A

Major Project Report On

"A Blockchain Based System for Healthcare Digital Twin"

Submitted to JNTUH in partial fulfilment of the Requirements
for the award of the Degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE & ENGINEERING
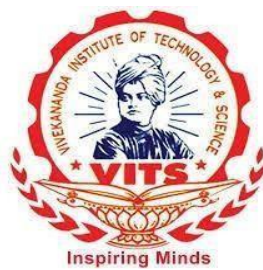
By

PARSIVAR MANIDEEP   **21N61A0515**

Under the Guidance of

Mrs. Thangallapally Sahithi
Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
VIVEKANANDA INSTITUTE OF TECHNOLOGY AND SCIENCE
(Approved by AICTE New Delhi & Affiliated to JNTU, Hyderabad
An ISO 9001:2015 certified Institute)
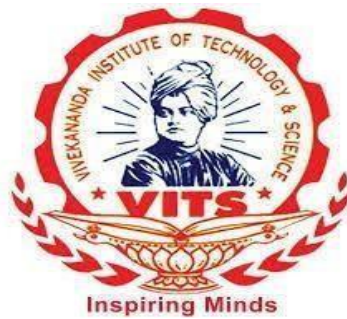**KARIMNAGAR-505501 2024-2025**

CERTIFICATE

This is to certify that the Major-project report titled **"A Blockchain Based System for Healthcare Digital Twin"** is being submitted by **PARSIVAR MANIDEEP 21N61A0515,** in B. Tech IV-II semester, Computer Science & Engineering is a record Bonafide work carried out by him. The results embodied in this report have not been submitted to any other University for the award of any degree.

Internal Guide   **Head of the Department**

Mrs. Thangallapally Sahithi   **Dr.M.V.Hanumantha Reddy**

External Examiner   **Principal**

                                                                      Dr.M.V.Hanumantha Reddy

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to our guide, **Mrs. Thangallapally Sahithi Assistant Professor** whose knowledge and guidance has motivated us to achieve goals we never thought possible. She has consistently been a source of motivation, encouragement, and inspiration. The time we have spent working under her supervision has truly been a pleasure.

We thank our H.O.D **Dr.M.V.Hanumanthareddy** for his effort and indefatigable guidance rendered throughout the progress of project work.

Thanks to programmers and nonteaching staff of CSE Department of VITS(N6).

Finally, Special thanks to our parents for their support and encouragement throughout this course. And thanks to our friends and well-wishers for their constant support.

NAME OF TEAM MEMBER	ROLL NO:
PARSIVAR MANIDEEP	**21N61A0515**

ABSTRACT

Digital Twin (DT) is an emerging technology that replicates any physical phenomenon from a physical space to a digital space in congruence with the physical state. However, devising a Healthcare DT model for patient care is seen as a challenging task as the lack of adequate data collection structure. There are also security and privacy concerns as healthcare data is very sensitive and can be used in malicious ways. Because of these current research gaps, the proper way of acquiring the structured data and managing them in a secure way is very important.

In this article, we present a mathematical data model to accumulate the patient relevant data in a structured and predefined way with proper delineation. Additionally, the provided data model is described in harmony with real life contexts. Then, we have used the patient centric mathematical data model to formally dene the semantic and scope of our proposed Healthcare Digital Twin (HDT) system based on Blockchain. Accordingly, the proposed system is described with all the key components as well as with detailed protocol flows and an analysis of its different aspects. Finally, the feasibility of the proposed model with a critical comparison with other relevant research works have been provided.

INDEX

1. INTRODUCTION

According to the latest statistics from the World Health Organization, about 930 million people worldwide are at risk of falling into poverty due to out-of-pocket health spending of 10% or more of their household budget [1]. Currently, there is a surge in improving the healthcare situation and a myriad of developments are ongoing in the healthcare sector with respect to Artificial Intelligence [2] _ [4], Big data [5], [6], and in another spectrum. Though, it is not something that can be assuaged outright, whereas the real problem is not in the slow advancement of the technology, rather the mishaps in real life, e.g., adverse events, late diagnosis, etc [7]. DT can bring an immediate alteration in the healthcare sector from its root by incorporating analysis, predictive measurements, decision making paradigm, and data collection [8].

There are some notable developments in the healthcare sector incorporating DT. Martinez-Velazquez *et al.* [9] have developed a cardio twin based on the heart that can mitigate the risk of any ischemic heart disease. Barbiero *et al.* [10] have proposed a general framework to provide a panoramic view over current and future physiological conditions. However, the recent developments in DT for the healthcare sector, have some drawbacks from the perspective of data sharing, storage, and access control [11]. Also, without any proper framework, collecting a large amount of data haphazardly will cause a disarray which will perpetuate when involving other data transformation techniques [12]. For these reasons, it is a prominent task to decide in which way DT will perceive which healthcare data from which dimensions [13]. To solve these problems, we propose a structured mathematical data model to collect the patients' data in a systematic and preened way so that a cluster of acute information about a physical patient and its surrounding environments can be accumulated while they are at the hospital. With the proposed data model, the patient can be individually identified as well as the patient portfolio can be concocted with the clinical data.

It is often reported that people show a lack of concern regarding the security of the health data which leads to integrity and confidentiality breaches [14]. Around 881 breach reports have been recorded within the last 24 months and are under investigation by the U.S. Department of Health & Human Services [15]. Therefore, to effectively solve this problem, a system is needed that can store and keep data securely with proper structure. Moreover, around 60% of the counties in the world have the capacity to review the progress and performance of the healthcare systems and around 59% can use data to drive policies and planning for the health sectors [16]. To cover these wide distributed nationwide healthcare sectors, having the mentioned potentials, a distributed network can be implemented by enforcing a distributed storage facility without any central governing authority [17]. For these reasons, the block chain technology can be integrated with DT to accumulate this insurmountable health data in a structured and distributed way with adequate security properties. In a block chain-based DT system for healthcare, block chain renders the services of collecting intricate and diverse data immutably with proper access and sharing mechanism, on the other hand, DT provides proper data analysis, aggregation, prognosis, and representation services which are conducive to build a proper healthcare DT. To mitigate these issues, in this article, we present a concrete mathematical model for patients' clinical data and then propose a block chain-based Healthcare Digital Twin system based on the presented data model.

1.2 Existing System

❖ Peng *et al.* [53], in their article have presented a construction case on hospital DT in China, which had already been built. The authors have delineated how the hospital twin has been developed based on Continuous Lifecycle Integration method. A lot of sensors, for acquiring real time data of the hospital, have been planted during construction and the whole system can be controlled from a single point through DT. However, there is nothing mentioned about access control and encryption mechanisms for the collected data.

❖ Liu *et al.* [54] have proposed a cloud-based framework with healthcare DT. The reason behind the project is that there are elder people who hardly take medical services because of their indifference toward diseases. The authors have developed the system comprising of 4 parts: Physical object, Virtual object, Cloud healthcare service platform, and healthcare data. Although, some important aspects have been described, however, no algorithm has been mentioned for the predictive measures.

❖ Shamanna *et al.* [55], introducing Precision Nutrition to DT. The paper is about Twin Precision Nutrition (TPN) which monitors a group of 64 years old type 2 diabetic patients to reduce HbA1c in blood. The platform collects data from body sensors and a mobile app (TPN) to track and analyse the body health signals in order to personalize the patients' treatment. Although the system is devising results based on real time data, the authors have not provided any mechanism by which they have conducted the analysis.

❖ Barbiero *et al.* [56], in their article have proposed an architecture combining the qualities of a generative model with a graph-based representation of pathophysiological conditions.

1.3 Disadvantages of Existing system

❖ The system is not implemented public blockchain and only implemented private blockchain in which security is very less.

❖ The data is not accumulated while patient goes through disease assessment phase.

1.4 Proposed System

1) A patient centric mathematical data model to represent the patient data in a defined and structured way.

2) The proper delineation of the clinical data with real life contexts which will be perceived by DT while the patient is on the treatment phase.

3) A blockchain integrated Healthcare Digital Twin System architecture based on the proposed data model with proper threat modelling and requirement analysis.

4) A number of protocol flows utilizing the blockchain based system which showcases how the system can be utilized in different scenarios.

5) A detailed analysis of the proposed system covering its feasibility, advantages/disadvantages, comparisons with Health Insurance Portability and Accountability Act (*HIPAA*) [18] and the General Data Protection Regulation (*GDPR*) [19] as well as with other existing research works.

6) Finally, the limitations and the future scopes of the presented system.

1.5 Advantages of Proposed System

➢ The system is implemented DT which stands for the representation of the anatomy of a digital asset in a digital space which is the depiction of a physical phenomena from a physical space.

➢ In the proposed system, the system is implemented DIGITAL TWIN FOR DEVELOPING A PRODUCT and DIGITAL TWIN FOR AN INDIVIDUAL INSTANCE which are more secure and safe.

2.PRELIMINARY INVESTIGATION

2.1 System Design

Input design

Input Design plays a vital role in the life cycle of software development; it requires very careful attention of developers. The input design is to feed data to the application as accurate as possible. So, inputs are supposed to be designed effectively so that the errors occurring while feeding are minimized. According to Software Engineering Concepts, the input forms or screens are designed to provide to have a validation control over the input limit, range and other related validations.

This system has input screens in almost all the modules. Error messages are developed to alert the user whenever he commits some mistakes and guides him in the right way so that invalid entries are not made. Let us see deeply about this under module design.

Input design is the process of converting the user created input into a computer-based format. The goal of the input design is to make the data entry logical and free from errors. The error is in the input are controlled by the input design. The application has been developed in user-friendly manner. The forms have been designed in such a way during the processing the cursor is placed in the position where must be entered. The user is also provided with in an option to select an appropriate input from various alternatives related to the field in certain cases.

Validations are required for each data entered. Whenever a user enters an erroneous data, error message is displayed and the user can move on to the subsequent pages after completing all the entries in the current page.

Output design

The Output from the computer is required to mainly create an efficient method of communication within the company primarily among the project leader and his team members, in other words, the administrator and the clients. The output of VPN is the system which allows the project leader to manage his clients in terms of creating new clients and assigning new projects to them, maintaining a record of the project validity and providing folder level access to each client on the user side depending on the projects allotted to him. After completion of a project, a new project may be assigned to the client. User authentication procedures are maintained at the initial stages itself. A new user may be created by the administrator himself or a user can himself register as a new user but the task of assigning projects and validating a new user rest with the administrator only.

The application starts running when it is executed for the first time. The server has to be started and then the internet explorer in used as the browser. The project will run on the local area network so the server machine will serve as the administrator while the other connected systems can act as the clients. The developed system is highly user friendly and can be easily understood by anyone using it even for the first time.

3. SYSTEM REQUIREMENT SPECIFICATION

3.1 Hardware Requirements:

- ➢ Processor   -   Pentium –IV

- ➢ RAM   -   4 GB (min)

- ➢ Hard Disk   -   20 GB

- ➢ Key Board   -   Standard Windows Keyboard

- ➢ Mouse   -   Two or Three Button Mouse

- ➢ Monitor   -   SVGA

3.2 Software Requirements:

- ➢ Operating System   -   Windows XP
- ➢ Coding Language   -   Java/J2EE(JSP,Servlet)
- ➢ Front End   -   J2EE
- ➢ Back End   -   MySQL

3.3 Feasibility Study

An important outcome of preliminary investigation is the determination that the system request is feasible. This is possible only if it is feasible within limited resource and time. The different feasibilities that have to be analysed are

- **Operational Feasibility**
- **Economic Feasibility**
- **Technical Feasibility**

Operational Feasibility

Operational Feasibility deals with the study of prospects of the system to be developed. This system operationally eliminates all the tensions of the admin and helps him in effectively tracking the project progress. This kind of automation will surely reduce the time and energy, which previously consumed in manual work. Based on the study, the system is proved to be operationally feasible.

