**OBJECTIVE AND SCOPE OF THE PROJECT**

Bhubaneswar tops the list of smart cities; thus, no stone should be left unturned to introduce smartness in our lives. It is the authentic information that helps us is the real world to stay updated So, as we run towards smartness why should we lack behind compromising the health facilities available in our city. Giving a boost to the smart concept, we have come up with an initiative to host a wide array of services under a single platform. The web application "Smart city" is a handy application for the citizens of smart city. such as

* It focuses on providing health related information and services at a click. Information regarding path lab and blood bank Hospitals, Burn Units, Trauma Care Centers, Ambulance Vans, Mortuary vans.
* Host of upcoming health camps and events can also be viewed from the application.
* Users can book Ambulance/Mortuary Van, Blood donation dates, etc
* All private/government Organisations can register themselves in the application to provide their services such as
* Institutions
* Blood banks
* Hospitals
* Vehicles
* Ambulances
* Path labs
* Hospitals

And users can view this important information and use the services accordingly..

* Tourist can get all information regarding different sight seeing places .
* Users can choose and book hotels according to their requirements.
* It is cost effective and saves time
* Booking 24/7, anywhere
* It's discreet
* Advance Booking
* Occupancy of the Room
* Departure of the Guest
* Billing

Most beneficial function of this application is people can get a comparative statement based on distance and price regarding the services. This is developed in java and the database used here is MY SQL SERVER. The new system developed includes the provision for future expansion. Now a day’s computerization is spreading with great speed. Many organizations are being computerized and are surely enjoying the benefits of computerization. It provides one click reliable and economical solution to the vast city where time is main concern for each individual. “Smart city ", web based application makes the reach of people from the remotest area easy and convenient.

**THEORETICAL BACKGROUND**

## Overview of Java Technology

Java, whether you love it, or hate it, it's here to stay. Like everyone's favorite language C, Java has had a major impact on the computing scene. When the history of computers is written, its name will be up there with the stars.

If you were to choose just one language to learn today, it should be Java. It's being pushed aggressively by Sun and is growing by leaps and bounds. There are lots of Java programmers out there and more join the party every day.

Java started out as a bit of an accident. A team under Bill Joy was working at Sun on a new programming language for embedded applications. Java was originally expected to work in toasters and fridges, not on modern computers! The initial prognosis for Java was not good and it was only the rise of the Internet which saved Java from oblivion. Since then, neither the Net nor Sun nor Java has looked back and all have grown from strength to strength.

World Wide Web is an open ended information retrieval system designed to be used in the distributed environment. This system contains web pages that provide both information and controls. We can navigate to a new web page in any direction. This is made possible worth HTML java was meant to be used in distributed environment such as internet. So java could be easily incorporated into the web system and is capable of supporting animation graphics, games and other special effect. The web has become more dynamic and interactive with support of java. We can run a java program on remote machine over internet with the support of web.

**Java Environment**

Java environment includes a large no. of tools which are part of the system known as java development kit (JDK) and hundreds of classes, methods, and interfaces grouped into packages forms part of java standard library (JSL).

### 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 are then interpreted on each platform by the runtime environment. Java also provides stringent compile and runtime checking and automatic memory management in order to ensure solid code.

**Java Virtual Machine**

When we compile the code, java compiler creates machine code (byte code) for a hypothetical machine called java virtual machine (jvm). The jvm will execute the byte code and overcomes the issue of portability. The code is written and compile for one machine and interpreted all other machines. This machine is called java virtual machine.

**Paradigm of Java**

* Dynamic down loading applets(small application programs);
* Elimination of flatware phenomenon that is providing those features of a product that user needs at a time. The remaining features of a product can remain in the server.
* Changing economic model of the software
* Up-to-date software availability
* Supports network entire computing
* Supports CORBA & DCOM

## An Overview of JSP

### The Java Server Pages Technology

Java Server Pages™ technology is the Java™ technology in the J2EE platform for building applications containing dynamic Web content such as HTML, DHTML, XHTML and XML. The Java Server Pages technology enables the authoring of Web pages that create dynamic content easily but with maximum power and flexibility. The Java Server Pages technology provides a textual description for the creation of a response from a request. The technology builds on the following concepts:

**Template Data**

Substantial portions of dynamic content are actually fixed. The JSP technology allow for the natural manipulation of this data.

**Addition of Dynamic Data**

The JSP technology allows the addition of dynamic data to the template data in a way that is simple yet powerful.

**Encapsulation of Functionality**

The JSP technology provides two related mechanisms for the encapsulation of functionality: the standard Java Beans component architecture and the tag library mechanism.

**Good Tool Support**

The JSP technology has features that enable the creation of good authoring tools. The result is a flexible and powerful server-side technology.

**Benefits of the Java Server Pages Technology**

The Java Server Pages technology offers a number of benefits:

Write Once, Run Anywhere properties

The Java Server Pages technology is platform independent, both in its dynamic Web pages, Web servers, and its underlying server components. You can author JSP pages on any platform, run them on any Web server or Web enabled application server, and access them from any Web browser.

**High quality tool support**

The Write Once, Run Anywhere properties of JSP allows the user to choose best-of-breed tools. Additionally, an explicit goal of the Java Server Pages design is to enable the creation of high quality portable tools.

**Separation of Roles**

JSP supports the separation of roles: developers write components that interact with server-side objects.

**Reuse of components and tag libraries**

The Java Server Pages technology emphasizes the use of reusable components such as Java Beans™ components, Enterprise Java Beans™ components and tag libraries.

**Separation of dynamic and static content**

The Java Server Pages technology enables the separation of static content from dynamic content that is inserted into the static template.

**Support for scripting and actions**

The Java Server Pages technology supports scripting elements as well as actions. Actions permit the encapsulation of useful functionality in a convenient form that can also be manipulated by tools; scripts provide a mechanism to glue together this functionality in a per-page manner.

**Web access layer for N-tier enterprise application architecture(s)**

The Java Server Pages technology is an integral part of the Java 2 Platform Enterprise Edition (J2EE), which brings Java technology to enterprise computing.

### 5.2.2 About HTML

HTML (hypertext markup language) is a language used to create hypertext documents that have hyperlinks embedded in them. It consists of tags embedded in the text of a document with HTML. We can build web pages or web document s. it is basically a formatting language and not a programming language. The browser reading the document interprets markup tags to help format the document for subsequent display to a reader. HTML is a language for describing structured documents. HTML is a platform independent. WWW (World Wide Web) pages are written using HTML. HTML tags control in part the representation of the WWW page when view with web browser. The browser interprets HTML tags in the web document and displays it. Different browsers show data differently.

**Examples of browsers used to be web pages include:**

* Netscape
* Internet Explorer

**About JAVA Script**

Java script is a general purpose prototype based objects oriented scripting language developed jointly by Sun and Netscape and is meant for the WWW. It is designed to be embedded in diverse applications and systems, without consuming much memory. Java script borrows most of its syntax from java but also inherits from awk and Perl, with some indirect influence from self in its object prototype system.

Java scripts dynamically typed that is programs do not declare variable types, and the type of variable is unrestricted and can change at runtime. Source can be generated at run time and evaluated against an arbitrary scope. Typical implementations compile by translating source into a specified byte code format, to check syntax and source consistency. Note that the availability to generate and interpreted programs at runtime implies the presence of a compiler at runtime.

Java script is a high level scripting language that does not depend on or expose particular machine representations or operating system services. It provides automatic storage management, typically using a garbage collector.

**FEATURES:**

* Java script is embedded into HTML documents and is executed with in them.
* Java script is browser dependent
* JavaScript is an interpreted language that can be interpreted by the browser at run time.
* Java script is loosely typed language
* Java script is an object based language.
* Java script is an Event-Driven language and supports event handlers to specify the functionality of a button.

**ADVANTAGES**

* 1. Java script can be used for client side application
  2. Java script provides means to contain multiframe windows for presentation of the web.
  3. Java script provides basic data validation before it is sent to the server. Eg : login and password checking or whether the values

entered are correct or whether all fields in a from are filled and reduced network traffic

* 1. It creates interactive forms and client side lookup tables.

**JavaDataBaseConnectivity (JDBC)**

**Overview of New Features**

**Result set enhancements**

The JDBC 1.0 API provided result sets that had the ability to scroll in a forward direc-tiononally. Scrollable result sets allow for more flexibility in the processing of results by providing both forward and backward movement through their contents. In addition, scrollable result sets allow for relative and absolute positioning. For example, it’s possible to move to the fourth row in a scrollable result set directly, or to move directly to the third row following the current row, provided the row exists. The JDBC API allows result sets to be directly updatable, as well.

**Batch updates**

The batch update feature allows an application to submit multiple update statements(insert/update/delete) in a single request to the database. This can provide a dramatic increase in performance when a large number of update statements need to be executed.

**Advanced data types**

Increased support for storing persistent Java programming language objects (Java ob-jects)and a mapping for SQL99 data types such as binary large objects, and structured types, have been added to the JDBC API. An application may also customize the mapping of SQL99 structured types into Java programming language classes.

**Row sets**

As its name implies, a rowset encapsulates a set of rows. A rowset may or may not maintain an open database connection. When a rowset is ‘disconnected’ from its data source, updates performed on the rowset are propagated to the underlying database using an optimistic concurrency control algorithm.Rowsets add support to the JDBC API for the JavaBeans component model. A rowset object is a bean. A rowset implementation may be serializable. Rowsets can be created at design time and used in conjunction with other JavaBeans components in a visual builder tool to construct an application.

**JNDI for naming databases**

The Java Naming and Directory Interface (JNDI) API can be used in addition to a JDBC technology-based driver manager (JDBC driver manager) to obtain a connection to a database. When an application uses the JNDI API, it specifies a logical name that identifies a particular database instance and JDBC driver for accessing that database.

This has the advantage of making the application code independent of a particular.

**JDBC driver and JDBC technology URL**

**Connection Pooling**

The JDBC API contains ‘hooks’ that allow connection pooling to be implemented on top of the JDBC driver layer. This allows for a single connection cache that spans the different JDBC drivers that may be in use. Since creating and destroying database connections is expensive, connection pooling is important for achieving good performance, especially for server applications.

**Distributed transaction support**

Support for distributed transactions has been added as an extension to the JDBC API. This feature allows a JDBC driver to support the standard 2-phase commit protocol used by the Java Transaction Service (JTS) API.

**Other new features**

Support for character streams has been added. This means that character data can be retrieved and sent to the database as a stream of internationalized Unicode characters. Methods to allow java.math.BigDecimal values to be returned with full precision have also been added. Support for time zones has been added.

**What’s actually changed?**

**A New Package**

The JDBC API has been factored into two complementary components. The first component is API that is core to the Java platform (the core JDBC 2.1 API) and comprises the updated contents of the java.sql package. This document contains the specification for the core JDBC 2.1 API. The second component, termed the JDBC 2.0 Optional Package API, comprises the contents of a new package, javax.sql, which as its name implies will be delivered as an optional package to the Java platform (formerly Java Standard Extension). The JDBC 2.0 Optional Package API is described in a separate document. The java.sql package contains all of the additions that have been made to the existing interfaces and classes, in addition to a few new classes and interfaces. The new javax.sql package has been introduced to contain the parts of the JDBC API which are closely related to other pieces of the Java platform that are themselves Optional Pack-ages. such as the Java Naming and Directory Interface (JNDI), and the Java Transaction Service (JTS).

