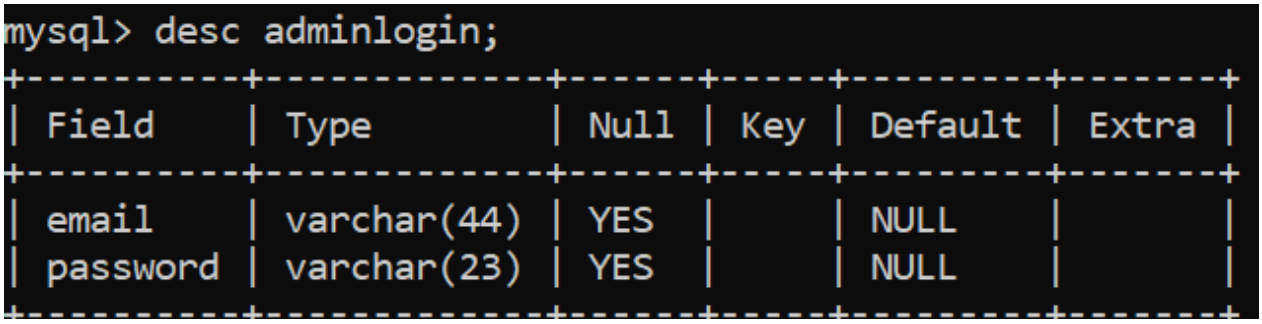
- Applying third normal form:
- A functional dependency X->Y in a relation scheme R is a transitive dependency if there is set of attributes Z i.e., neither a candidate Key nor a subset of any key of R and both X->Z and Z->Y hold.

  Here cardholder is dependent on FPS and FPS dependent on Dist_id, Div_id etc... i.e., rationcard_no is dependent on FPS Licence_no and FPS licence_no is dependent on dist_id, Div_id etc… so here transitive dependency holds. So rationalcard is dependent on Dist_id, div_id etc...

7.4 Database tables



Tables_in_dcare
adminlogin
apoint
apointment
appointment
contact
doctors
feedback



```
mysql> desc adminlogin;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| email    | varchar(44) | YES  |     | NULL    |       |
| password | varchar(23) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
```

```
mysql> desc apoint;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| fname     | varchar(23) | YES  |     | NULL    |       |
| lname     | varchar(23) | YES  |     | NULL    |       |
| selectopt | varchar(23) | YES  |     | NULL    |       |
| phone     | varchar(23) | YES  |     | NULL    |       |
| date      | varchar(23) | YES  |     | NULL    |       |
| time      | varchar(23) | YES  |     | NULL    |       |
| message   | varchar(23) | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
```

```
mysql> desc apointment;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| id       | varchar(45) | YES  |     | NULL    |       |
| name     | varchar(45) | YES  |     | NULL    |       |
| dob      | varchar(45) | YES  |     | NULL    |       |
| address  | varchar(45) | YES  |     | NULL    |       |
| gender   | varchar(45) | YES  |     | NULL    |       |
| contact  | varchar(45) | YES  |     | NULL    |       |
| email    | varchar(45) | YES  |     | NULL    |       |
| password | varchar(45) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
```

```
mysql> desc appointment;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| apid        | varchar(34) | YES  |     | NULL    |       |
| name        | varchar(34) | YES  |     | NULL    |       |
| email       | varchar(34) | YES  |     | NULL    |       |
| contact     | varchar(34) | YES  |     | NULL    |       |
| age         | varchar(34) | YES  |     | NULL    |       |
| day         | varchar(34) | YES  |     | NULL    |       |
| specialty   | varchar(34) | YES  |     | NULL    |       |
| description | varchar(34) | YES  |     | NULL    |       |
| id          | varchar(34) | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
9 rows in set (0.00 sec)
```

```
mysql> desc contact;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| name    | varchar(23) | YES  |     | NULL    |       |
| email   | varchar(23) | YES  |     | NULL    |       |
| subject | varchar(23) | YES  |     | NULL    |       |
| message | varchar(23) | YES  |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

```
mysql> desc doctors;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| id        | int(11)     | YES  |     | NULL    |       |
| docname   | varchar(66) | YES  |     | NULL    |       |
| email     | varchar(66) | YES  |     | NULL    |       |
| password  | varchar(66) | YES  |     | NULL    |       |
| specialty | varchar(66) | YES  |     | NULL    |       |
| contact   | varchar(66) | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

```
mysql> desc feedback;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| name       | varchar(55) | YES  |     | NULL    |       |
| email      | varchar(55) | YES  |     | NULL    |       |
| contact    | varchar(55) | YES  |     | NULL    |       |
| suggestion | varchar(55) | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

8 .TESTING

- The process of executing a system with the intent of finding an error.