Economic Feasibility

Economic Feasibility or Cost-benefit is an assessment of the economic justification for a computer-based project. As hardware was installed from the beginning & for lots of purposes thus the cost on project of hardware is low. Since the system is a network based, any number of employees connected to the LAN within that organization can use this tool from at any time. The Virtual Private Network is to be developed using the existing resources of the organization. So, the project is economically feasible.

Technical Feasibility

According to Roger S. Pressman, Technical Feasibility is the assessment of the technical resources of the organization. The organization needs IBM compatible machines with a graphical web browser connected to the Internet and Intranet. The system is developed for platform independent environment. Java Server Pages, JavaScript, HTML, SQL server and WebLogic Server are used to develop the system. The technical feasibility has been carried out. The system is technically feasible for development and can be developed with the existing facility.

4. DESIGN METHODOLOGY
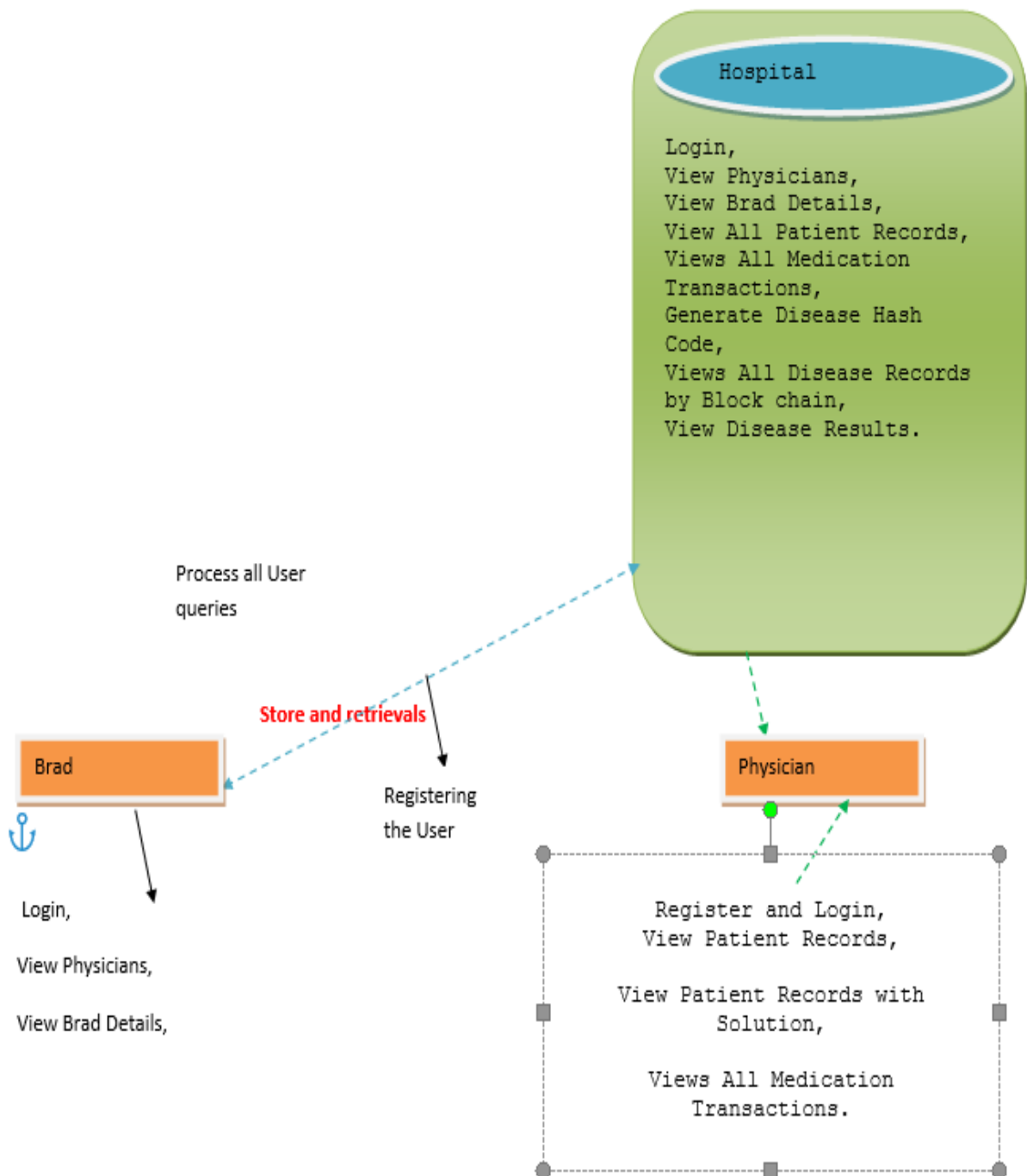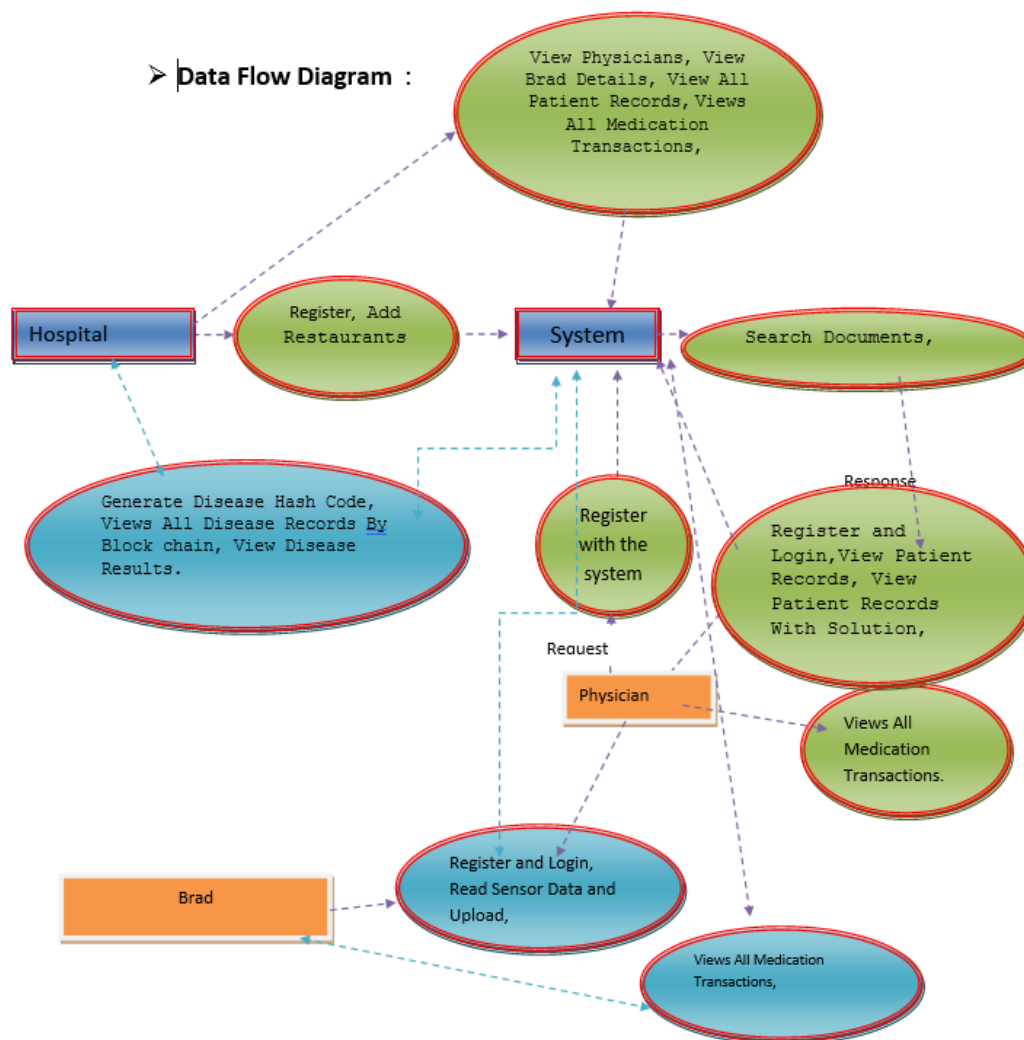
4.1 Architecture Diagram

Fig: 4.1 Architecture Diagram

4.2 Dataflow Diagram

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.

2. The data flow diagram (DFD) is one of the most important modelling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.

3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.

➤ **Data Flow Diagram :**

View Physicians, View Brad Details, View All Patient Records, Views All Medication Transactions,

Hospital

Register, Add Restaurants

System

Search Documents,

Generate Disease Hash Code, Views All Disease Records By Block chain, View Disease Results.

Register with the system

Response

Register and Login, View Patient Records, View Patient Records With Solution,

Request

Physician

Views All Medication Transactions.

Brad

Register and Login, Read Sensor Data and Upload,

Views All Medication Transactions,

4.3 UML Diagrams

UML stands for Unified Modelling Language. UML is a standardized general-purpose modelling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: a Metamodel and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modelling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modelling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modelling of large and complex systems.

The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS:

The Primary goals in the design of the UML are as follows:

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

USE CASE DIAGRAM:

A use case diagram in the Unified Modelling Language (UML) is a type of behavioural diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.
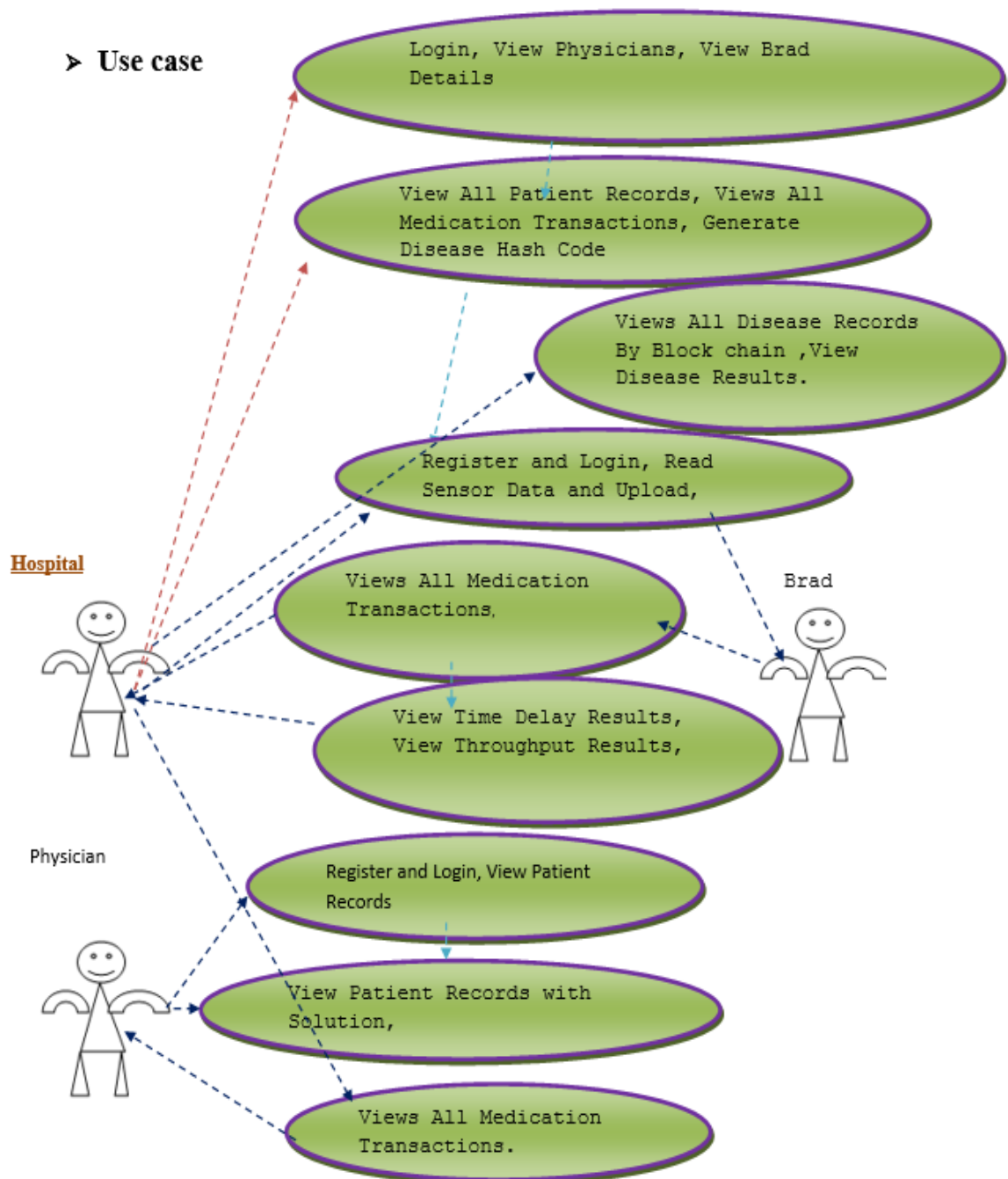

Figure 1Fig: 4.3.1 USE CASE DIAGRAM

12

CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modelling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

➢ **Class Diagram:**

Hospital

Methods	Login, View Physicians, View Brad Details, View All Patient Records, Views All Medication Transactions, Generate Disease Hash Code, Views All Disease Records by Block chain, View Disease Results.
Members	ID, Report Name, Medical Contents, Trapdoor, Rank, Date & Time, Data Owner, Patient Name, Medical History, Disease, Age, Gender, Address.

Brad

Methods	Register and Login, Read Sensor Data and Upload, Views All Medication Transactions,
Members	ID, Report Name, Medical Contents, Trapdoor, Rank, Date & Time, Data Owner, Patient Name, Medical History, Disease, Age, Gender, Address.

Login

Methods	Login (), Reset (), Register ().
Members	User Name, Password.

Register

Methods	Register (), Reset ()
Members	User Name, Password, E-mail, Mobile, Address, DOB, Gender, Pin code, Image

Physician

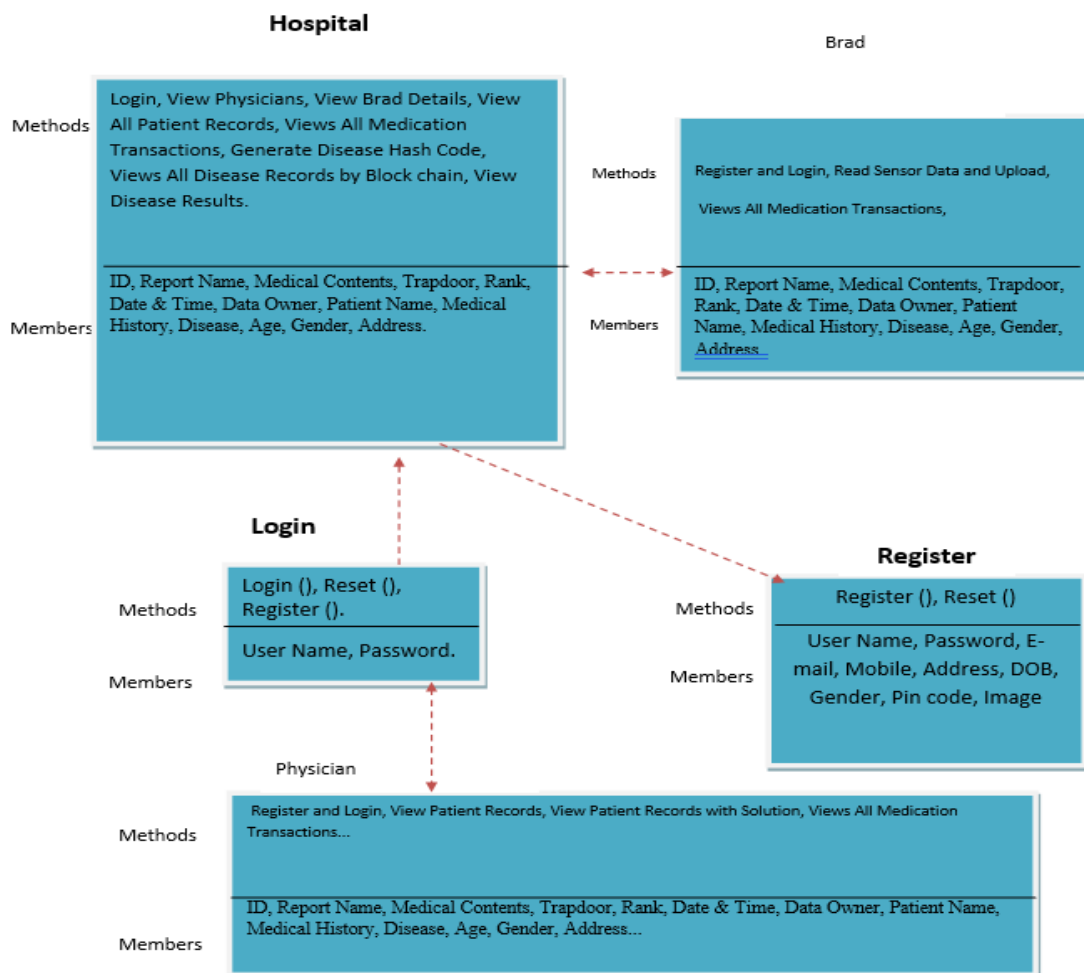|	
Methods	Register and Login, View Patient Records, View Patient Records with Solution, Views All Medication Transactions...
Members	ID, Report Name, Medical Contents, Trapdoor, Rank, Date & Time, Data Owner, Patient Name, Medical History, Disease, Age, Gender, Address...

Fig: 4.3.2 CLASS DIAGRAM

SEQUENCE DIAGRAM:

A sequence diagram in Unified Modelling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.
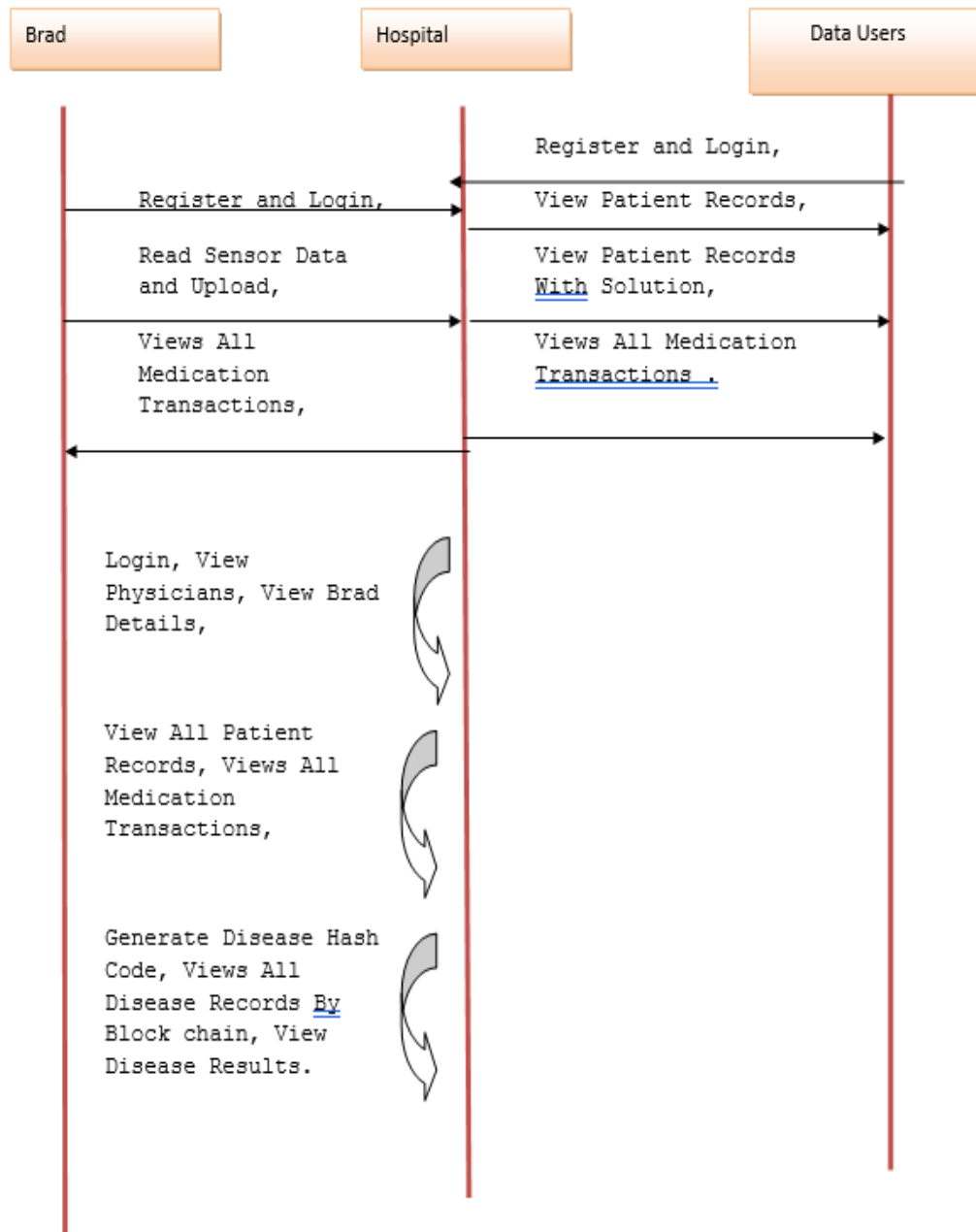

Fig: 4.3.3 SEQUENCE DIAGRAM

5.IMPLEMENTATION METHODOLOGY

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus, it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective. The implementation stage involves careful planning, investigation of the existing system and its constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods. Implementation is the process of converting a new system design into option.

It is the phase that focuses on user training, site preparation and file conversion for installing a candidate system. The important factor that should be considered here is that the conversion should not disrupt the functioning of the organization.

The application is implemented in the Internet Information Services 5.0 web server under the windows 2000 Professional and accessed from various clients. An analysis of user training focuses on two factors

- User capabilities

- Nature of the system

Users range from the native to highly sophisticated. Hence, they should be trained about the usage of software. The user should take care to see that in the event of interruption due to power failure.