In addition, some advanced features that are easily separable from the core JDBC API, such as connection pooling and rowsets, have also been added to javax.sql. Putting these advanced facilities into an optional package instead of into core will help keep the core JDBC API small and focused. Since optional packages are downloadable, it will always be possible to deploy an application which uses the features in the JDBC Optional Package that will “run any-where,” since if an optional package isn’t installed on a client machine, it can be downloaded along with the application that uses it.

**Changes to Classes and Interfaces**

The list below contains all of the JDBC 2.1 API core classes and interfaces. Interfaces and classes that are new are listed in bold type. All of the interfaces and classes present in the JDBC 1.0 API are also present in the core JDBC 2.1 API, however, some of the JDBC 1.0 technology interfaces have gained additional methods.

The separate core JDBC 2.1 API documentation contains the Java programming language definitions of the java.sql interfaces and classes listed above. The figure below shows the more important core interfaces and their relationships. The important relationships between interfaces have not changed with the introduction of the new JDBCAPI.

**Fig 2.1**

**Result Set Enhancements**

This chapter discusses the new functionality that has been added to result sets. The goal of the enhancements is to add two new basic capabilities to result sets: scrolling and updatability. Several methods have also been added to enable a JDBC driver to deliver improved performance when processing results. A variety of examples are included to illustrate the new features.

**Scrolling**

A result set created by executing a statement may support the ability to move backward (last-to-first) through its contents, as well as forward (first-to-last). Result sets that support this capability are called scrollable result sets. Result sets that are scrollable also support relative and absolute positioning. Absolute positioning is the ability to move directly to a row by specifying its absolute position in the result set, while relative positioning gives the ability to move to a row by specifying a position that is relative to the current row. The definition of absolute and relative positioning in the JDBC API is modeled on the X/Open SQL CLI specification.

**Result Set types**

The JDBC 1.0 API provided one result set type—forward-only. The JDBC 2.1 core API provides three result set types: forward-only, scroll-insensitive, and scroll-sensitive. As their names suggest, the new result set types support scrolling, but they differ in their ability to make changes visible while they are open.

A scroll-insensitive result set is generally not sensitive to changes that are made while it is open. A scroll-insensitive result set provides a static view of the underlying data it contains. The membership, order, and column values of rows in a scroll-insensitive resultset are typically fixed when the result set is created. On the other hand, a scroll-sensitive result set is sensitive to changes that are made while it is open, and provides a ‘dynamic’ view of the underlying data. For example, when using a scroll-sensitive result set, changes in the underlying column values of rows are visible. The membership and ordering of rows in the result set may be fixed—this is implementation defined.

**Performance**

Two performance hints may be given to a JDBC 2.1 technology-enabled driver to make access to result set data more efficient. Specifically, the number of rows to be fetched from the database each time more rows are needed can be specified, and a direction for processing the rows—forward, reverse, or unknown—can be given as well. These values can be changed for an individual result set at any time. A JDBC driver may ignore a performance hint if it chooses.

**Introduction to Servlets**

Servlets provide a Java(TM)-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 applets in 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.

**What is a Servlet?**

Servlets are modules that extend request/response-oriented servers, such as Java-enabled web servers. For example, a servlet might be responsible for taking data in an HTML order-entry form and applying the business logic used to update company's order database.



**Fig 2.2**

Servlets are to servers what applets are to browsers. Unlike applets, however, servlets have no graphical user interface.

Servlets can be embedded in many different servers because the servlet API, which you use to write servlets, assumes nothing about the server's environment or protocol. Servlets have become most widely used within HTTP servers; many web servers support the Servlet API.

**Use Servlets instead of CGI Scripts!**

Servlets are an effective replacement for CGI scripts. They provide a way to generate dynamic documents that is both easier to write and faster to run. Servlets also address the problem of doing server-side programming with platform-specific APIs: they are developed with the Java Servlet API, a standard Java extension.

So use servlets to handle HTTP client requests. For example, have servlets process data posted over HTTPS using an HTML form, including purchase order or credit card data. A servlet like this could be part of an order-entry and processing system, working with product and inventory databases, and perhaps an on-line payment system.

**Other Uses for Servlets:**

Here are a few more of the many applications for servlets:

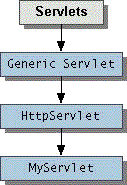
* Allowing collaboration between people. A servlet can handle multiple requests concurrently, and can synchronize requests. This allows servlets to support systems such as online conferencing.
* Forwarding requests. Servlets can forward requests to other servers and servlets. Thus servlets can be used to balance load among several servers that mirror the same content, and to partition a single logical service over several servers, according to task type or organizational boundaries.

## 5.4 Architecture of the Servlet Package

The javax.servlet package provides interfaces and classes for writing servlets. The architecture of the package is described below.

**The Servlet Interface**

The central abstraction in the Servlet API is the Servlet interface. All servlets implement this interface, either directly or, more commonly, by extending a class that implements it such as HttpServlet.



**Fig 2.3**

The Servlet interface declares, but does not implement, methods that manage the servlet and its communications with clients. Servlet writers provide some or all of these methods when developing a servlet.

**Client Interaction**

When a servlet accepts a call from a client, it receives two objects:

* A ServletRequest, which encapsulates the communication from the client to the server.
* A ServletResponse, which encapsulates the communication from the servlet back to the client.

ServletRequest and ServletResponse are interfaces defined by the javax.servlet package.

**The ServletRequest interface allows the servlet access to:**

* Information such as the names of the parameters passed in by the client, the protocol (scheme) being used by the client, and the name of the remote host that made the request and the server that received it.
* The input stream, ServletInputStream. Servlets use the input stream to get data from clients that use application protocols such as the HTTP POST and PUT methods.

Interfaces that extend ServletRequest interface allow the servlet to retrieve more protocol-specific data. For example, the HttpServletRequest interface contains methods for accessing HTTP-specific header information.

The ServletResponse interface gives the servlet methods for replying to the client. It:

* Allows the servlet to set the content length and MIME type of the reply.
* Provides an output stream, ServletOutputStream, and a Writer through which the servlet can send the reply data.

Interfaces that extend the ServletResponse interface give the servlet more protocol-specific capabilities. For example, the HttpServletResponse interface contains methods that allow the servlet to manipulate HTTP-specific header information.

}

}

That's it!

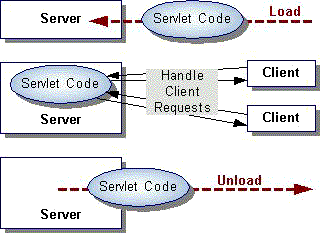
The classes mentioned in the Architecture of the Servlet Package section are shown in the example in bold:

* SimpleServlet extends the HttpServlet class, which implements the Servlet interface.
* SimpleServlet overrides the doGet method in the HttpServlet class. The doGet method is called when a client makes a GET request (the default HTTP request method), and results in the simple HTML page being returned to the .

##### Servlet Lifecycle:

**Each servlet has the same life cycle:**

* A server loads and initializes the servlet
* The servlet handles zero or more client requests
* The server removes the servlet

**Fig 2.4**

**Initializing a Servlet:**

When a server loads a servlet, the server runs the servlet's init method. Initialization completes before client requests are handled and before the servlet is destroyed.

Even though most servlets are run in multi-threaded servers, servlets have no concurrency issues during servlet initialization. The server calls the init method once, when the server loads the servlet, and will not call the init method again unless the server is reloading the servlet. The server can not reload a servlet until after the server has destroyed the servlet by calling the destroy method.

**The init Method**

The init method provided by the HttpServlet class initializes the servlet and logs the initialization. To do initialization specific to your servlet, override the init() method following these rules:   
If an initialization error occurs that renders the servlet incapable of handling client requests, throw an UnavailableException.

An example of this type of error is the inability to establish a required network connection.

* Do not call the System.exit method

### Initialization Parameters

The second version of the init method calls the getInitParameter method. This method takes the parameter name as an argument and returns a String representation of the parameter's value.

If, for some reason, you need to get the parameter names, use the getParameterName method.

**Destroying a Servlet**

Servlets run until the server are destroys them, for example at the request of a system administrator. When a server destroys a servlet, the server runs the servlet's destroy method. The method is run once; the server will not run that servlet again until after the server reloads and reinitializes the servlet. When the destroy method runs, another thread might be running aservicerequest.

**PROBLEM DEFINITION**

Earlier people used to move from hospital to hospital for availability of a suitable doctor. Certain path tests were not carried out in all path labs. So getting a test done was a headache. Blood which is the primary necessity in a patient wasn’t available across all blood banks. So getting a suitable blood in time of emergency was too hectic. Moreover, long waits for ambulance or mortuary vans was again a problem. And also, in hotel sector tourist information centres spend a lot of their time calling hotels to check the availability. Also, many smaller hotels suffer from not being able to advertise their availability to tourist centres. Moreover, in education sector when students from outside the states migrate to other cities, they face a lot of problem regarding the studies for searching of a good institute, they have to visit a lot of places that take a lot of time. people facing a dilemma where to and how to spend their weekends according to their requirements. Above all these things, people were duped with money being charged by the organizations with these services. Sometimes Organisations also had few people availing their services because of their low popularity. Computerization helps to overcome all these problems, by integrating the system that is the above said jobs can be done by a single click. In this techno world where almost, everyone can reach internet things at just a single click, making things faster, easier and effective.

## 1.2.1 Existing System:

1. One who needs to consult a doctor for certain treatment has to move from hospital to hospital to check availability of a suitable doctor in a specified timing. And sometimes hospitals do loot people with money.
2. In case of path labs, we people are least aware about whether a particular test is performed in a specified patho-lab and sometimes we are not prepared with the prices of the tests.
3. Well The gift of blood is the gift of life. There is no substitute for human blood. Every two seconds someone needs blood. India has an acute shortage of blood and hundreds shed their lives for shortage of blood. Getting blood availability information from various blood banks at the time of need is quite disheartening.
4. More than 300 blood donations are needed every day for our city. But managing them and distributing blood equally across all blood banks is another challenge.
5. We do have Uber, OLA as our cab service but we don't have any such service to track or use any ambulance to ferry us to hospitals.
6. Its more pathetic when we require an emergency service like a burn unit or trauma care unit to reach out in minutes. And sometimes long waits do turn fatal.
7. Several health check-up events do take off around the city but to no effect as information regarding those doesn’t reach everyone.
8. Several old , beautiful , informatic sight seeing places but n awareness or information regarding such places .
9. Having no hotel rooms available at a place and have to move from one to other and paying heavy prices without any awareness.
10. Roaming from institutions to institutions for collecting information regarding courses and also getting stuck with the location.
11. Well another interesting fact comes up, which we don’t face very often but sometimes we do have to. It’s in time of someone’s death we require a mortuary van to take the dead for funeral but cabs generally oppose to provide that service. So getting a mortuary van available takes long time.