- Testing is defined as the process in which defects are identified, isolated, subjected for rectification and ensured that product is defect free in order to produce the quality product and hence customer satisfaction.
- Quality is defined as justification of the requirements
- Defect is nothing but deviation from the requirements
- Defect is nothing but bug.
- Testing --- The presence of bugs.
- Testing can demonstrate the presence of bugs, but not their absence.
- Debugging and Testing are not the same thing.
- Testing is a systematic attempt to break a program or the AUT. Debugging is the art or method of uncovering why the script /program did not execute properly.

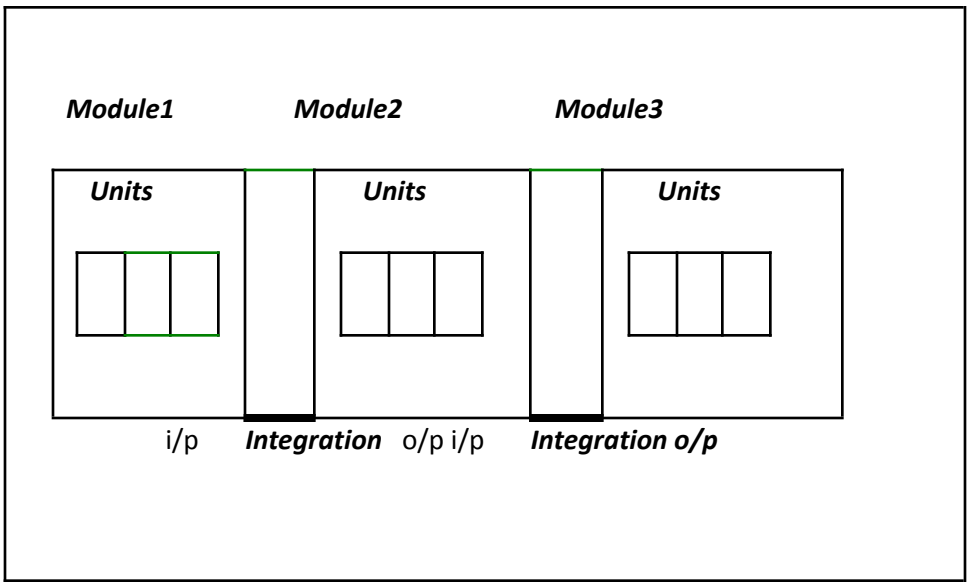
8.1Testing Methodologies

- **Black box Testing**: is the testing process in which tester can perform testing on an application without having any internal structural knowledge of application.
  Usually Test Engineers are involved in the black box testing.

- **White box Testing**: is the testing process in which tester can perform testing on an application with having internal structural knowledge.
  Usually The Developers are involved in white box testing.

- **Gray Box Testing**: is the process in which the combination of black box and white box techniques are used.

8.2 Levels of Testing



System Testing: Presentation + business +Databases

UAT: user acceptance testing

8.3 Types of Testing

- **Smoke Testing**: is the process of initial testing in which tester looks for the availability of all the functionality of the application in order to perform detailed testing on them. (Main check is for available forms)

- **Sanity Testing:** is a type of testing that is conducted on an application initially to check for the proper behavior of an application that is to check all the functionality are available before the detailed testing is conducted by on them.

- **Regression Testing:** is one of the best and important testing. Regression testing is the process in which the functionality, which is already tested before, is once again tested whenever some new change is added in order to check whether the existing functionality remains same.

- **Re-Testing:** is the process in which testing is performed on some functionality which is already tested before to make sure that the defects are reproducible and to rule out the environments issues if at all any defects are there.

- **Static Testing:** is the testing, which is performed on an application when it is not been executed. ex: GUI, Document Testing

- **Dynamic Testing:** is the testing which is performed on an application when it is being executed. ex: Functional testing.