6. SOFTWARE ENVIRONMENT

6.1 JAVA TECHNOLOGY

Java technology is both a programming language and a platform.

The Java programming language is a high-level language that can be characterized by all of the following buzzwords:

- Simple
- Architecture neutral
- Object oriented
- Portable
- Distributed
- High performance
- Interpreted
- Multithreaded
- Robust
- Dynamic
- Secure

With most programming languages, you either compile or interpret a program so that you can run it on your computer. The Java programming language is unusual in that a program is both compiled and interpreted. With the compiler, first you translate a program into an intermediate language called Java byte codes —the platform-independent codes interpreted by the interpreter on the Java platform. The interpreter parses and runs each Java byte code instruction on the computer. Compilation happens just once; interpretation occurs each time the program is executed. The following figure illustrates how this works.
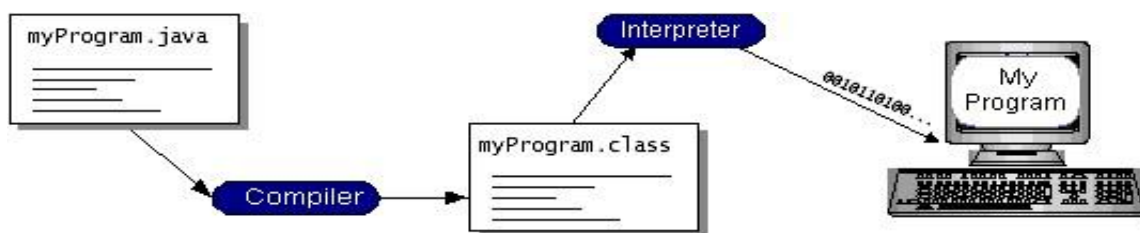
Fig: 6.1 Java Programmig Structure

You can think of Java byte codes as the machine code instructions for the Java Virtual Machine (Java VM). Every Java interpreter, whether it's a development tool or a Web browser that can run applets, is an implementation of the Java VM. Java byte codes help make "write once, run anywhere" possible. You can compile your program into byte codes on any platform that has a Java compiler. The byte codes can then be run on any implementation of the Java VM. That means that as long as a computer has a Java VM, the same program written in the Java programming language can run on Windows 2000, a Solaris workstation, or on an iMac.
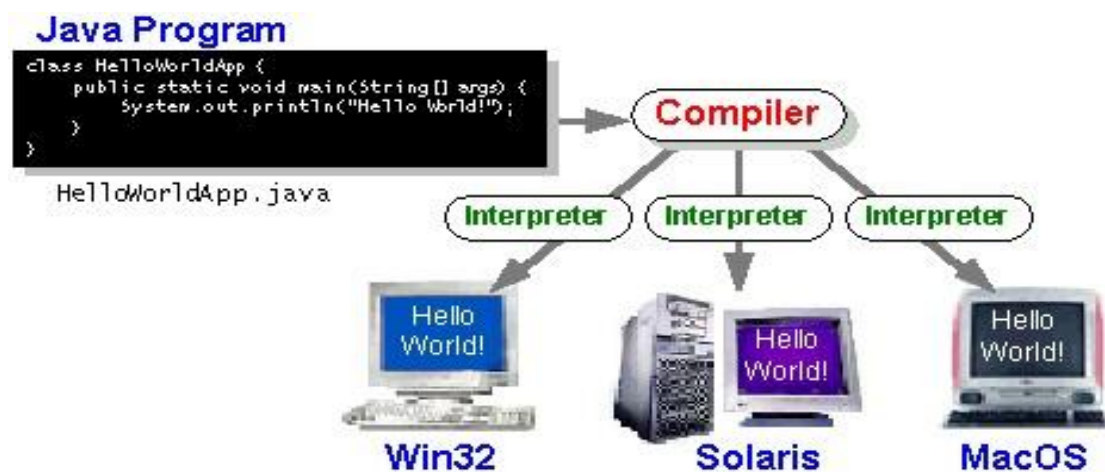
Fig: 6.1.2 Java Compiler Can Run Any Where

The Java Platform

A platform is the hardware or software environment in which a program runs. We've already mentioned some of the most popular platforms like Windows 2000, Linux, Solaris, and MacOS. Most platforms can be described as a combination of the operating system and hardware. The Java platform differs from most other platforms in that it's a software-only platform that runs on top of other hardware-based platforms.

The Java platform has two components:

- The Java Virtual Machine (Java VM)
- The Java Application Programming Interface (Java API)

You've already been introduced to the Java VM. It's the base for the Java platform and is ported onto various hardware-based platforms.

The Java API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets. The Java API is grouped into libraries of related classes and interfaces; these libraries are known as packages.

The following figure depicts a program that's running on the Java platform. As the figure shows, the Java API and the virtual machine insulate the program from the hardware.
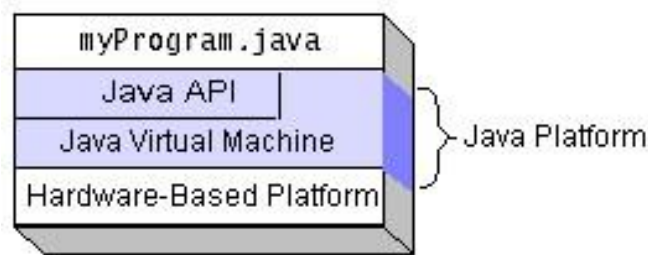

Fig: 6.1.3 Java Platform Diagram

Native code is code that after you compile it, the compiled code runs on a specific hardware platform. As a platform-independent environment, the Java platform can be a bit slower than native code. However, smart compilers, well-tuned interpreters, and just-in-time byte code compilers can bring performance close to that of native code without threatening portability.

What Can Java Technology Do?

The most common types of programs written in the Java programming language are applets and applications. If you've surfed the Web, you're probably already familiar with applets. An applet is a program that adheres to certain conventions that allow it to run within a Java-enabled browser.

However, the Java programming language is not just for writing cute, entertaining applets for the Web. The general-purpose, high-level Java programming language is also a powerful software platform. Using the generous API, you can write many types of programs.

An application is a standalone program that runs directly on the Java platform. A special kind of application known as a server serves and supports clients on a network. Examples of servers are Web servers, proxy servers, mail servers, and print servers. Another specialized program is a servlet. A servlet can almost be thought of as an applet that runs on the server side. Java

Servlets are a popular choice for building interactive web applications, replacing the use of CGI scripts. Servlets are similar to applets in that they are runtime extensions of applications.

Instead of working in browsers, though, servlets run within Java Web servers, configuring or tailoring the server.

Every full implementation of the Java platform gives you the following features:

The essentials:

Objects, strings, threads, numbers, input and output, data structures, system properties, date and time, and so on.

Applets:

The set of conventions used by applets.

Networking:

URLs, TCP (Transmission Control Protocol), UDP (User Data gram Protocol) sockets, and IP (Internet Protocol) addresses.

Internationalization:

Help for writing programs that can be localized for users worldwide. Programs can automatically adapt to specific locales and be displayed in the appropriate language.

Security:

Both low level and high level, including electronic signatures, public and private key management, access control, and certificates.

Software components:

Known as JavaBeans, can plug into existing component architectures.

Object serialization:

Allows lightweight persistence and communication via Remote Method Invocation (RMI).

Java Database Connectivity (JDBCTM):

Provides uniform access to a wide range of relational databases.

The Java platform also has APIs for 2D and 3D graphics, accessibility, servers, collaboration, telephony, speech, animation, and more. The following figure depicts what is included in the Java 2 SDK.
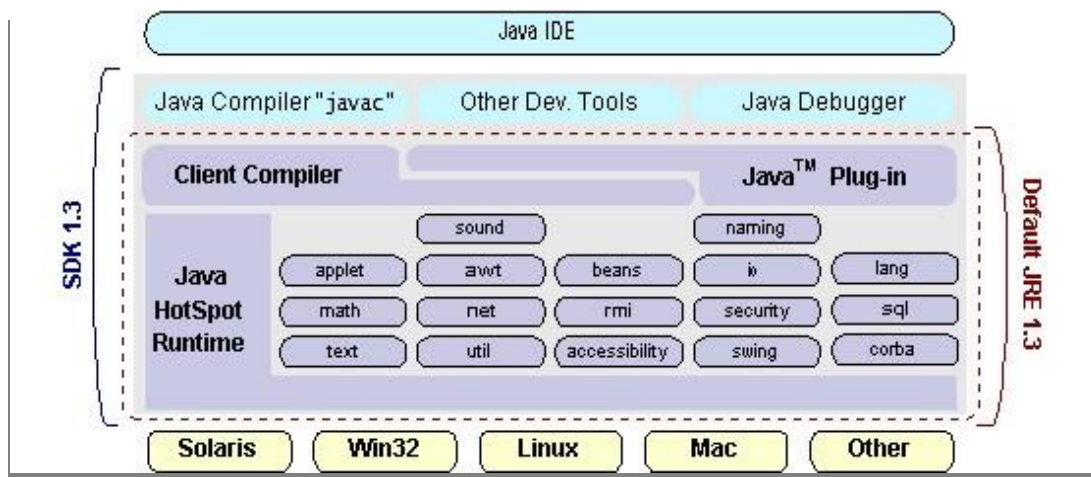
Fig: 6.1.4 Java SDK

How Will Java Technology Change My Life?

We can't promise you fame, fortune, or even a job if you learn the Java programming language. Still, it is likely to make your programs better and requires less effort than other languages. We believe that Java technology will help you do the following:

Get started quickly:

Although the Java programming language is a powerful object-oriented language, it's easy to learn, especially for programmers already familiar with C or C++.

Write less code:

Comparisons of program metrics (class counts, method counts, and so on) suggest that a program written in the Java programming language can be four times smaller than the same program in C++.

Write better code:

The Java programming language encourages good coding practices, and its garbage collection helps you avoid memory leaks. Its object orientation, its JavaBeans component architecture, and its wide-ranging, easily extendible API let you reuse other people's tested code and introduce fewer bugs.

Develop programs more quickly:

Your development time may be as much as twice as fast versus writing the same program in C++. Why? You write fewer lines of code and it is a simpler programming language than C++.

Avoid platform dependencies with 100% Pure Java:

You can keep your program portable by avoiding the use of libraries written in other languages. The 100% Pure JavaTM Product Certification Program has a repository of historical process manuals, white papers, brochures, and similar materials online.

Write once, run anywhere:

Because 100% Pure Java programs are compiled into machine-independent byte codes, they run consistently on any Java platform.

6.2 ODBC

Microsoft Open Database Connectivity (ODBC) is a standard programming interface for application developers and database systems providers. Before ODBC became a de facto standard for Windows programs to interface with database systems, programmers had to use proprietary languages for each database they wanted to connect to. Now, ODBC has made the choice of the database system almost irrelevant from a coding perspective, which is as it should be. Application developers have much more important things to worry about than the syntax that is needed to port their program from one database to another when business needs suddenly change.

Through the ODBC Administrator in Control Panel, you can specify the particular database that is associated with a data source that an ODBC application program is written to use. Think of an ODBC data source as a door with a name on it. Each door will lead you to a particular database. For example, the data source named Sales Figures might be a SQL Server database, whereas the Accounts Payable data source could refer to an Access database. The physical database referred to by a data source can reside anywhere on the LAN. The ODBC system files are not installed on your system by Windows 95. Rather, they are installed when you setup a separate database application, such as SQL Server Client or Visual Basic 4.0. When the ODBC icon is installed in Control Panel, it uses a file called ODBCINST.DLL. It is also possible to administer your ODBC data sources through a stand-alone program called ODBCADM.EXE.