## Proposed System:

1. All services related to health under one platform.
2. Services providing information based on GPS.
3. One touch facility to avail ambulance, hospitals ,path-labs, blood banks, Mortuary vans and a host of emergency services, institutions , tourist places ,hotels.
4. A comparative selection of services possible through various filters like distance, money, etc.
5. A host of related services like blood donation facilities along with information regarding various events like health check-up camps, etc.

**SYSTEM ANALYSIS**

Analysis is the process of understanding the existing system by gathering and interpreting the facts, diagnosing the problems. It is not just to determining the how best to solve the manual system problems, it should also work for the system observes the feasibility of system then design, coding phases will be executed. Analysis phase delivers requirements specification .The system specification serves as an interface between the designer and developer as well as between developers and users. This describes the external behaviour of the software without bothering about the internal implementation. Specification must be carefully checked for suitability, omission, inconsistencies and ambiguities.

Problem analysis is performed to getting a clear understanding of the needs of the clients and the users and what exactly desired form the software. Analysis leads to the actual specification. During the process of analysis, a massive amount of information is collected in the form of interviews, questionnaires, and information from documentation, and so forth. The major problem during analysis is resolving how to organize the information from documentation, and so forth. So the information can be effectively evaluated for completeness and consistency.

## Dataflow Diagrams

Dataflow Diagram is a structure analysis tool that is used for graphical representation of Data processes through any organization. The data flow approach emphasis on the logic underlying the system, by using combination of only 4 symbols. It follows a top down approach. A full description of a system actually consists of set of DFDs, which comprises of various levels. And initial over view model is exploded lower level diagrams that show additional feature of the system. Further each process can be broken down into a more detailed DFD. This occurs repeatedly until sufficient details are described.

### DFD symbols

* **Square**

It defines a source (originator) or destination of system data.

* **Arrow**

It indicates data flow-data in motion. It is a pipeline through which information flows.

* **Circle or Bubble**

It represents a process that transforms incoming data flow(s) into outgoing data flow(s).

* **Open Rectangle**

Description: bä

It is a data store-data at rest, or a temporary repository of data. Here we are providing only the Data Flow Diagram.

They are explained by

* GANE and SARON method
* DEMACRO YORDAN method

#### GANE AND SARSON NOTATION

* DATA FLOW
* DATA STRUCTURE

* EXTERNAL ENTITY

OR DATA LINK

* PROCESS
* DATA BASE Description: bä

**DEMACRO YORDAN NOTATION**

* DATA FLOW
* DATA STRUCTURE

* EXTERNAL ENTITY

OR DATA LINK

* PROCESS
* DATA BASE 

## 

## UML Diagrams

**UML (UNIFIED MODELLING LANGUAGE)**

The unified modeling language is a standard language for specifying, Visualizing, Constructing and Documenting the software system and its components. It is a graphical language which provides a vocabulary and set of semantics and rules. The UML focuses on the conceptual and physical representation of the system. It captures the decisions and understandings about systems that must be constructed.

It is used to understand, design, configure, maintain and control information about the systems.

**Visualizing:**

Through UML we see or visualize an existing system and ultimately we visualize how the system is going to be after implementation. Unless we think we cannot implement.

UML helps to visualize how the components of the system communicate and interact with each other.

**Specifying:**

Specifying means building models that are precise, unambiguous and complete UML addresses the specification of all the important Analysis Design, Implementation decisions that must be made in developing and deploying a software system.

Constructing: UML’s models can be directly connected to a variety of programming language through mapping a model from UML to a programming language like Java or C++ or VB.

Forward Engineering and Reverse Engineering is possible through UML.

**Documenting:**

The deliverables of a project apart from coding are some artifacts which are critical in controlling, measuring and communicating about a system during its development viz.

Requirements, Architecture, Design, Source code, Project plans, Tests, Prototypes, Releases etc.

**Diagrams in UML:**

Diagrams are graphical presentation of set of elements. Diagrams project a system, or visualize a system from different angles and perspectives. The UML has nine diagrams these diagrams can be classified into the following groups.

**Static:**

1. Class diagrams.

2. Object diagrams.

3. Component diagrams.

4. Deployment diagrams

**Dynamic:**

1. Use case diagram.

2. Sequence diagram.

3. Collaboration diagram.

4. State chart diagram.

5. Activity diagram.

**Static or structural diagrams:**

**Class diagram:**

This shows a set of classes, interfaces, collaborations and their relationships. There are the most common diagrams in modelling the object oriented systems and are used to give the static view of a system.

**Object diagram:**

Shows a set of objects and their relationships and are used to show the data structures, the static snapshots of instances of the elements in a class diagram. Like class diagram, the object diagrams also address the static design view or process view of a system.

**User: Smartcity**

**Component diagram:**

Shows a set of components and their relationships and are used to illustrate the static implementation view of a system. They are related to class diagrams where in components map to one or more classes, interfaces of collaborations.

**Deployment diagram:**

Shows a set of nodes and their relationships. They are used to show the static deployment view of the architecture of a system. They are related to the component diagrams where a node encloses one or more components.

**Dynamic or behavioral diagrams:**

**Use Case diagram:**

Shows a set of use cases and actors and their relationships. These diagrams illustrate the static use case view of a system and are important in organizing and modeling the behaviors of a system.

**Fig 2.8**

**Sequence diagram & collaboration diagram:**

These two diagrams are semantically same i.e. the dynamics of a system can be modeled using one diagram and transform it to the other kind of diagram without loss of information. Both form the, Interaction diagram. Sequence diagram:

**Sequence diagram:**

Sequence diagram is an interaction diagram which focuses on the time ordering of messages it shows a set of objects and messages exchange between these objects. This diagram illustrates the dynamic view of a system.

**Collaboration diagram:**

This diagram is an interaction diagram that stresses or emphasizes the structural organization of the objects that send and receive messages. It shows a set of objects, links between objects and messages send and received by those objects. There are used to illustrate the dynamic vies of a system.

**State Chart Diagram and Activity Diagrams:**

These Diagrams are semantically similar. State chart diagram shows a state machine consisting of states, transitions and activities these illustrates the dynamic view of a system. They focuses on the event ordered Behavior of an object.

**Activity Diagrams:**

Activity diagram shows the flow from one activity to another within a system. The activities may be sequential or branching objects that act and are acted upon. These also show the dynamic view of the system.

Analysis is the detailed study of the various operations performed by a system and their relationships within and outside of the system. A key question is: What must be done to solve the problem? One aspect of analysis is defining the boundaries of the system and determining whether or not candidate system should consider other related systems. During analysis, data are collected on the available files, decision points, and transactions handled by the present system.

## MODULES

The Smartcity includes

1. Admin Module
2. Organisation Module
3. user Module
4. Hotel Module
5. Tourist Module
6. Institute

The description of above module is given in general workflow **Admin Module:**

1. Can vie/update own Information.
2. Can view all registered profile details.
3. Accept or reject new users.
4. Accept or reject new Organisation’s.
5. Can view all registered Organisation details.
6. Grant or deny a service.

**Organisation Module:**

1. Can view their own information.
2. Can view their customers.
3. Can Update their own information.
4. Approve Requests.
5. Further process the requests.

**User Module:**

1. Can view only his/her own profile.
2. Can view Organisations based on the service he/she wants.
3. Request for Services.
4. Get Comparative Statements based on price and distance.
5. Receive bill details and pay.
6. Get Updated About the various Health Events Going on in the city.

**Hotel Module:**

1. Can view their own information.
2. Can view their customers.
3. Can Update their own information.
4. Approve Requests.
5. Further process the requests.
6. Can book and further process payment.

**Tourist Module:**

1. Can view their own information.
2. Can view their customers.

**Institute Module:**

1. Can view their own information.
2. Can view their customers.

### Apart from the above, each registered member can independently choose to change his/her login password and can update the details as well.