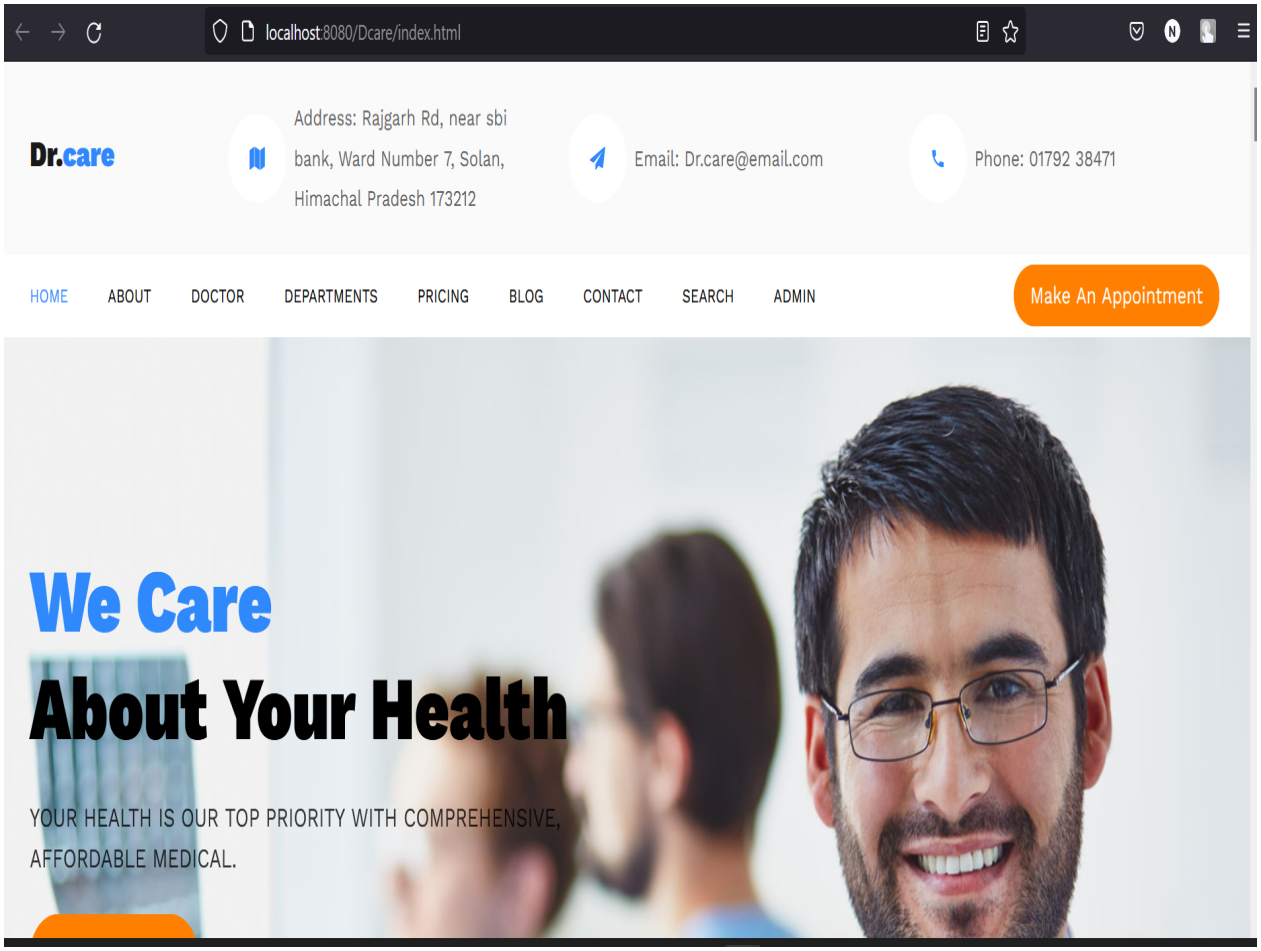
- **Alpha Testing:** it is a type of user acceptance testing, which is conducted on an application when it is just before released to the customer.

- **Monkey Testing:** is the process in which abnormal operations, beyond capacity operations are done on the application to check the stability of it in spite of the users abnormal behavior.