From a programming perspective, the beauty of ODBC is that the application can be written to use the same set of function calls to interface with any data source, regardless of the database vendor. The source code of the application doesn't change whether it talks to Oracle or SQL Server. We only mention these two as an example. There are ODBC drivers available for several dozen popular database systems. Even Excel spreadsheets and plain text files can be turned into data sources. The operating system uses the Registry information written by ODBC Administrator to determine which low-level ODBC drivers are needed to talk to the data source (such as the interface to Oracle or SQL Server). The loading of the ODBC drivers is transparent to the ODBC application program. In a client/server environment, the ODBC API even handles many of the network issues for the application programmer.

The advantages of this scheme are so numerous that you are probably thinking there must be some catch. The only disadvantage of ODBC is that it isn't as efficient as talking directly to the native database interface. ODBC has had many detractors make the charge that it is too slow.

Microsoft has always claimed that the critical factor in performance is the quality of the driver software that is used. In our humble opinion, this is true. The availability of good ODBC drivers has improved a great deal recently. And anyway, the criticism about performance is somewhat analogous to those who said that compilers would never match the speed of pure assembly language. Maybe not, but the compiler (or ODBC) gives you the opportunity to write cleaner programs, which means you finish sooner. Meanwhile, computers get faster every year.

6.3 JDBC

In an effort to set an independent database standard API for Java; Sun Microsystems developed Java Database Connectivity, or JDBC. JDBC offers a generic SQL database access mechanism that provides a consistent interface to a variety of RDBMSs. This consistent interface is achieved through the use of "plug-in" database connectivity modules, or drivers. If a database vendor wishes to have JDBC support, he or she must provide the driver for each platform that the database and Java run on.

To gain a wider acceptance of JDBC, Sun based JDBC's framework on ODBC. As you discovered earlier in this chapter, ODBC has widespread support on a variety of platforms. Basing JDBC on ODBC will allow vendors to bring JDBC drivers to market much faster than developing a completely new connectivity solution.

JDBC was announced in March of 1996. It was released for a 90-day public review that ended June 8, 1996. Because of user input, the final JDBC v1.0 specification was released soon after. The remainder of this section will cover enough information about JDBC for you to know what it is about and how to use it effectively. This is by no means a complete overview of JDBC.

That would fill an entire book.

JDBC Goals

Few software packages are designed without goals in mind. JDBC is one that, because of its many goals, drove the development of the API. These goals, in conjunction with early reviewer feedback, have finalized the JDBC class library into a solid framework for building database applications in Java.

The seven design goals for JDBC are as follows:

1. SQL Level API

The designers felt that their main goal was to define a SQL interface for Java. Although not the lowest database interface level possible, it is at a low enough level for higher-level tools and APIs to be created. Conversely, it is at a high enough level for application programmers to use it confidently. Attaining this goal allows for future tool vendors to "generate" JDBC code and to hide many of JDBC's complexities from the end user.

2. SQL Conformance

SQL syntax varies as you move from database vendor to database vendor. In an effort to support a wide variety of vendors, JDBC will allow any query statement to be passed through it to the underlying database driver. This allows the connectivity module to handle non-standard functionality in a manner that is suitable for its users.

3. JDBC must be implemental on top of common database interfaces

The JDBC SQL API must "sit" on top of other common SQL level APIs. This goal allows JDBC to use existing ODBC level drivers by the use of a software interface. This interface would translate JDBC calls to ODBC and vice versa.

4. Provide a Java interface that is consistent with the rest of the Java system

Because of Java's acceptance in the user community thus far, the designers feel that they should not stray from the current design of the core Java system.

5. Keep it simple

This goal probably appears in all software design goal listings. JDBC is no exception. Sun felt that the design of JDBC should be very simple, allowing for only one method of completing a task per mechanism. Allowing duplicate functionality only serves to confuse the users of the API.

6. Use strong, static typing wherever possible

Strong typing allows for more error checking to be done at compile time; also, less error appear at runtime.

7. Keep the common cases simple

Because more often than not, the usual SQL calls used by the programmer are simple SELECT's, INSERT's, DELETE's and UPDATE's, these queries should be simple to perform with JDBC. However, more complex SQL statements should also be possible.

Finally, we decided to proceed the implementation using Java Networking. And for dynamically updating the cache table we go for MSAccess database.

Java has two things:

- A programming language and a platform.

- Java is a high-level programming language that is all of the following

• Simple	• Architecture-neutral
• Object-oriented	• Portable
• Distributed	• High-performance
• Interpreted	• multithreaded
• Robust	• Dynamic
• Secure	

Java is also unusual in that each Java program is both compiled and interpreted. With a compile you translate a Java program into an intermediate language called Java byte codes the platform-independent code instruction is passed and run on the computer. Compilation happens just once; interpretation occurs each time the program is executed. The figure illustrates how this works.

JavaProgram   Interpreter

Compilers   My Program
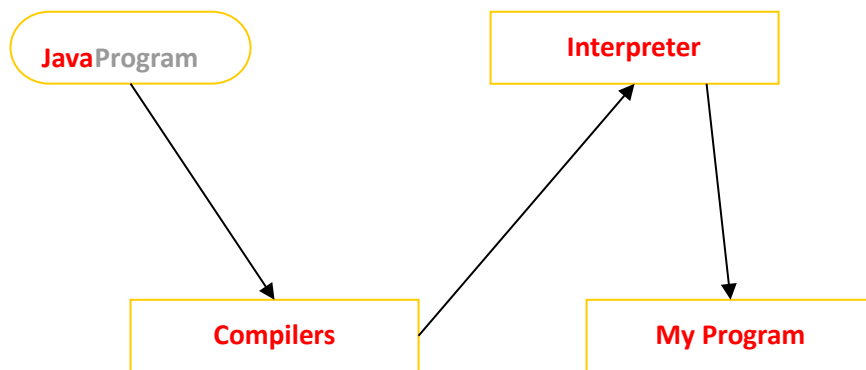
Fig: 6.3.1 Java overview

6.4 NETWORK CONCEPTS

TCP/IP stack

The TCP/IP stack is shorter than the OSI one:
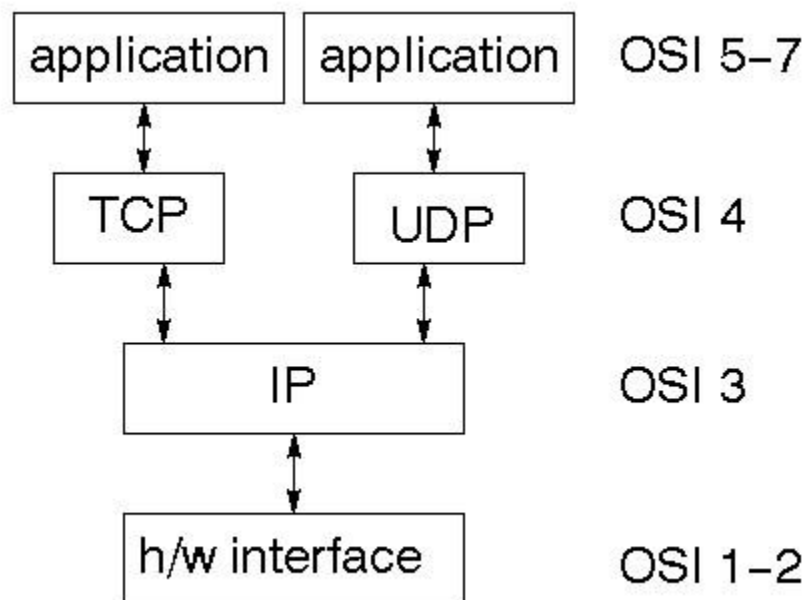
TCP is a connection-oriented protocol; UDP (User Datagram Protocol) is a connectionless protocol.

IP datagram's

The IP layer provides a connectionless and unreliable delivery system. It considers each datagram independently of the others. Any association between datagram must be supplied by the higher layers. The IP layer supplies a checksum that includes its own header. The header includes the source and destination addresses. The IP layer handles routing through an Internet. It is also responsible for breaking up large datagram into smaller ones for transmission and reassembling them at the other end.

UDP

UDP is also connectionless and unreliable. What it adds to IP is a checksum for the contents of the datagram and port numbers. These are used to give a client/server model - see later.

TCP

TCP supplies logic to give a reliable connection-oriented protocol above IP. It provides a virtual circuit that two processes can use to communicate.

Internet addresses

In order to use a service, you must be able to find it. The Internet uses an address scheme for machines so that they can be located. The address is a 32-bit integer which gives the IP address. This encodes a network ID and more addressing. The network ID falls into various classes according to the size of the network address.

Network address

Class A uses 8 bits for the network address with 24 bits left over for other addressing. Class B uses 16-bit network addressing. Class C uses 24-bit network addressing and class D uses all 32.

Subnet address

Internally, the UNIX network is divided into sub networks. Building 11 is currently on one sub network and uses 10-bit addressing, allowing 1024 different hosts.

Host Address

8 bits are finally used for host addresses within our subnet. This places a limit of 256 machines that can be on the subnet.

Total address

137.92.11.13

network   subnet   host
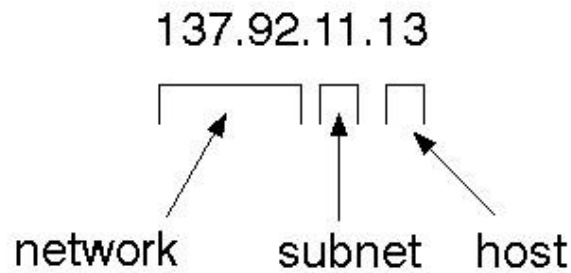
The 32-bit address is usually written as 4 integers separated by dots

Port addresses

A service exists on a host, and is identified by its port. This is a 16 bit number. To send a message to a server, you send it to the port for that service of the host that it is running on. This is not location transparency! Certain of these ports are "well known".

Sockets

A socket is a data structure maintained by the system to handle network connections. A socket is created using the call socket. It returns an integer that is like a file descriptor. In fact, under Windows, this handle can be used with Read File and Write File functions.

```
#include <sys/types.h> #include
<sys/socket.h>
int socket (int family, int type, int protocol);
```

Here "family" will be AF_INET for IP communications, protocol will be zero, and type will depend on whether TCP or UDP is used. Two processes wishing to communicate over a network create a socket each. These are similar to two ends of a pipe - but the actual pipe does not yet exist.

6.5 J2ME

Sun Microsystems defines J2ME as "a highly optimized Java run-time environment targeting a wide range of consumer products, including pagers, cellular phones, screen-phones, digital set top boxes and car navigation systems." Announced in June 1999 at the JavaOne Developer Conference, J2ME brings the cross-platform functionality of the Java language to smaller devices, allowing mobile wireless devices to share applications. With J2ME, Sun has adapted the Java platform for consumer products that incorporate or are based on small computing devices.