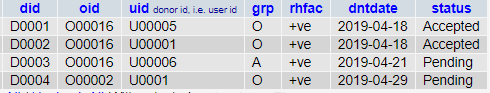
**DESIGN DOCUMENT**

**Database design**

ADMIN



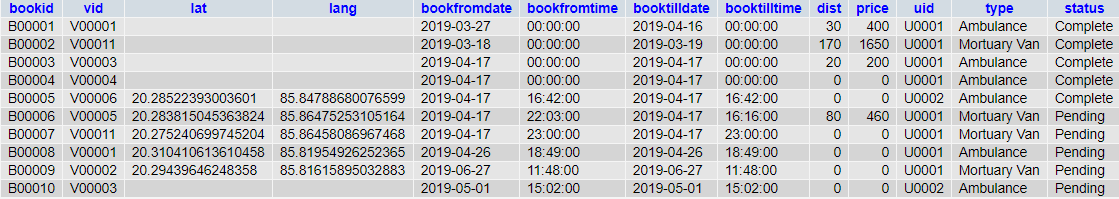
BLOOD DONATION



BLOOD



BOOKING



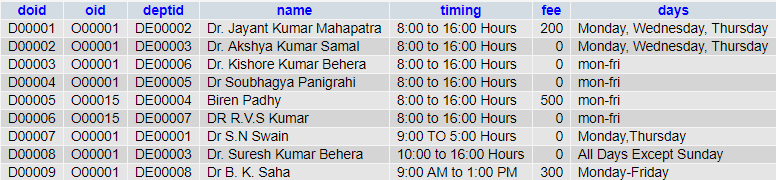
BOOKING ROOM



CUSTOMER



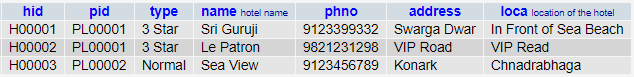
DOCTOR



EMPLOYEE



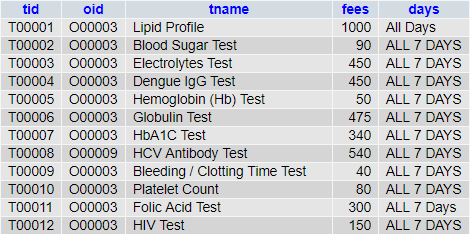
HOTEL



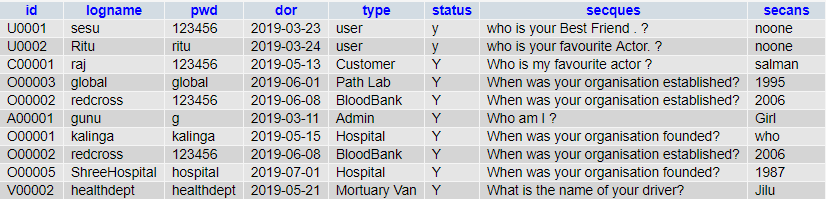
INSTITUTE



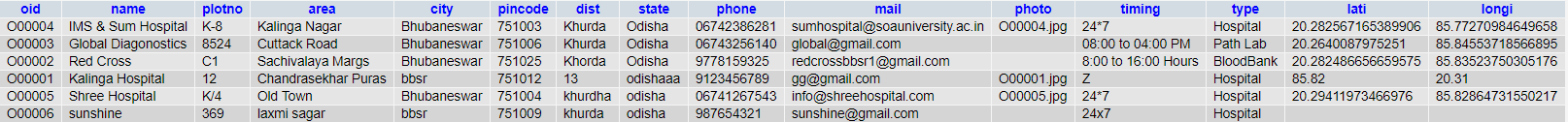
LAB



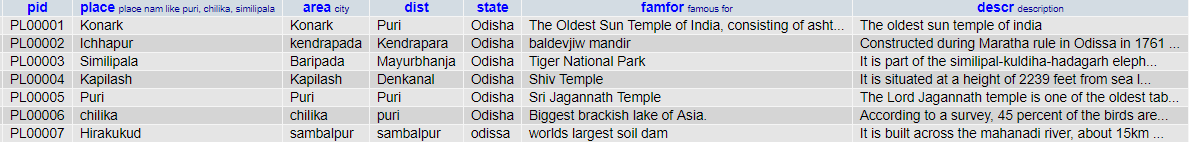
LOGIN



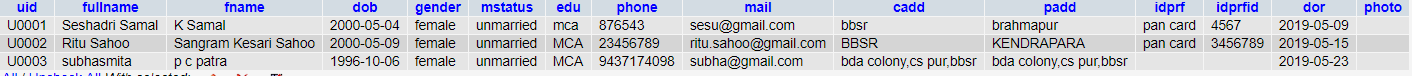
ORGANISATION



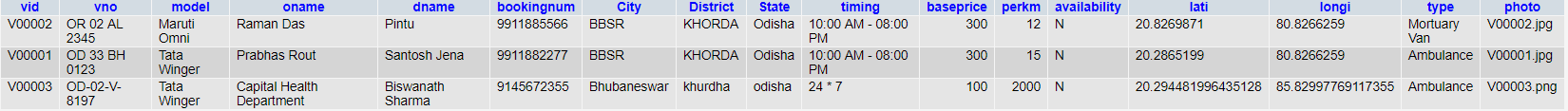
TOURIST PLACE



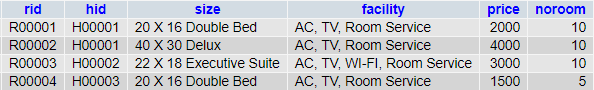
USER



VEHICLE



ROOM



## FEASIBILITY STUDY

Feasibility study is an important phase in the software development process. It enables the developer to have an assessment of the product being developed. It refers to the feasibility study of the product in terms of outcomes of the product, operational use and technical support required for implementing 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
4. **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.

If the benefits are more or less the same as the older system, then it is not feasible to develop the product. This product is economically feasible as it is a need of every mobile developer. And also because it is commercially available to him on the net, which he can download by paying the reasonable price.

1. **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 additional human resource required and their technical expertise.

This product is operationally feasible as it is designed specifically for the mobile application developers. Even this product is tested by implementing this in some other projects like Mobile Browser, MMS editor and even in Client Provisioning.

1. **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 training needed to be given to the people to make the application work. As this product is tested by implementation in several other projects. Its technical feasibility is guaranteed.

**IMPLEMENTATION PLAN**

The main plan for the system developed is to mimic the existing system as it is in the proposed system.

**HARDWARE AND SOFTWARE REQUIREMENT**

## WHAT IS SRS?

Software Requirement Specification (SRS) is the starting point of the software developing activity. As system grew more complex it became evident that the goal of the entire system cannot be easily comprehended. Hence the need for the requirement phase arose. The software project is initiated by the client needs. The SRS is the means of translating the ideas of the minds of clients (the input) into a formal document (the output of the requirement phase.)

The SRS phase consists of two basic activities:

1. **Problem/Requirement Analysis**:

The process is order and more nebulous of the two, deals with understanding the problem, the goal and constraints.

1. **Requirement Specification:** Here, the focus is on specifying what has been found giving analysis such as representation, specification languages and tools, and checking the specifications are addressed during this activity.

The requirement phase terminates with the production of the validate SRS document. Producing the SRS document is the basic goal of this phase.

## ROLE OF SRS:

The purpose of the Software Requirement Specification is to reduce the communication gap between the clients and the developers. Software Requirement Specification is the medium though which the client and user needs are accurately specified. It forms the basis of software development. A good SRS should satisfy all the parties involved in the system.

**PURPOSE:**

The purpose of this document is to describe all external requirements for the E-Learning System. It also describes the interfaces for the system.

**SCOPE:**

This document is the only one that describes the requirements of the system. It is meant for the use by the developers, and will also by the basis for validating the final delivered system. Any changes made to the requirements in the future will have to go through a formal change approval process. The developer is responsible for asking for clarifications, where necessary, and will not make any alterations without the permission of the client.

**Software Requirements :**

#### Operating System : Windows 7 or higher*.*

Back End : My Sql 5.1

Validation : JavaScript

Front End : HTML, CSS

Technologies : JDK 1.6,J2EE, JDBC, SERVLETS, JSP

Web Server : Apache Tomcat 8.0

Web Technologies : HTML, JavaScript

Web Browsers : Mozila Firefox,Google Chrome

IDE : Dreamweaver8

Documentation Tool : Microsoft word 2016

**Hardware Requirements:**

Processor : Intel core -2 duo (min).

RAM : 500MB (min).

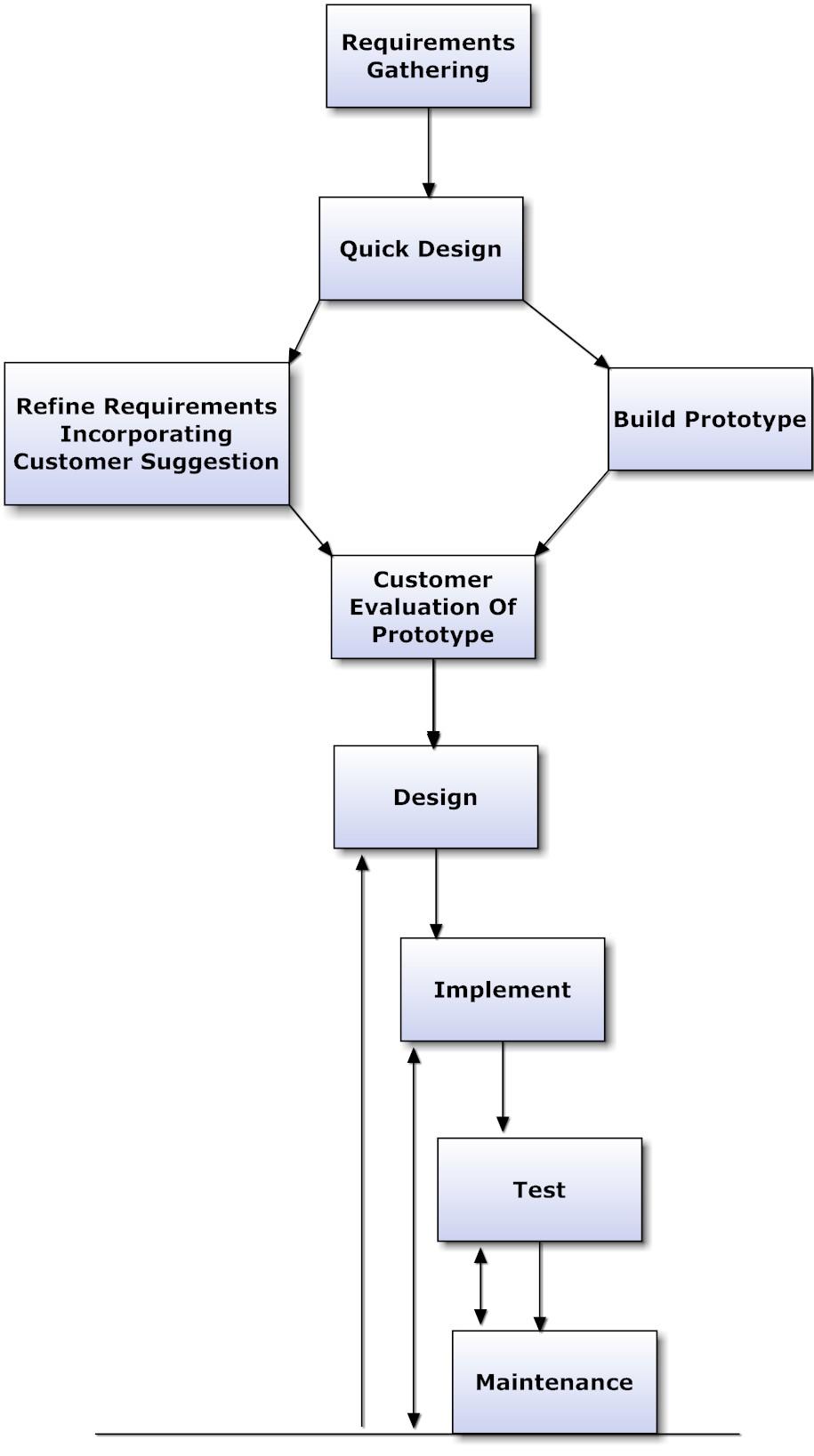
Hard Disk : 40 GB (min).

**PROJECT LIFECYCLE**

A prototype model has been used that suggests that a working prototype of the system should be built before carrying out the development of the actual software. A prototype is a toy implementation of the system. By constructing the prototype and submitting it for user evaluation, many customer requirements get properly defined and technical issues get resolved by experimenting with the prototype which finally minimizes the change requests from the customer and the associated redesign costs.

The software engineering paradigm is also called a process model. A process model for software engineering is chosen based on the nature of the project and application, the methods and tools to be used, and the controls and deliverables that are required.