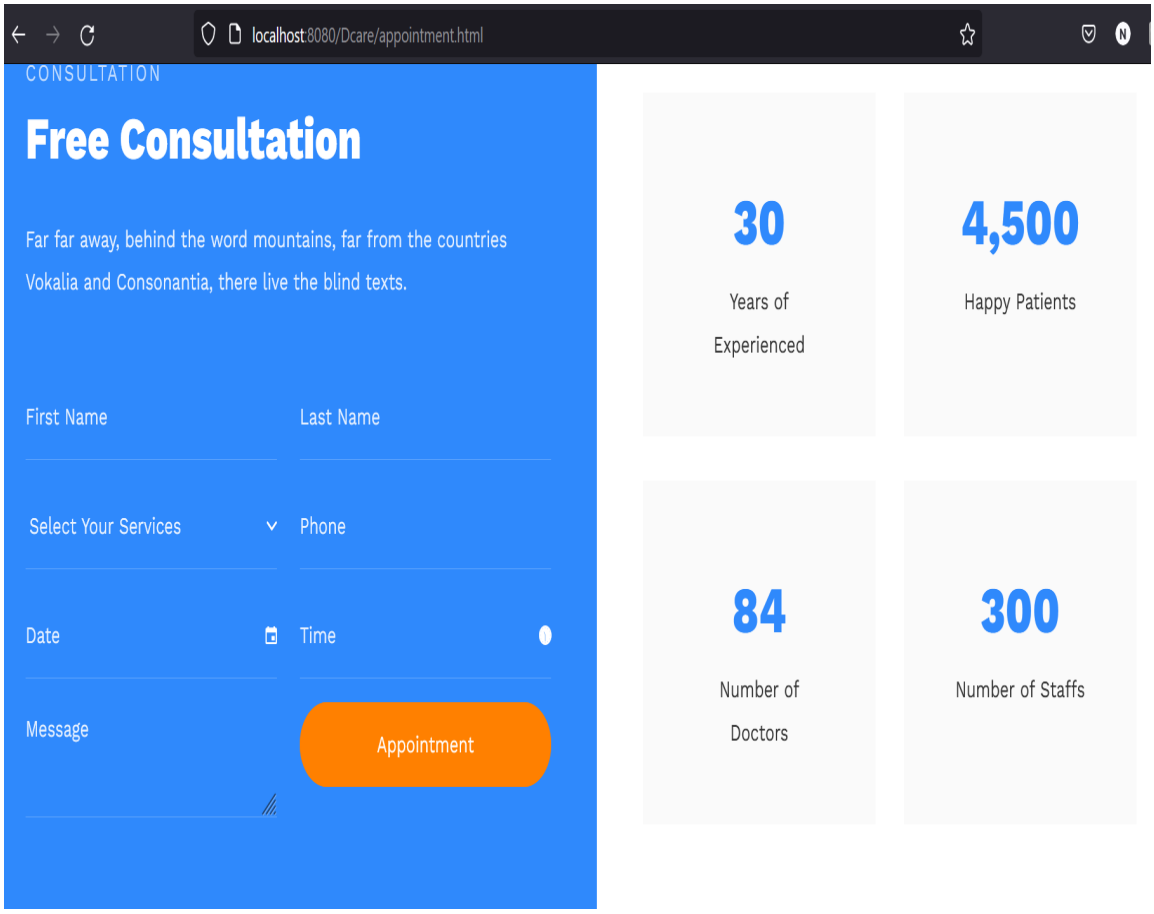
- **Compatibility testing:** it is the testing process in which usually the products are tested on the environments with different combinations of databases.
- **Installation Testing:** it is the process of testing in which the tester try to install or try to deploy the module into the corresponding environment by following the guidelines produced in the deployment document and check whether the installation is successful or not.

9 .OUTPUT SCREENS

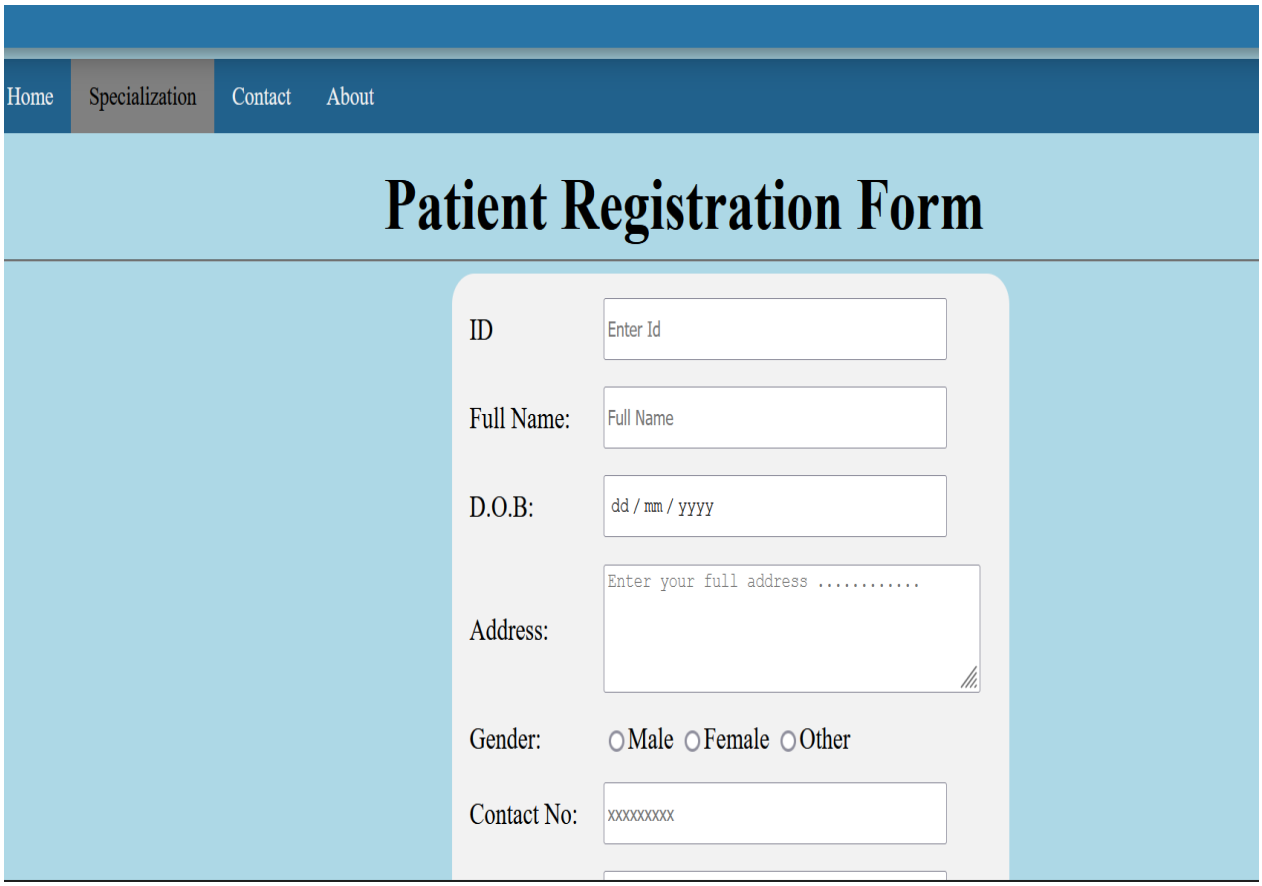
Home page:



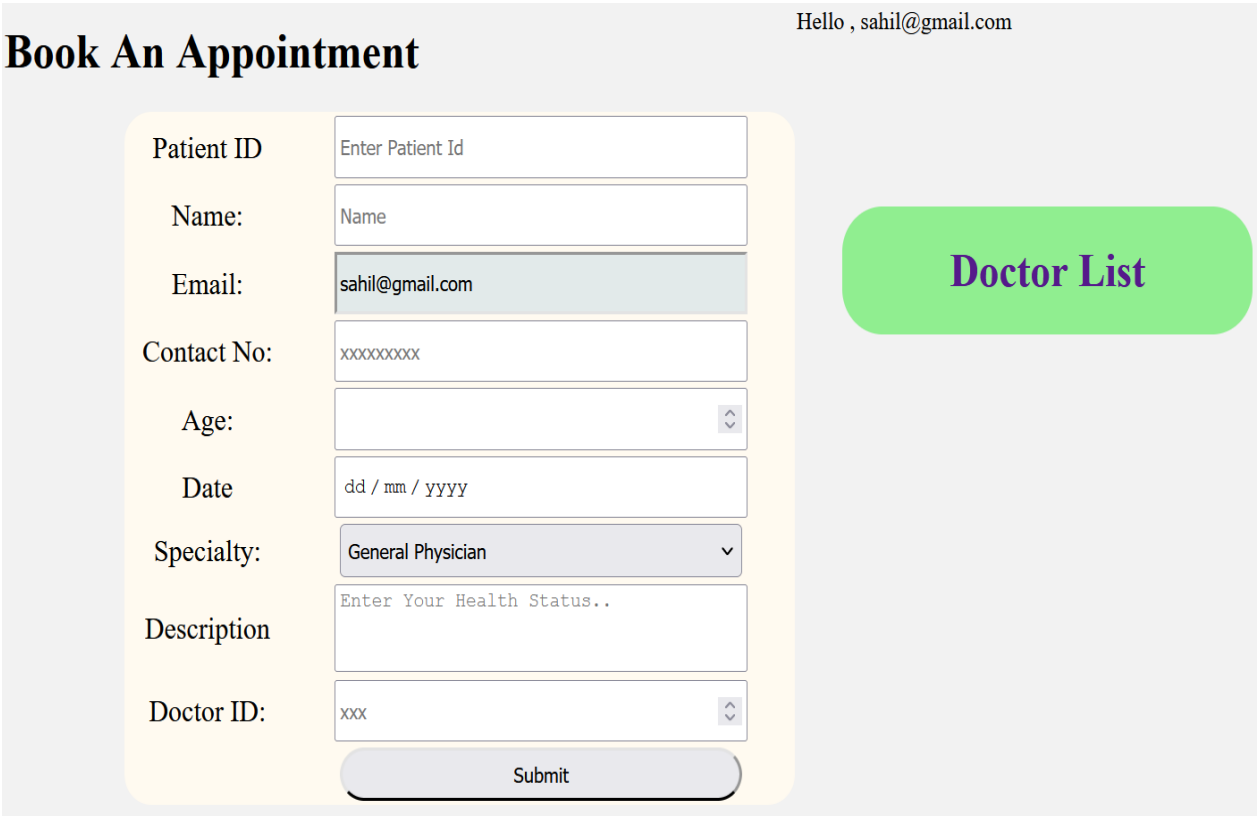
New user Registration page:



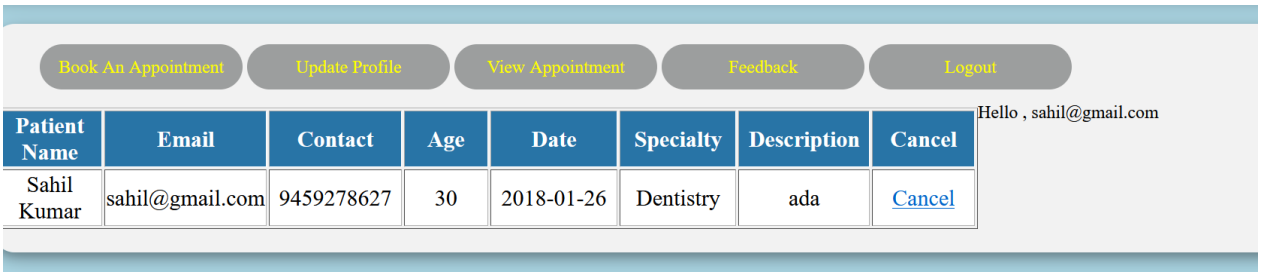
Patient Registration:



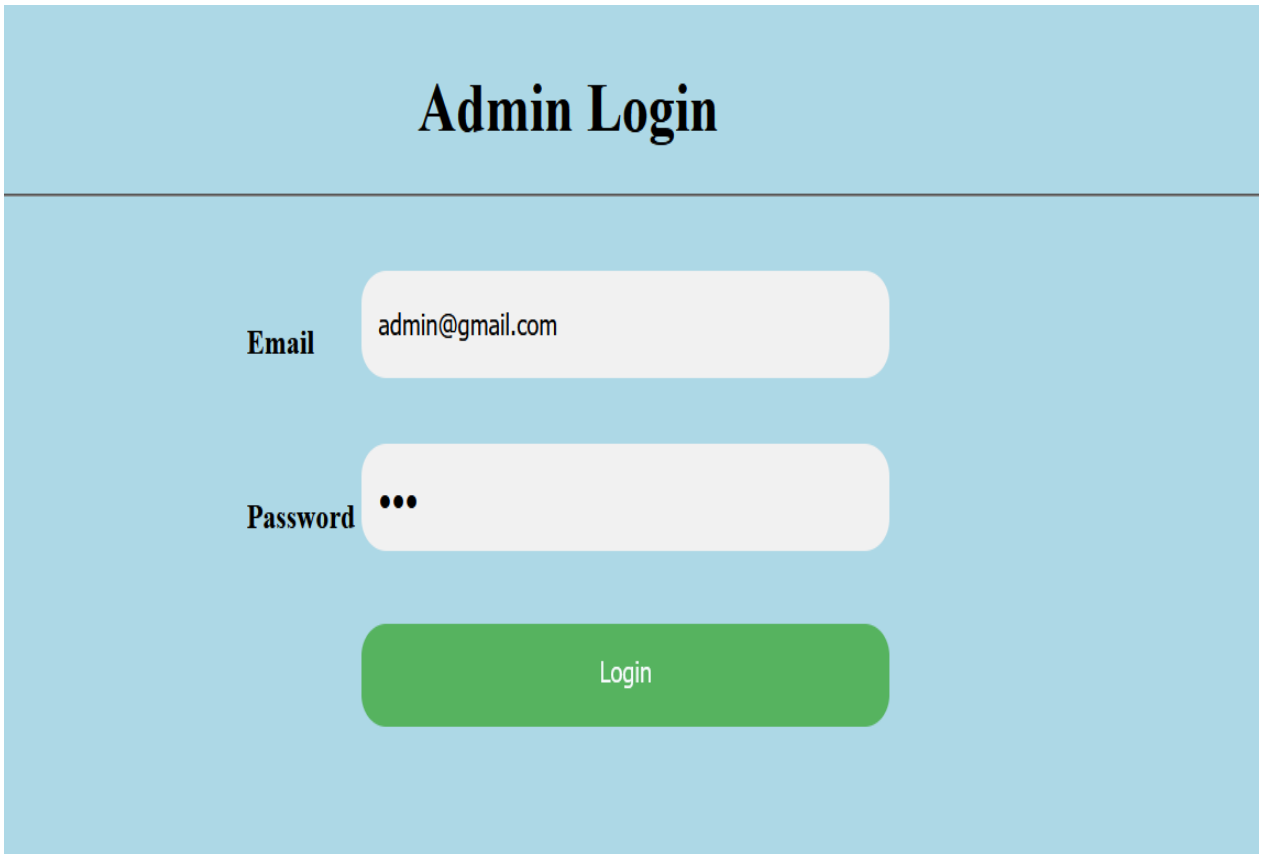
Appointment Registration page:



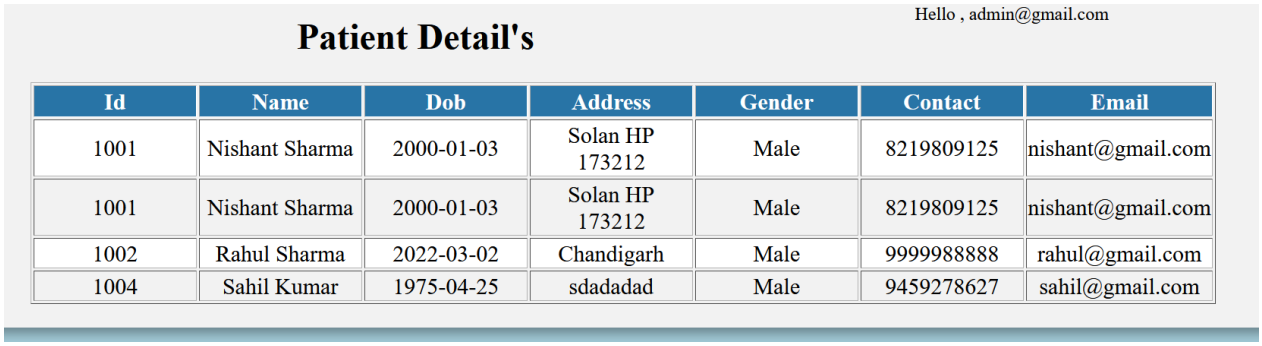
Appointment Request Sent:



Admin Login page:



Patient Request :

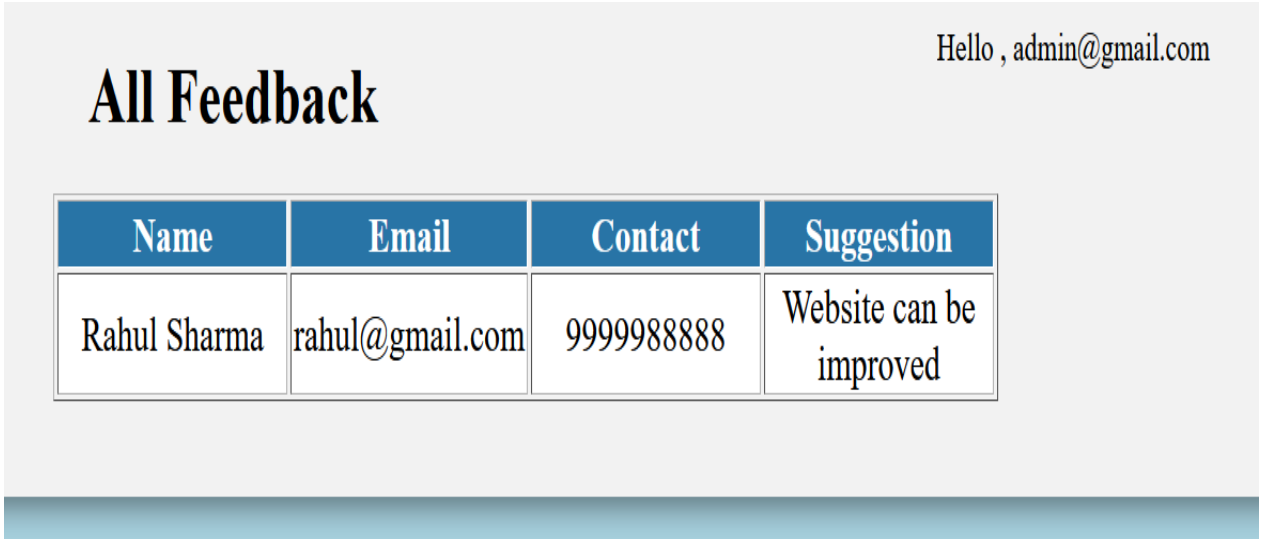


Id	Name	Dob	Address	Gender	Contact	Email
1001	Nishant Sharma	2000-01-03	Solan HP 173212	Male	8219809125	nishant@gmail.com
1001	Nishant Sharma	2000-01-03	Solan HP 173212	Male	8219809125	nishant@gmail.com