1. General J2ME architecture
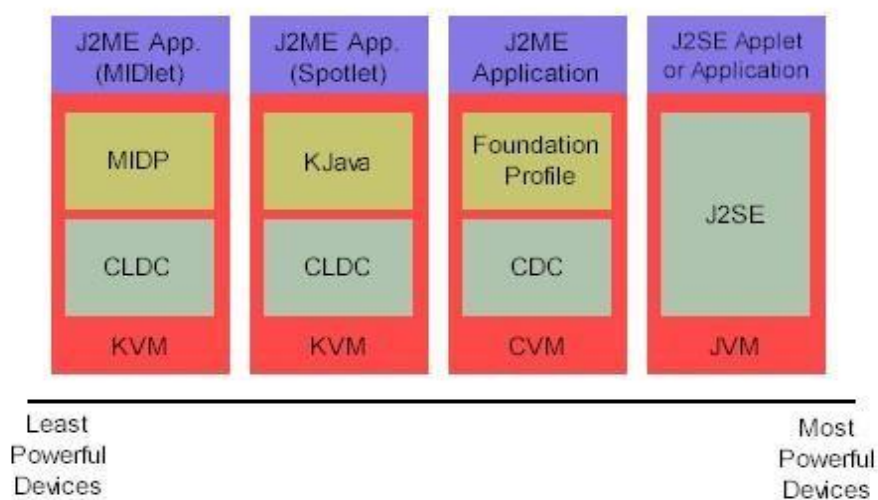
Fig: 6.5.1 J2ME Architecture

J2ME uses configurations and profiles to customize the Java Runtime Environment (JRE). As a complete JRE, J2ME is comprised of a configuration, which determines the JVM used, and a profile, which defines the application by adding domain-specific classes. The configuration defines the basic run-time environment as a set of core classes and a specific JVM that run on specific types of devices. We'll discuss configurations in detail in the profile defines the application; specifically, it adds domain-specific classes to the J2ME configuration to define certain uses for devices. We'll cover profiles in depth in the following graphic depicts the relationship between the different virtual machines, configurations, and profiles.

2. Developing J2ME applications

Introduction In this section, we will go over some considerations you need to keep in mind when developing applications for smaller devices. We'll take a look at the way the compiler is invoked when using J2SE to compile J2ME applications. Finally, we'll explore packaging and deployment and the role pre-verification plays in this process.

3. Design considerations for small devices

Developing applications for small devices requires you to keep certain strategies in mind during the design phase. It is best to strategically design an application for a small device before you begin coding. Correcting the code because you failed to consider all of the "gotchas" before developing the application can be a painful process. Here are some design strategies to consider:

Smaller is better. This consideration should be a "no brainer" for all developers. Smaller applications use less memory on the device and require shorter installation times. Consider packaging your Java applications as compressed Java Archive (jar) files.

Minimize run-time memory use. To minimize the amount of memory used at run time, use scalar types in place of object types. Also, do not depend on the garbage collector.

4. Configuration overview

The configuration defines the basic run-time environment as a set of core classes and a specific JVM that run on specific types of devices. Currently, two configurations exist for J2ME, though others may be defined in the future:

Connected Limited Device Configuration (CLDC) is used specifically with the KVM for 16bit or 32-bit devices with limited amounts of memory. This is the configuration (and the virtual machine) used for developing small J2ME applications. Its size limitations make CLDC more interesting and challenging (from a development point of view) than CDC. CLDC is also the configuration that we will use for developing our drawing tool application. An example of a small wireless device running small applications is a Palm hand-held computer.

Connected Device Configuration (CDC) is used with the C virtual machine (CVM) and is used for 32-bit architectures requiring more than 2 MB of memory. An example of such a device is a Net TV box.

5. J2ME profiles

What is a J2ME profile?

As we mentioned earlier in this tutorial, a profile defines the type of device supported. The Mobile Information Device Profile (MIDP), for example, defines classes for cellular phones. It adds domain-specific classes to the J2ME configuration to define uses for similar devices. Two profiles have been defined for J2ME and are built upon CLDC: KJava and MIDP. Both Java and MIDP are associated with CLDC and smaller devices. Profiles are built on top of configurations. Because profiles are specific to the size of the device (amount of memory) on which an application runs, certain profiles are associated with certain configurations.

6.6 TOMCAT SERVER

Tomcat is an open-source web server developed by Apache Group. Apache Tomcat is the servlet container that is used in the official Reference Implementation for the Java Servlet and Java Server Pages technologies. The Java Servlet and Java Server Pages specifications are developed by Sun under the Java Community Process. Web Servers like Apache Tomcat support only web components while an application server supports web components as well as business components (BEAs WebLogic, is one of the popular application servers).
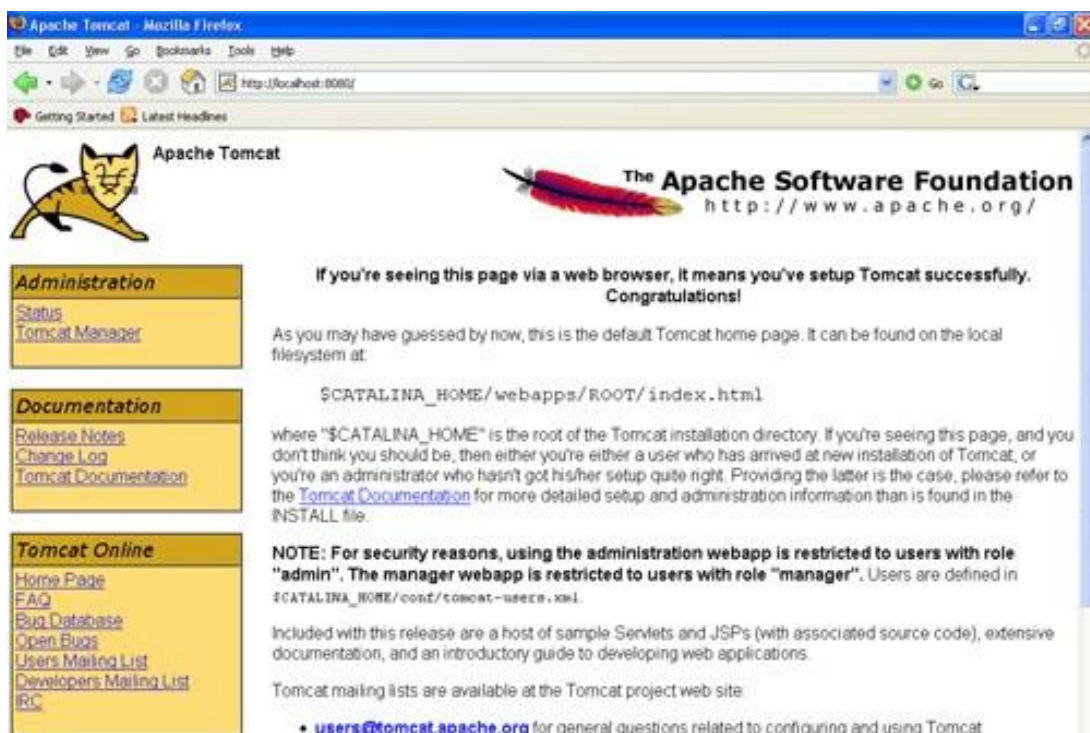
Fig: 6.6.1 Apache Tomcat 7.0

Tomcat's architecture is built upon a modular design, allowing it to function as a standalone server or be integrated seamlessly with other web servers like Apache HTTP Server. Its core components include the Catalina servlet container, which handles the execution of Java Servlets and JSP pages, and the Coyote connector, responsible for managing communication between Tomcat and external clients using various protocols such as HTTP.

Key Features:

Servlet and JSP Support:

Tomcat provides full support for Java Servlets and Java Server Pages, enabling developers to build dynamic and interactive web applications.

Open-Source Nature:

Being open-source, Tomcat is freely available, fostering a vibrant community of developers who contribute to its improvement and ensure its continual evolution.

Platform Independence:

Tomcat is platform-independent, making it compatible with various operating systems, including Windows, Linux, and macOS.

Scalability:

Tomcat offers scalability, allowing it to efficiently handle a growing number of requests by adding more resources or by clustering multiple Tomcat instances.

Security Features:

Security is a top priority for Tomcat, with features such as SSL/TLS support, authentication, and authorization mechanisms, safeguarding web applications from potential threats.

Management and Monitoring:

Tomcat includes a web-based management interface, the Tomcat Manager, facilitating easy deployment, un-deployment, and monitoring of applications. Additionally, tools like JConsole and JVisualVM can be employed for in-depth performance analysis.

Deploying applications on Tomcat is a straightforward process. Developers can package their applications as WAR (Web Application Archive) files, which can then be deployed to Tomcat using the Tomcat Manager or by placing the WAR files directly into the designated deployment directory.

Apache Tomcat continues to be a stalwart choice for developers seeking a reliable, scalable, and open-source solution for hosting Java web applications.

7. TESTING AND VALIDATION

7.1 Testing

TESTING METHODOLOGIES

The following are the Testing Methodologies:

- o **Unit Testing.**
- o **Integration Testing.**
- o **User Acceptance Testing.**
- o **Output Testing.**
- o **Validation Testing.**

Unit Testing

Unit testing focuses verification effort on the smallest unit of Software design that is the module. Unit testing exercises specific paths in a module's control structure to

ensure complete coverage and maximum error detection. This test focuses on each module individually, ensuring that it functions properly as a unit. Hence, the naming is Unit Testing.

During this testing, each module is tested individually and the module interfaces are verified for the consistency with design specification. All-important processing path are tested for the expected results. All error handling paths are also tested.

Integration Testing

Integration testing addresses the issues associated with the dual problems of verification and program construction. After the software has been integrated a set of high order tests are conducted. The main objective in this testing process is to take unit tested modules and builds a program structure that has been dictated by design.

The following are the types of Integration Testing:

1. Top-Down Integration

This method is an incremental approach to the construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main program module. The module subordinates to the main program module are incorporated into the structure in either a depth first or breadth first manner.

In this method, the software is tested from main module and individual stubs are replaced when the test proceeds downwards.

2. Bottom-up Integration

This method begins the construction and testing with the modules at the lowest level in the program structure. Since the modules are integrated from the bottom up, processing required for modules subordinate to a given level is always available and the need for stubs is eliminated. The bottom-up integration strategy may be implemented with the following steps:

- The low-level modules are combined into clusters into clusters that perform a specific Software sub-function.
- A driver (i.e.) the control program for testing is written to coordinate test   case input and output.
- The cluster is tested.
- Drivers are removed and clusters are combined moving upward in the program structure

The bottom-up approaches test each module individually and then each module is module is integrated with a main module and tested for functionality.

7.1.3 User Acceptance Testing

User Acceptance of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes wherever required. The system developed provides a friendly user interface that can easily be understood even by a person who is new to the system.

7.1.4 Output Testing

After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in the specified format. Asking the users about the format required by them tests the outputs generated or displayed by the system under consideration. Hence the output format is considered in 2 ways – one is on screen and another in printed format.

7.1.5 Validation Checking

Validation checks are performed on the following fields.

Text Field:

The text field can contain only the number of characters lesser than or equal to its size. The text fields are alphanumeric in some tables and alphabetic in other tables. Incorrect entry always flashes and error message.

Numeric Field:

The numeric field can contain only numbers from 0 to 9. An entry of any character flashes an error message. The individual modules are checked for accuracy and what it has to perform. Each module is subjected to test run along with sample data. The individually tested modules are integrated into a single system. Testing involves executing the real data information is used in the program the existence of any program defect is inferred from the output. The testing should be planned so that all the requirements are individually tested.