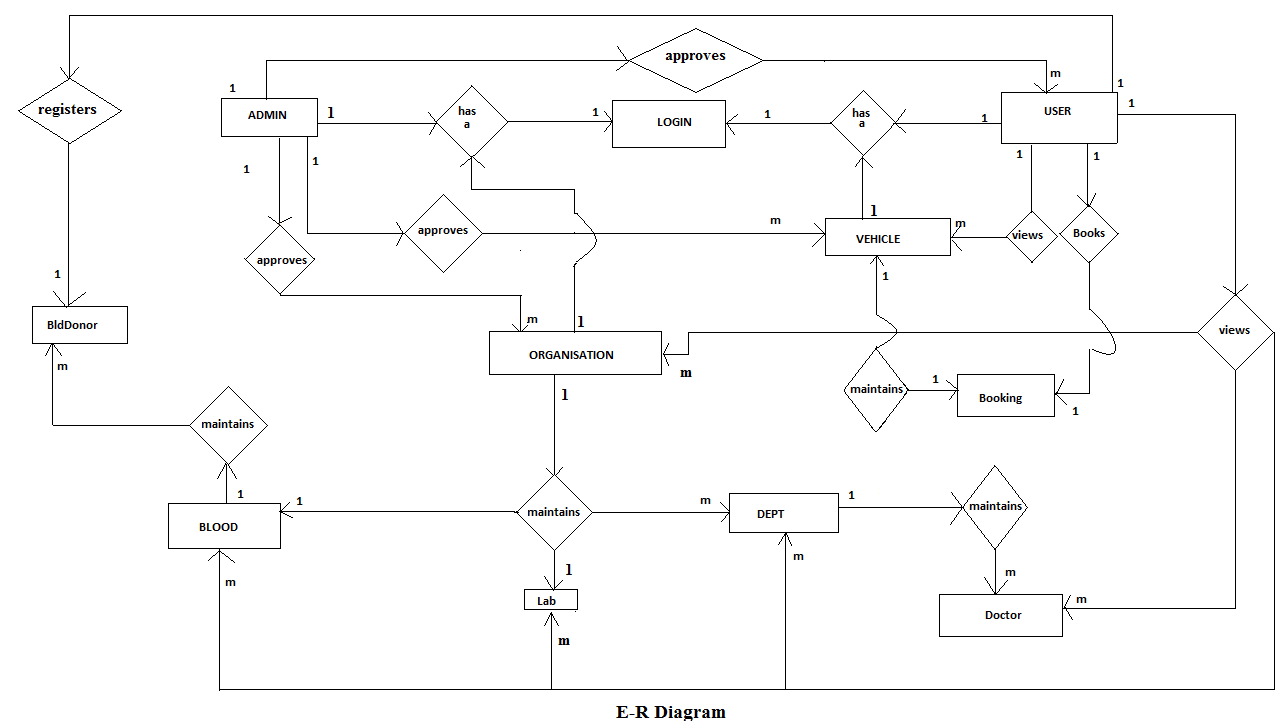
In this project, System Development Life Cycle (SDLC) model is used. The various phase of this method can be described briefly follows.

****

**Fig: Prototyping model of software development**

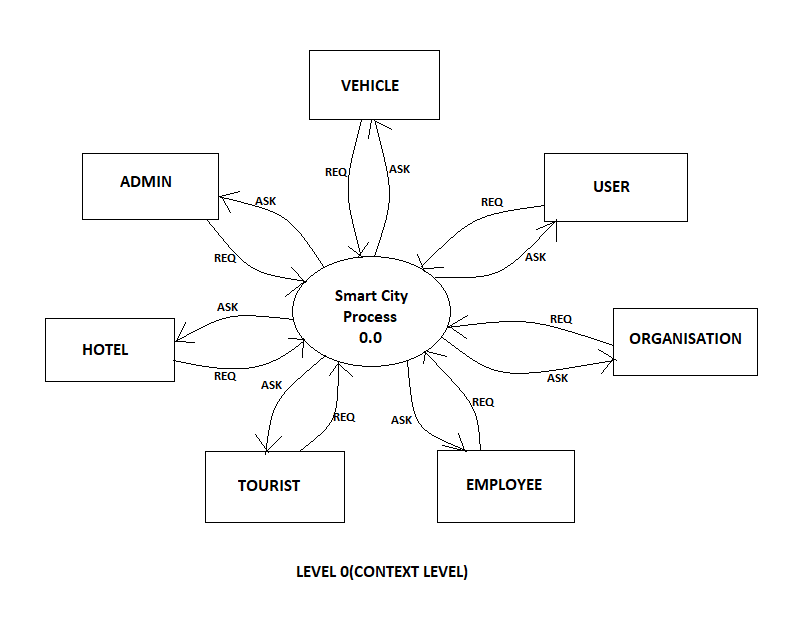
|  |  |  |
| --- | --- | --- |
| **Phase** | **What is done?** | **What is the end product?** |
| Requirements  Determination | Determine requirements to be met by System being compiled. | Set of priorities |
| Requirements Speciation | Draw up understandable plan of what the system will provide as outputs.  Determine needs and priority by consensus among end users. | Detailed specifications of information to be provided. |
| Feasibility Study | Taking into account available resources such as human, computers, time and money and  find whether specified requirements can be met. | Feasibility Document specifying resource needs and availability. Expected costs vs. Benefits of the system are carried |
| System Design | Logical design of programs, design of databases, test and implementation plan. | Design programs. Database, test plan. |
| System Evolution | Find out from users if system meets their requirement. | Evaluation report with suggestion to improve. |
| System  Modification/ Maintenance | Change system, adding or deleting features to satisfy users(modified) needs | Improved system containing modifications and improvement |
|  |  |  |

**ER DIAGRAM**

****

**DATA FLOW DIAGRAM**

### CONTEXT LEVEL DIAGRAM



**Fig 2.1 : Context Level Diagram**

organisation

dept

doctor

lab

vehicle

user details

place

hotel

booking

booking

room

Location,timing

hotelname,location

Status,bookid,bookfrom,uid

uid

Distance,price,uid

Vid,vno,model,oname

tid,name,price,timing,oid

doid

depid

Phone,mall,timing

Login details

oid

vid

uid

Id,pwd,secques,secans

**Level 1 DFD (Registration)**

LOGIN

USER

Check user name

0.1.2

Validate user name interface

0.1.1

Registration process

0.1.3

Select username

User name

Username exist

PersonalDetails

Username,pwd

Personal data

Username

blddonation

doctor

name,location

tid,name,price,timing

Doid,name,fees,days

,bldid,oid,group

organisation

Login details

Oid,type

Oid,type

Oid,type

instid

Id,psw

status

Oid,type

Oid

uid,pwd

,did,lastdate,futuredate

**Level 2 DFD (Organisation Management)**

vehicle

vehicle

Login details

booking

Id,pwd

Distance,price,uid

Availability

type

vid

**Level 2 DFD (Vehicle Management)**

User details

Login details

blood

Bld donation

booking

User details

uid

Group,uid,oid

Did,oid,lastdate

Bookform,bookid,status

Name,fname,mail,phone

unit

uid

uid

uid

uid

uid

status

Name,fname,cadd,padd,phone

uid,psw

uid

**Level 2 DFD (User Activity)**

booking

placeUser name does not exist

hotel

room

paymentvid.,type

select

detail

selectFutrdate,status

select

select

detail

detail

detailblood

Select cust

Get booking details

details

Insert customer id room idlab

Booking id generate billvehicle

bookinguser details

**Level 2 DFD (Hotel Management)**

Login detailslab

pid

pidUserid,pwd

pid

status,bookid,vid,bookfrom

placeDepid,name,type,timing

timing,location

**Level 2 DFD (Tourist Management)**

organisation

deptRegistered user

0.1.4

dept

doctor

doctorRegistered user

Phone,mail,plot no

Depid,name,type,timimgVid,vno,model,oname

Doid,depidname,fees,days

depidVid,type

doidVid,type

oidstatus

oiddept

oid

oid

Oid,typeinstitute

**Level 2 DFD (Hospital Management)**

organisation

lab

oId, type

oid

Phone,mail,timing

Tid,name,pricetiming,oid

Tid,name,price,oid,timing

oid

oid

**Figure 2.7 : Level 3 DFD (Path lab Management)**

blood

bldodnation

organisation

Bldid,oid

Group,rhfac unit

Did,oid,lastdate,futuredate

Oid,nameplotno,mail,phone

oid

oid

oid

Oid,type

**Figure 2.7 : Level 3 DFD (blood bank management)**

**TESTING**

Testing is a process, which reveals errors in the program. It is the major quality measure employed during software development. During software development. During testing, the program is executed with a set of test cases and the output of the program for the test cases is evaluated to determine if the program is performing as it is expected to perform. In order to make sure that the system does not have errors, the different levels of testing strategies that are applied at differing phases of software development are:

**1. Unit Testing:**

Unit Testing is done on individual modules as they are completed and become executable. It is confined only to the designer's requirements.

Each module can be tested using the following two strategies:

1. **Black Box Testing:**

In this strategy some test cases are generated as input conditions that fully execute all functional requirements for the program. This testing has been uses to find errors in the following categories:

* 1. Incorrect or missing functions
  2. Interface errors
  3. Errors in data structure or external database access
  4. Performance errors
  5. Initialization and termination errors.

In this testing only the output is checked for correctness.

The logical flow of the data is not checked.

Ii) **White Box testing:**

In this the test cases are generated on the logic of each module by drawing flow graphs of that module and logical decisions are tested on all the cases. It has been uses to generate the test cases in the following cases:

1. Guarantee that all independent paths have been executed.
2. Execute all logical decisions on their true and false Sides.
3. Execute all loops at their boundaries and within their operational bounds.
4. Execute internal data structures to ensure their validity.

**2. Integration Testing:**

Integration testing ensures that software and subsystems work together a whole. It tests the interface of all the modules to make sure that the modules behave properly when integrated together.

**3. System Testing:**

Involves in-house testing of the entire system before delivery to the user. It's aim is to satisfy the user the system meets all requirements of the client's specifications.

**4. Acceptance Testing:**

It is a pre-delivery testing in which entire system is tested at client's site on real world data to find errors.

#### Test Approach:

**Testing can be done in two ways:**

* 1. Bottom up approach
  2. Top down approach

1. **Bottom up Approach:**

Testing can be performed starting from smallest and lowest level modules and proceeding one at a time. For each module in bottom up testing a short program executes the module and provides the needed data so that the module is asked to perform the way it will when embedded within the larger system. When bottom level modules are tested attention turns to those on the next level that use the lower level ones they are tested individually and then linked with the previously examined lower level modules.

1. **Top down approach:**

This type of testing starts from upper level modules. Since the detailed activities usually performed in the lower level routines are not provided stubs are written. A stub is a module shell called by upper level module and that when reached properly will return a message to the calling module indicating that proper interaction occurred. No attempt is made to verify the correctness of the lower level module.

**Validation:**

The system has been tested and implemented successfully and thus ensured that all the requirements as listed in the software requirements specification are completely fulfilled. In case of erroneous input corresponding error messages are displayed.

# **IMPLEMENTATION & MAINTANANCE**

## User Manual:

**Login:**

If you (end user) want to enter into the application, then if you are existing user then you should enter through login form which checks for authorized users. If you are new user then you have to register your details through registration form with your own identification name and password which gives you an unique identification to you or your firm.