1002	Rahul Sharma	2022-03-02	Chandigarh	Male	9999988888	rahul@gmail.com
1004	Sahil Kumar	1975-04-25	sdadadad	Male	9459278627	sahil@gmail.com

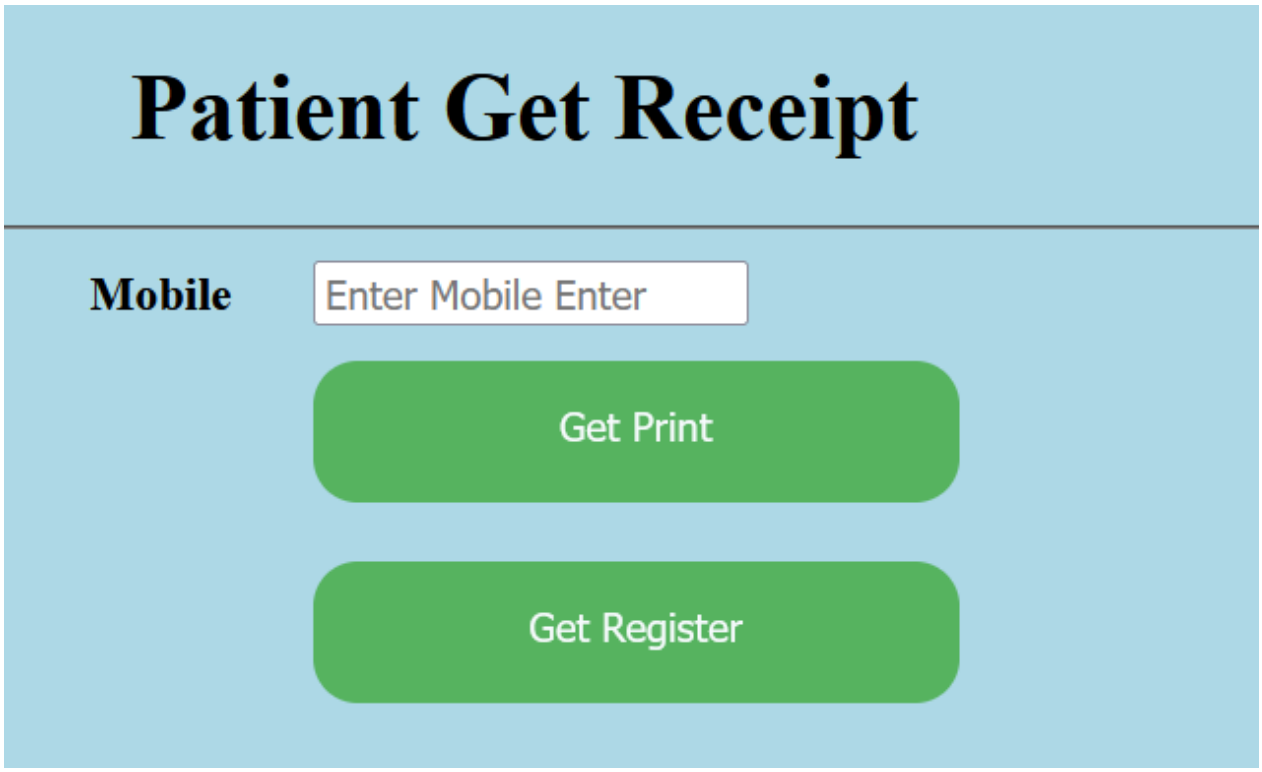
Add Doctor:


Hello , admin@g
Add Doctor
Doctor ID:
Enter ID
Doctor Name:
Full Name
Email Id:
xyz@gmail.com
Password :
****
Specialty:
General Physician
Contact No:
xxxxxxxxx
Submit

View Patient Feedback:



Bill Generation:



10. Future Enhancements

We can enhance this system by including more facilities like a pharmacy system for the stock details of medicines in the pharmacy.

Ambulance facility can also be added to the website.

11. Limitations

The size of the database increases day-by-day, increasing the load on the database backup and data maintenance activity.

Training for simple computer operations is necessary for the users working on the system.

12 .CONCLUSION

The growing quality demand in the hospital sector makes it necessary to exploit the whole potential of stored data efficiently, not only the clinical data, in order to improve diagnoses and treatments, but also on management, in order to minimize costs and improve the care given to the patients.

In this sense, Data Mining (DM) can contribute with important benefits to the health sector, as a fundamental tool to analyze the data gathered by hospital information systems (HIS) and obtain models and patterns which can improve patient assistance and a better use of resources and pharmaceutical expense.

13. BIBLIOGRAPHY

1. https://www.w3schools.com/
2. https://www.javatpoint.com/
3. https://www.tutorialspoint.com/
4. https://docs.oracle.com/