A successful test is one that gives out the defects for the inappropriate data and produces and output revealing the errors in the system.

Preparation of Test Data

Taking various kinds of test data does the above testing. Preparation of test data plays a vital role in the system testing. After preparing the test data the system under study is tested using that test data. While testing the system by using test data errors are again uncovered and corrected by using above testing steps and corrections are also noted for future use.

Using Live Test Data:

Live test data are those that are actually extracted from organization files. After a system is partially constructed, programmers or analysts often ask users to key in a set of data from their normal activities. Then, the systems person uses this data as a way to partially test the system. In other instances, programmers or analysts extract a set of live data from the files and have them entered themselves.

It is difficult to obtain live data in sufficient amounts to conduct extensive testing. And, although it is realistic data that will show how the system will perform for the typical processing requirement, assuming that the live data entered are in fact typical, such data generally will not test all combinations or formats that can enter the system. This bias toward typical values then does not provide a true systems test and in fact ignores the cases most likely to cause system failure.

Using Artificial Test Data:

Artificial test data are created solely for test purposes, since they can be generated to test all combinations of formats and values. In other words, the artificial data, which can quickly be prepared by a data generating utility program in the information systems department, make possible the testing of all login and control paths through the program.

The most effective test programs use artificial test data generated by persons other than those who wrote the programs. Often, an independent team of testers formulates a testing plan, using the systems specifications.

The package "Virtual Private Network" has satisfied all the requirements specified as per software requirement specification and was accepted.

7.2 USER TRAINING

Whenever a new system is developed, user training is required to educate them about the working of the system so that it can be put to efficient use by those for whom the system has been primarily designed. For this purpose, the normal working of the project was demonstrated to the prospective users. Its working is easily understandable and since the expected users are people who have good knowledge of computers, the use of this system is very easy.

7.3 MAINTAINENCE

This covers a wide range of activities including correcting code and design errors. To reduce the need for maintenance in the long run, we have more accurately defined the user's requirements during the process of system development. Depending on the requirements, this system has been developed to satisfy the needs to the largest possible extent. With development in technology, it may be possible to add many more features based on the requirements in future. The coding and designing is simple and easy to understand which will make maintenance easier.

TESTING STRATEGY:

A strategy for system testing integrates system test cases and design techniques into a well-planned series of steps that results in the successful construction of software. The testing strategy must co-operate test planning, test case design, test execution, and the resultant data collection and evaluation. A strategy for software testing must accommodate low-level tests that are necessary to verify that a small source code segment has been correctly   implemented   as well as high level tests that validate   major system functions against user requirements.

Software testing is a critical element of software quality assurance and represents the ultimate review of specification design and coding. Testing represents an interesting anomaly for the software. Thus, a series of testing   are performed for the proposed system before the system is ready for user acceptance testing.

SYSTEM TESTING:

Software once validated must be combined with other system elements (e.g. Hardware, people, database). System testing verifies that all the elements are proper and that overall system function performance is

achieved. It also tests to find discrepancies between the system and its original objective, current specifications and system documentation.

UNIT TESTING:

In unit testing different are modules are tested against the specifications produced during the design for the modules. Unit testing is essential for verification of the code produced during the coding phase, and hence the goals to test the internal logic of the modules. Using the detailed design description as a guide, important Conrail paths are tested to uncover errors within the boundary of the modules. This testing is carried out during the programming stage itself. In this type of testing step, each module was found to be working satisfactorily as regards to the expected output from the module.

In Due Course, latest technology advancements will be taken into consideration. As part of technical build-up many components of the networking system will be generic in nature so that future projects can either use or interact with this. The future holds a lot to offer to the development and refinement of this project.
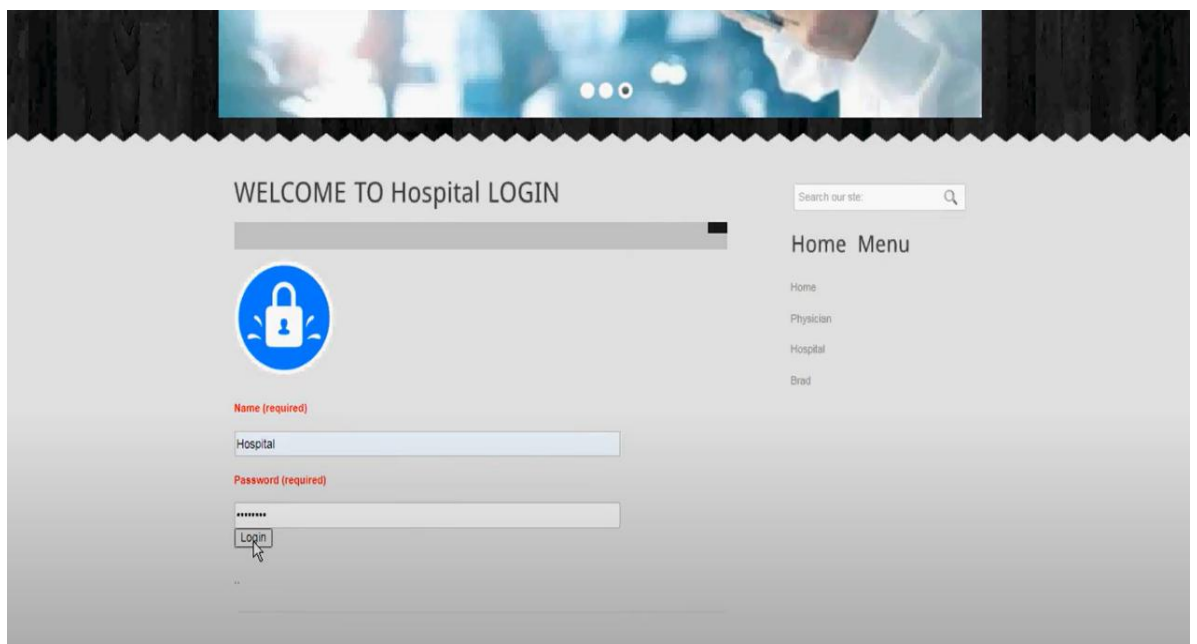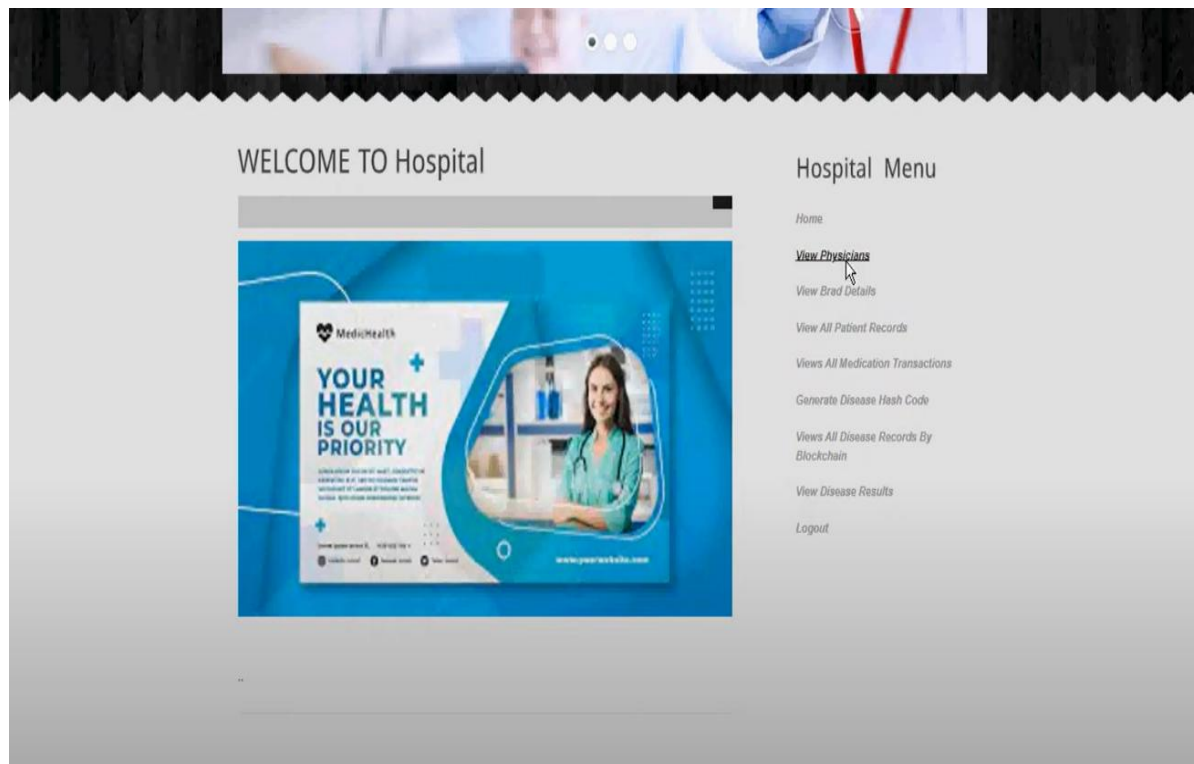
8. OUTPUT SCREENS

WELCOME TO Hospital LOGIN

Search our site:

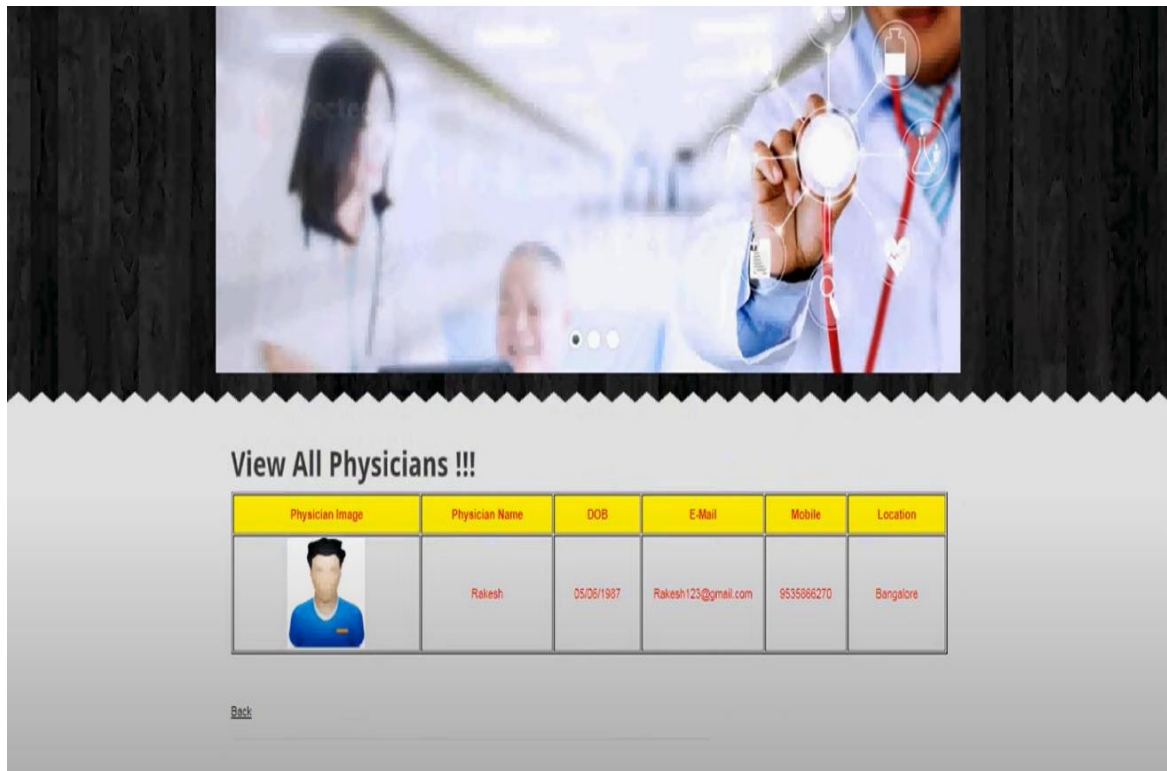## Home Menu

Home

Physician

Hospital

Brad

Name (required)