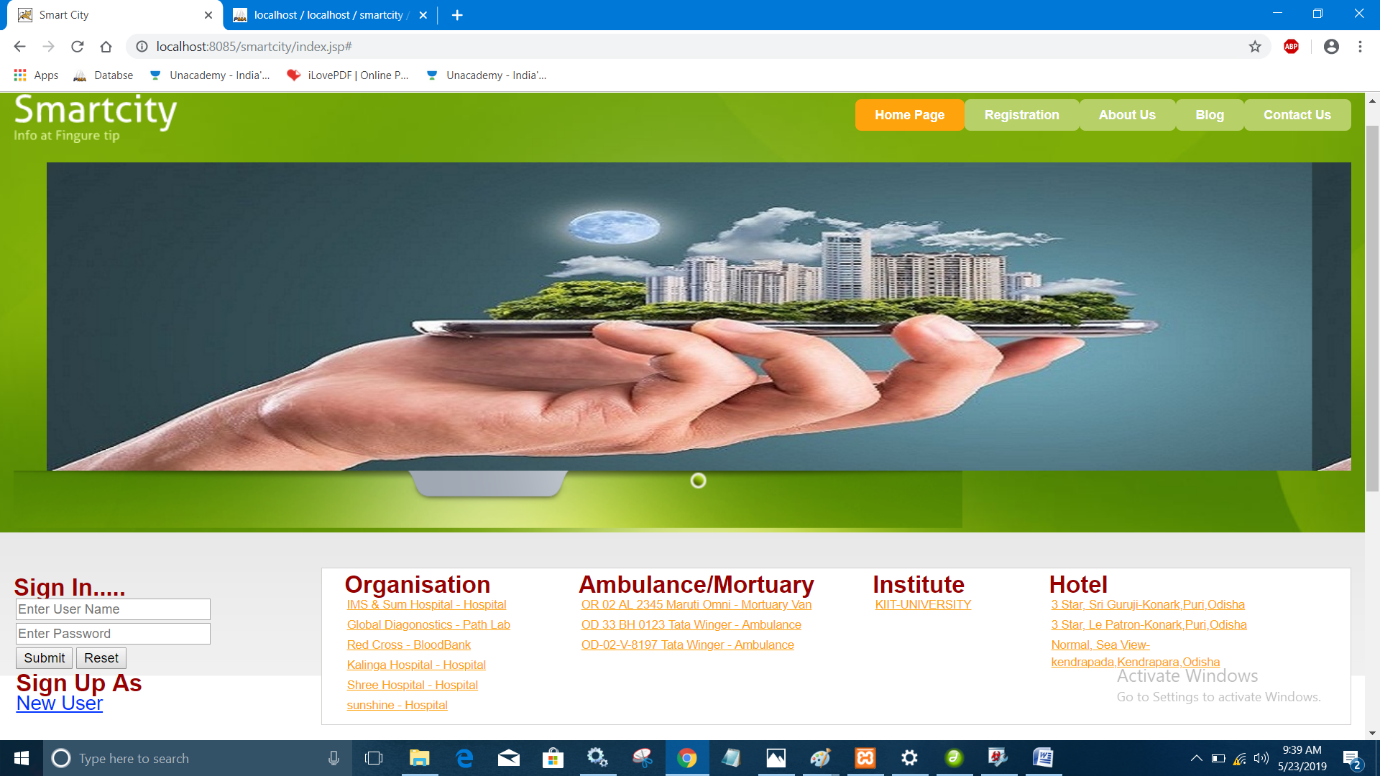
**User type:**

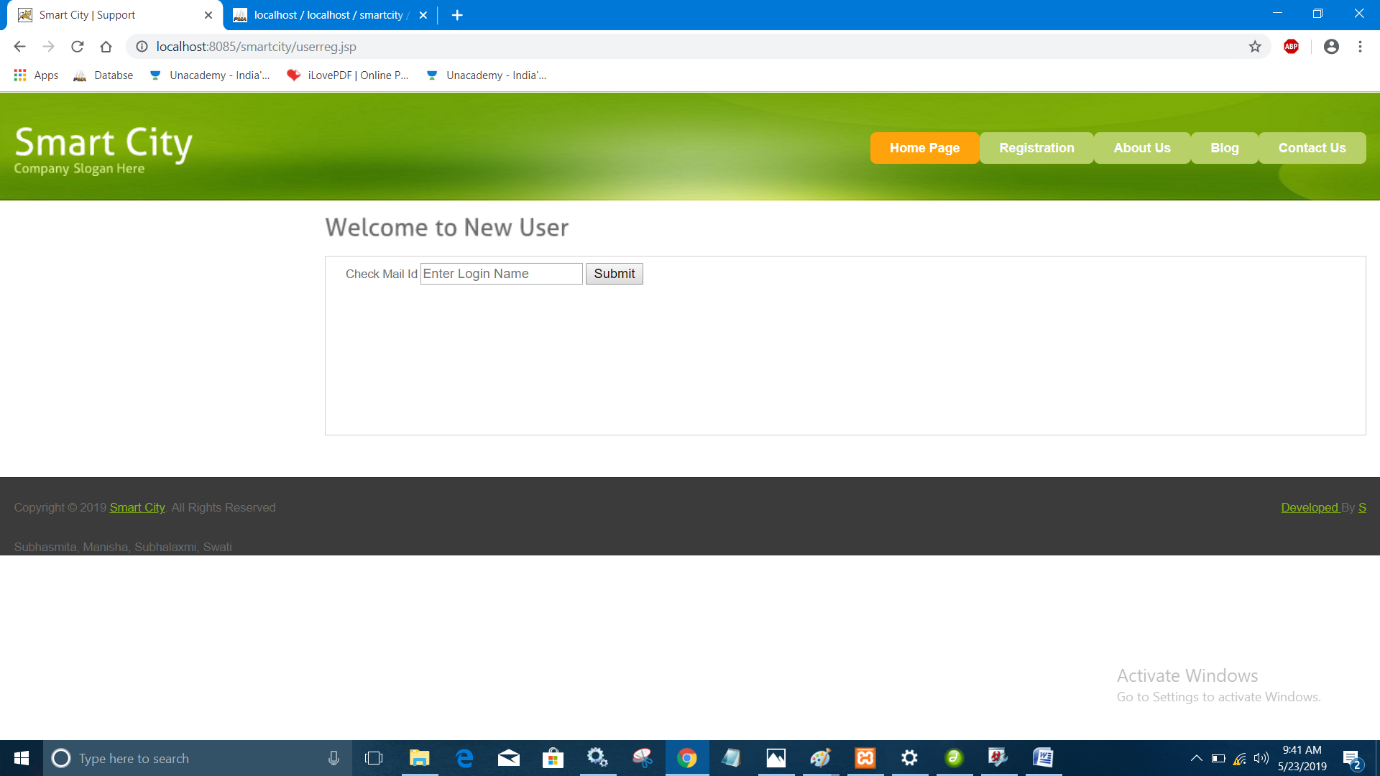
Once registered, depending on your designation you can enter your login id and password and enjoy the facilities available to you as per your id, i.e., Admin or Hospital or Ambulance or General User.

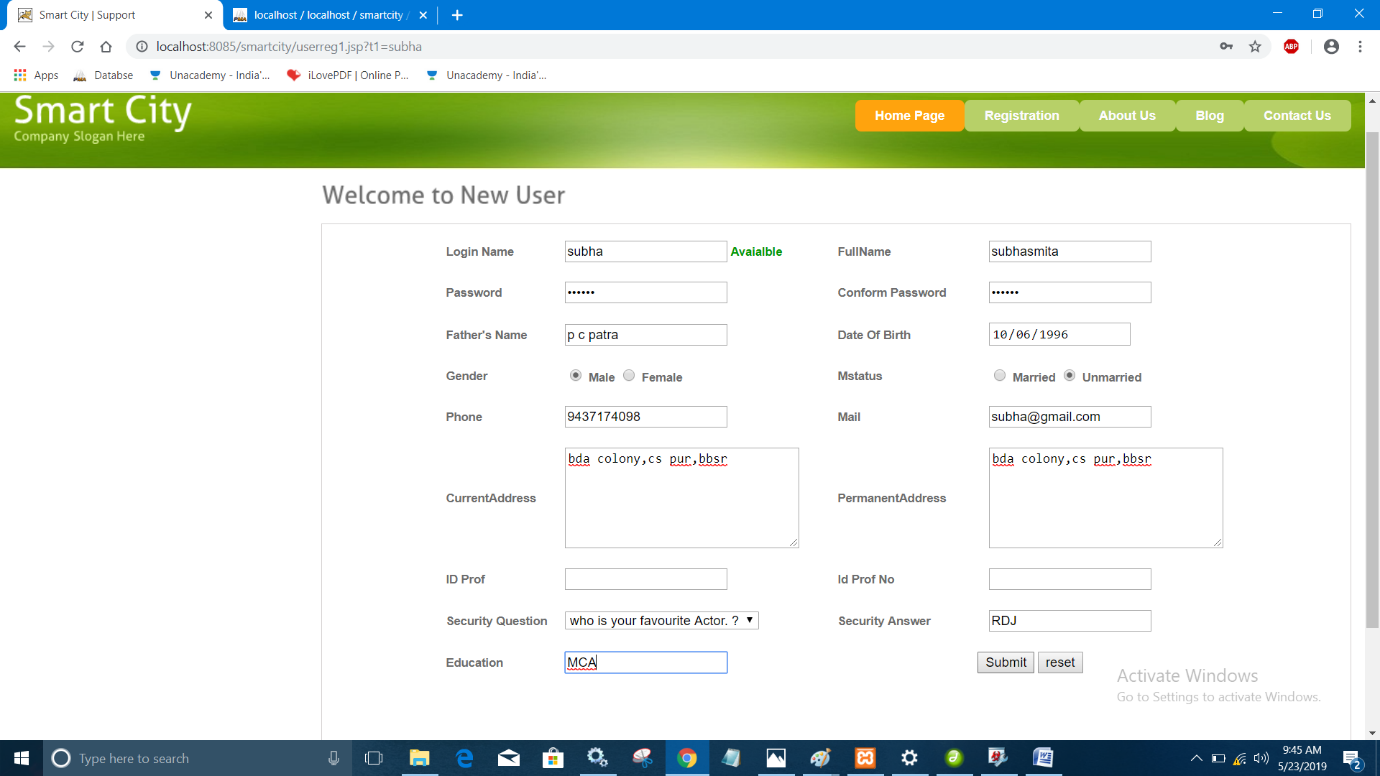
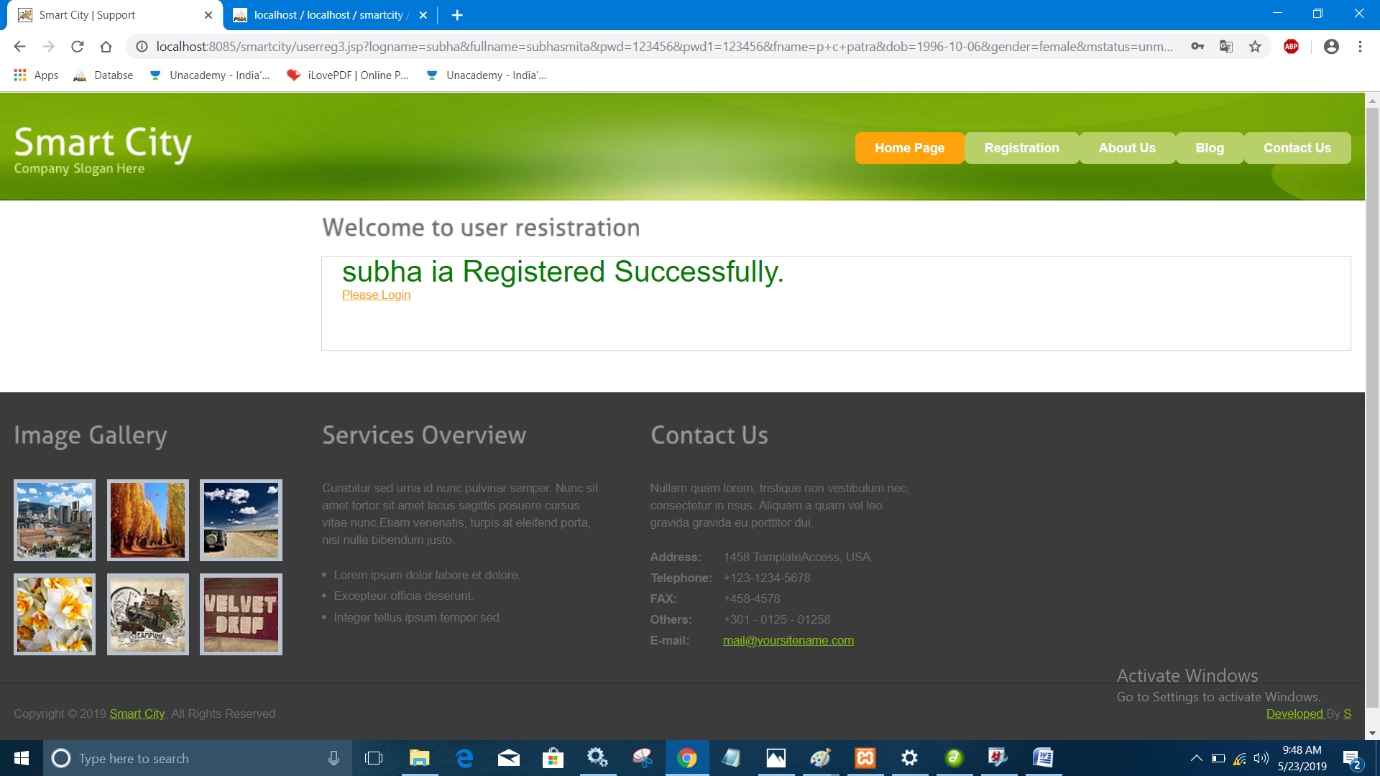
**Profile Functionalities:**

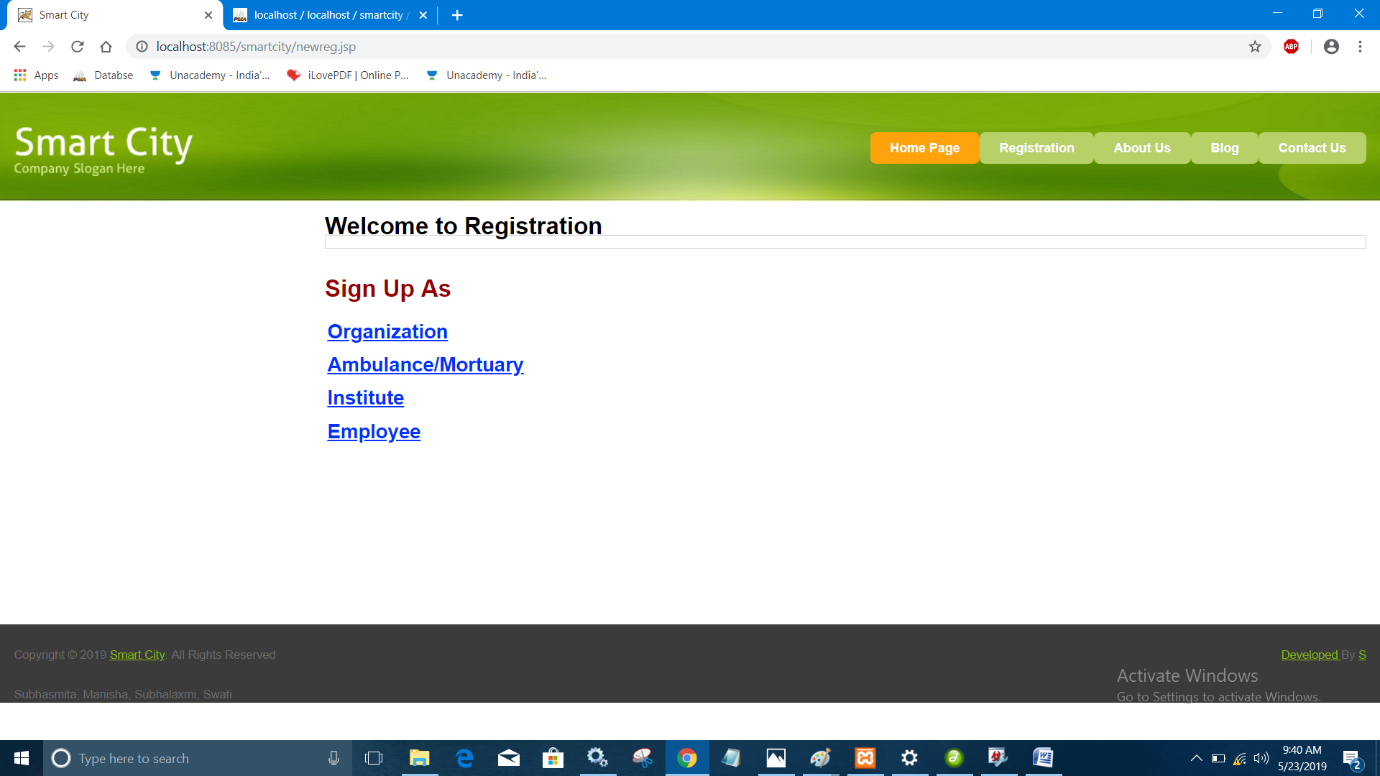
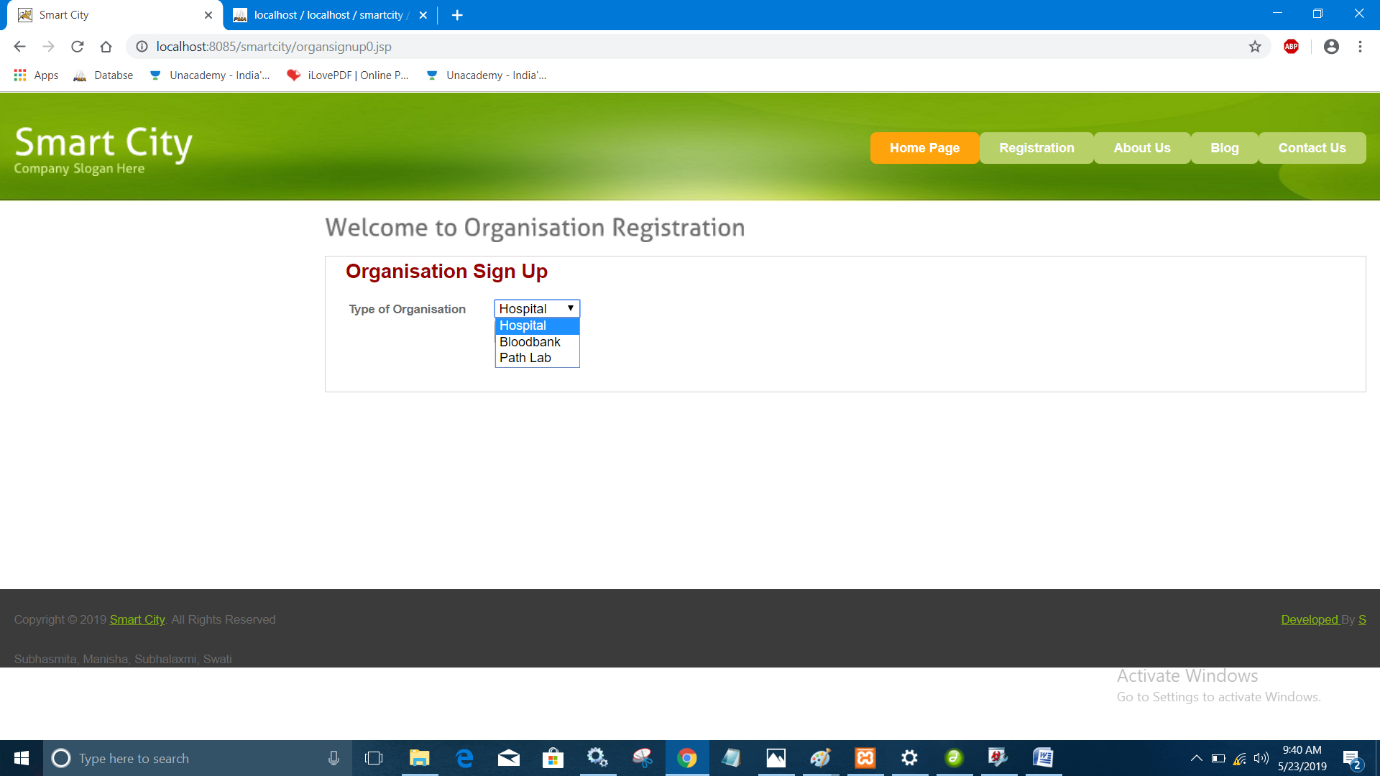
Based on the type of user you are, the application provides you a list of Activities. You can View, Edit, Manipulate, Delete data as per your necessity.