Hospital

Password (required)

Login

FIGURE 8.1 HOME PAGE LOGIN

FIGURE 8.2 HOSPITAL MENU

FIGURE 8.3 VIEW ALL PHYSICIANS
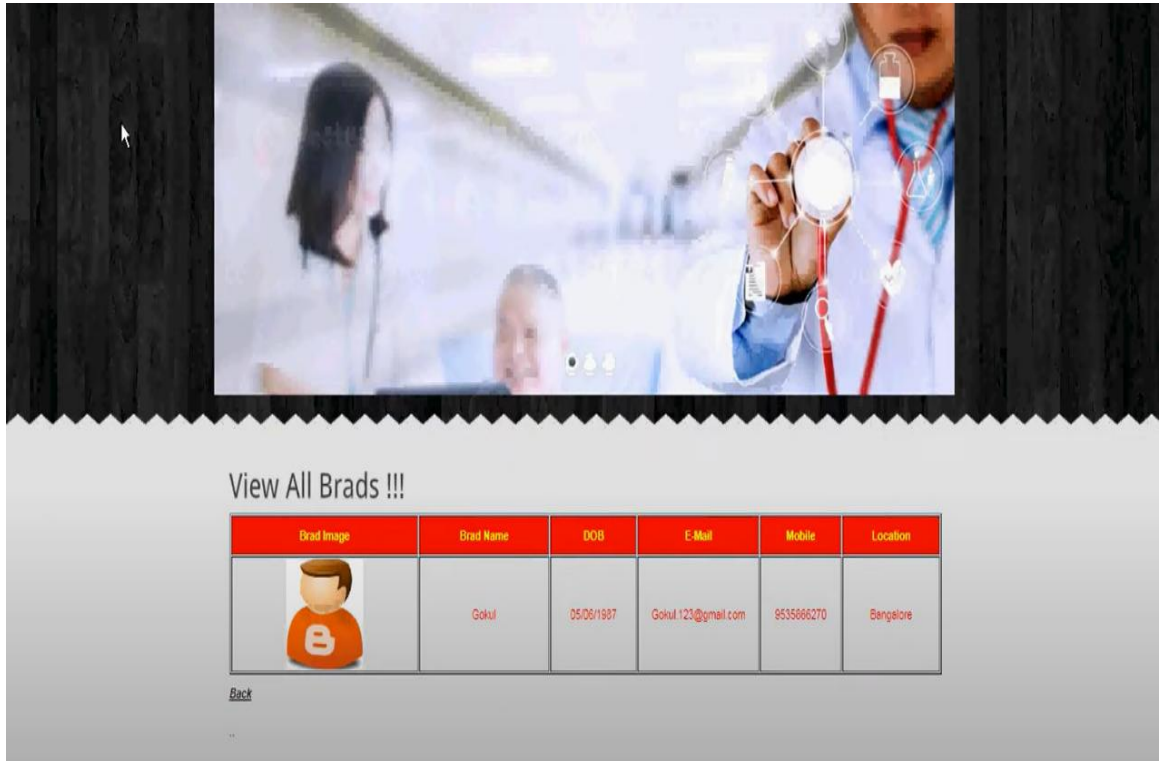
FIGURE 8.4 VIEW ALL BRADS DETAILS
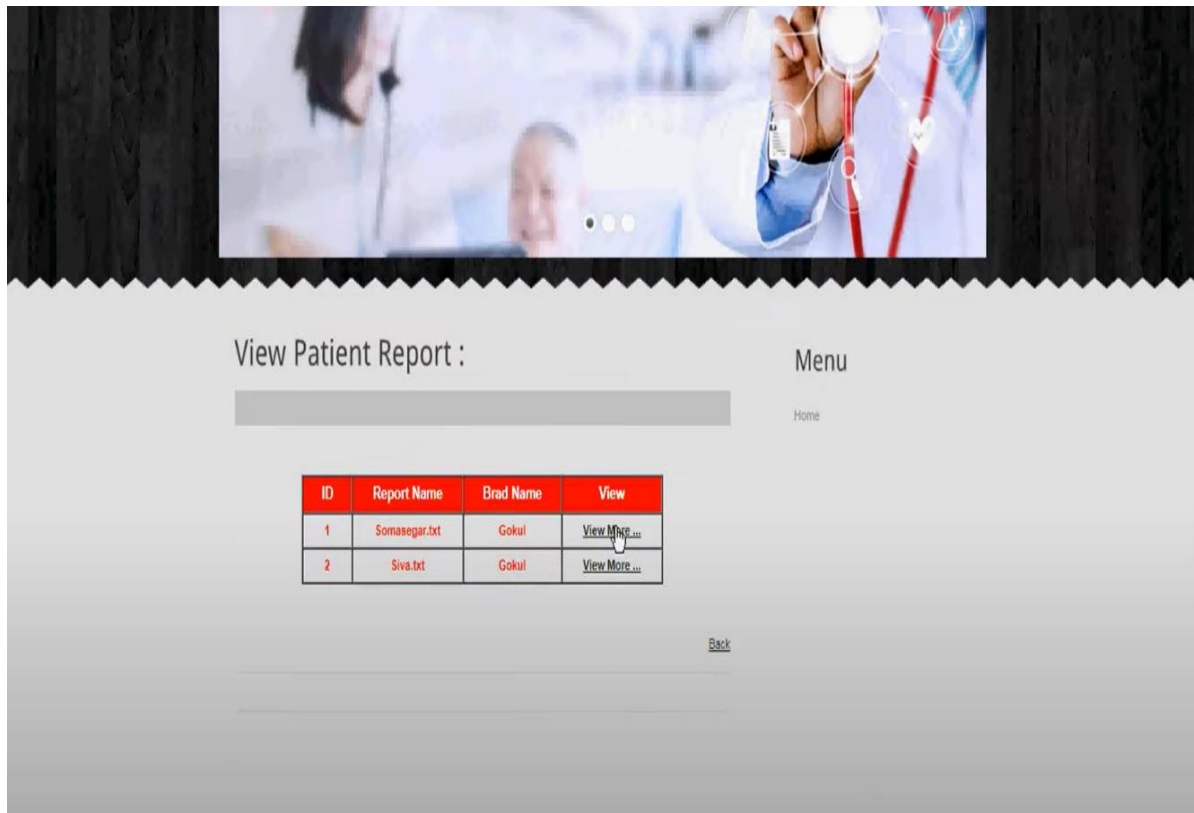
FIGURE 8.5 VIEW PATIENT RECORDS

FIGURE 8.6

FIGURE 8.7 RECORDS

FIGURE 8.8 PATIENT LOGIN

FIGURE 8.9 SENSOR DATA AND UPLOAD PATIENT

9. CONCLUSION

At present, many developments are going on in order to subside the uncertain health mishaps. Artificial Intelligence, Big data, and many more techniques are being used without any due consideration of how this vast and diverse data can be accumulated from the real world conveniently and store them securely. The digital twin technology can enable an effective way for collecting data and generating insight through analysis. But this data, being generated through numerous processes, needs to be systematically stored with proper security and handled by a compact system, which can also render all the requirements to create a digital twin in the healthcare sector. With these motivations in mind, our article presents a concrete mathematical model of Digital Twin for healthcare, proposes the Healthcare Digital Twin (*HDT*) system and provides the protocol flow for the system to coincide with the mathematical model.

The main contributions of this article are the following. The *HDT* is proposed with the incentive of remedying the segregated data collection process by incorporating a defined mathematical data model with which patient relevant data can be collected in a regulated way. The model has emphasized three core stages: Pre-Hospital Admit, Patient Disease Diagnose, and Surgical Operative Procedure, as these stages present the three most important stages for a patient. Next, the architecture of the system, being integrated with block chain, is constructed with the defined data model in consideration, so that users can use the data for other purposes without any conflicts. With proper protocol flows, there are some illustrations of how the system can be used for different use cases.

It is understandable that, even with the state-of-the-art technologies, a digital twin of a full patient body is still out of reach because of the extant nuances in the human body. There is a raft of opportunities to decrease this gap. We strongly believe that the proposed model and system in this article will be a step towards fulfilling this goal. In future, we will develop the proposed system and examine its applicability and performance.

REFERENCES

1. *Primary Health Care, World Health Organization*. Accessed:Apr. 8, 2022. [Online]. Available: https://www.who.int/news-room/factsheets/detail/primary-health-care

2. F. Alshehri and G. Muhammad, ``A comprehensive survey of the Internet of Things (IoT) and AI-based smart healthcare,'' *IEEE Access*, vol. 9, pp. 3660_3678, 2021.

3. A. N. Navaz, M. A. Serhani, H. T. El Kassabi, N. Al-Qirim, and H. Ismail,``Trends, technologies, and key challenges in smart and connected healthcare,''*IEEE Access*, vol. 9, pp. 74044_74067, 2021.

4. U. Bharti, D. Bajaj, H. Batra, S. Lalit, S. Lalit, and A. Gangwani, ``Medbot:Conversationalarti_cial intelligence powered chatbot for delivering telehealth

5. M. A. Bazel, F. Mohammed, and M. Ahmed, ``Blockchain technology in healthcare big data management: Bene_ts, applications and challenges,'' in *Proc. 1st Int. Conf. Emerg. Smart Technol. Appl. (eSmarTA)*, Aug. 2021, pp. 1_8.

6. J. Stauffer and Q. Zhang, ``S2Cloud: A novel cloud system for mobile health big data management,'' in *Proc. IEEE Int. Conf. Internet Things (iThings) IEEE Green Comput.; Commun. (GreenCom) IEEE Cyber, Phys. Social Comput. (CPSCom) IEEE Smart Data (SmartData) IEEE Congr. Cybermatics (Cybermatics)*, Dec. 2021, pp. 380_383.

7. T. L. Rodziewicz, B. Houseman, and J. E. Hipskind, *Medical Error Reduction and Prevention*. Tampa, FL, USA: StatPearls Publishing, 2021.

8. Z. Liu, N. Meyendorf, and N. Mrad, ``The role of data fusion in predictive maintenance using digital twin,'' *AIP Conf. Proc.*, vol. 1949, no. 1, 2018, Art. no. 020023.

9. R. Martinez-Velazquez, R. Gamez, and A. El Saddik, ``Cardio twin: A digital twin of the human heart running on the edge,'' in *Proc. IEEE Int. Symp. Med. Meas. Appl. (MeMeA)*, Jun. 2019, pp. 1_6.

10. P. Barbiero, R. V. Torné, and P. Lió, ``Graph representation forecasting of patient's medical conditions: Towards a digital twin,'' 2020,