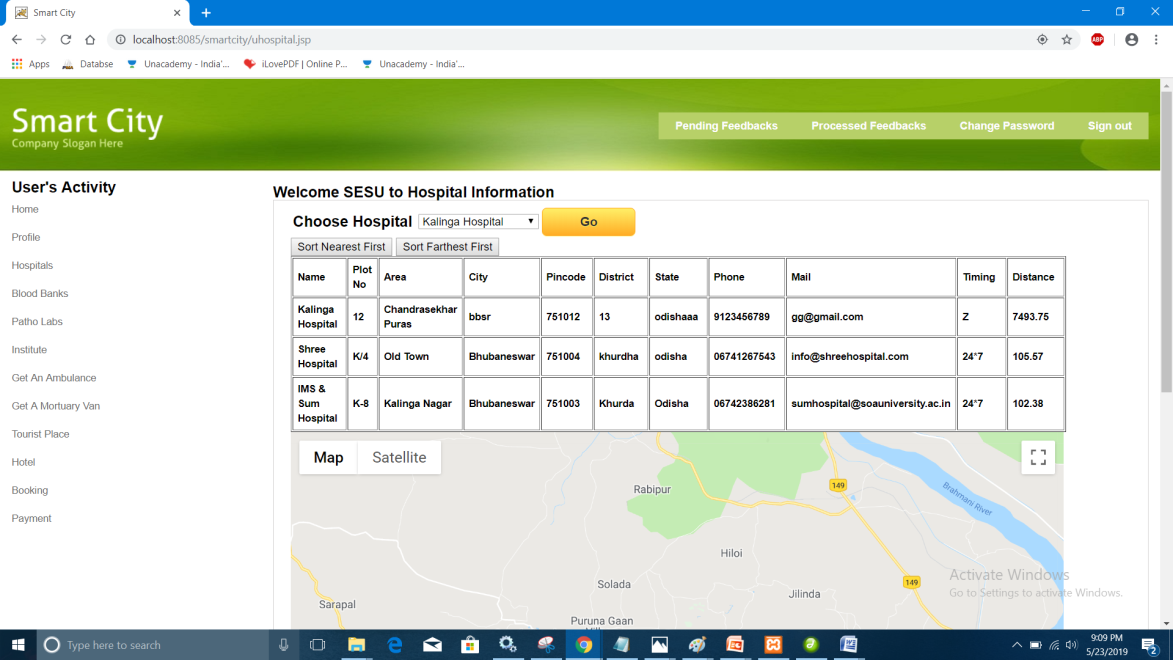
**SCREENSHOT OF INPUT-OUTPUT DESIGN**

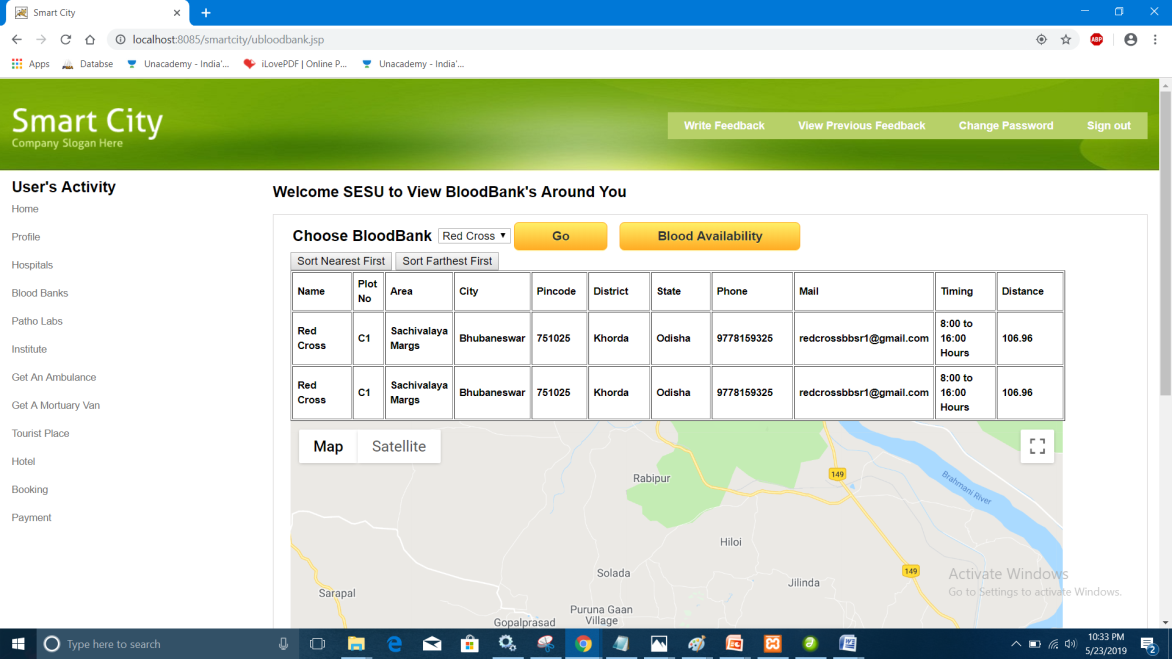


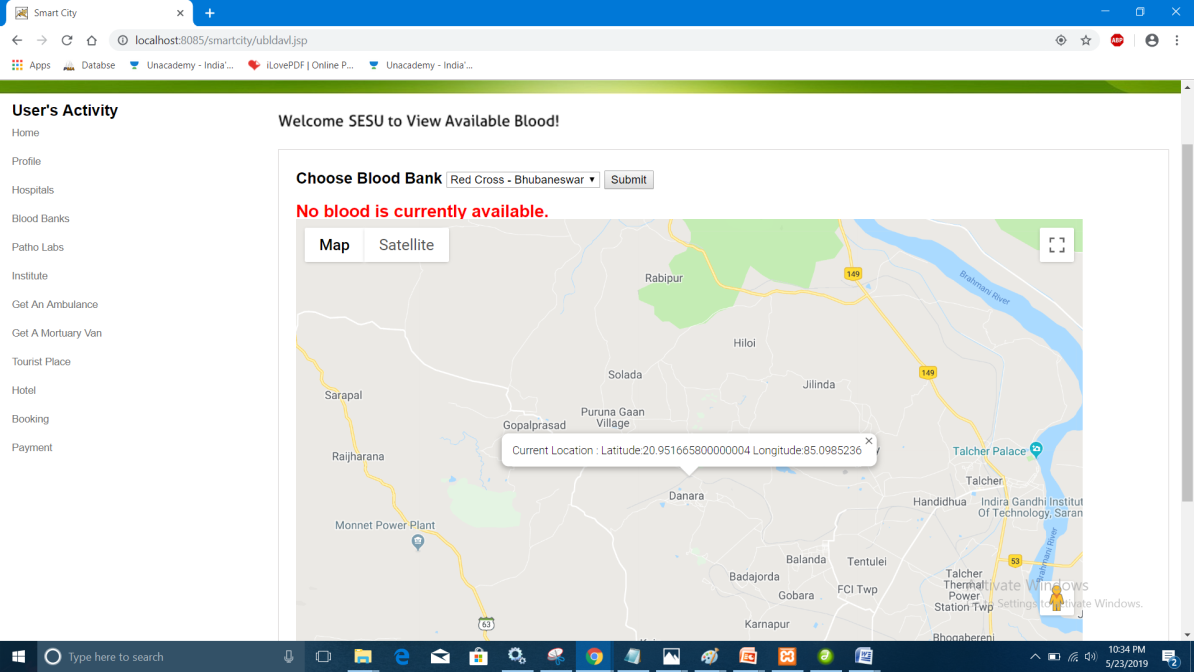


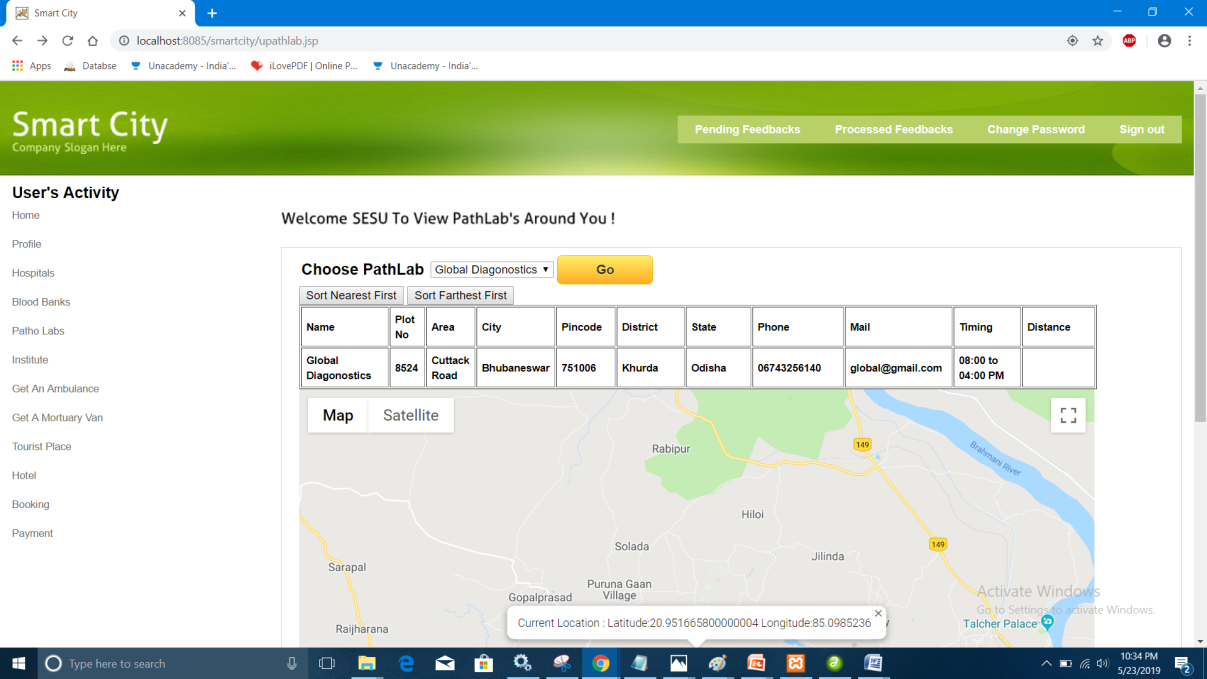


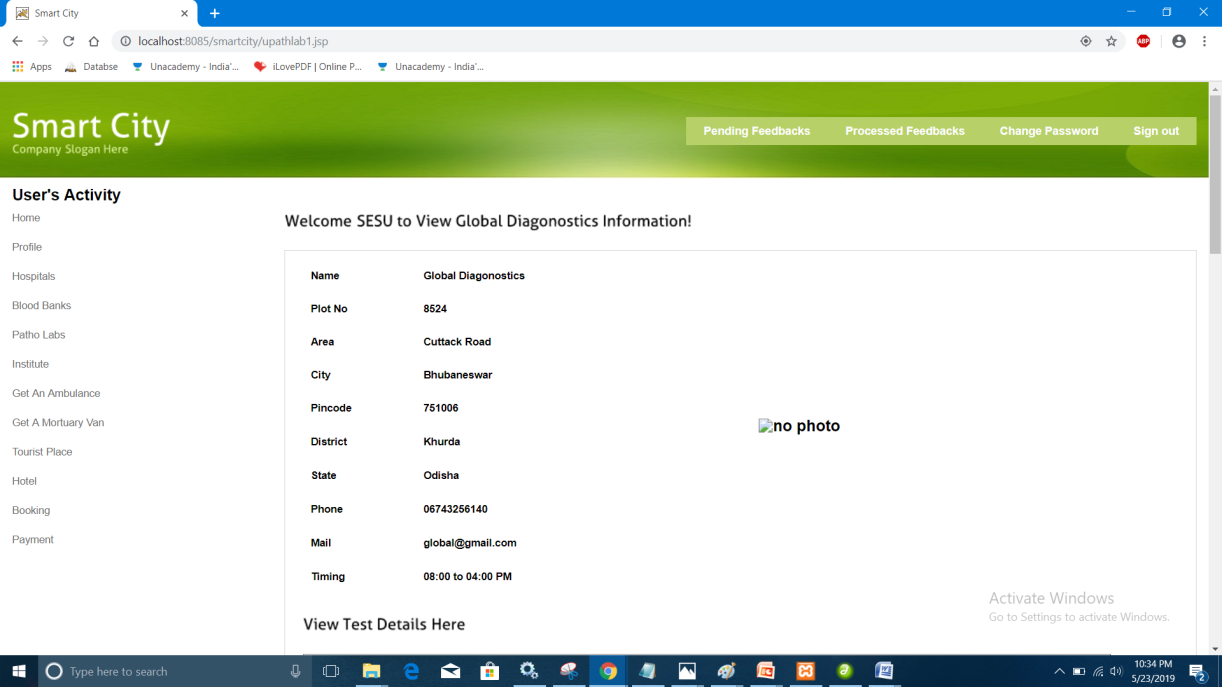


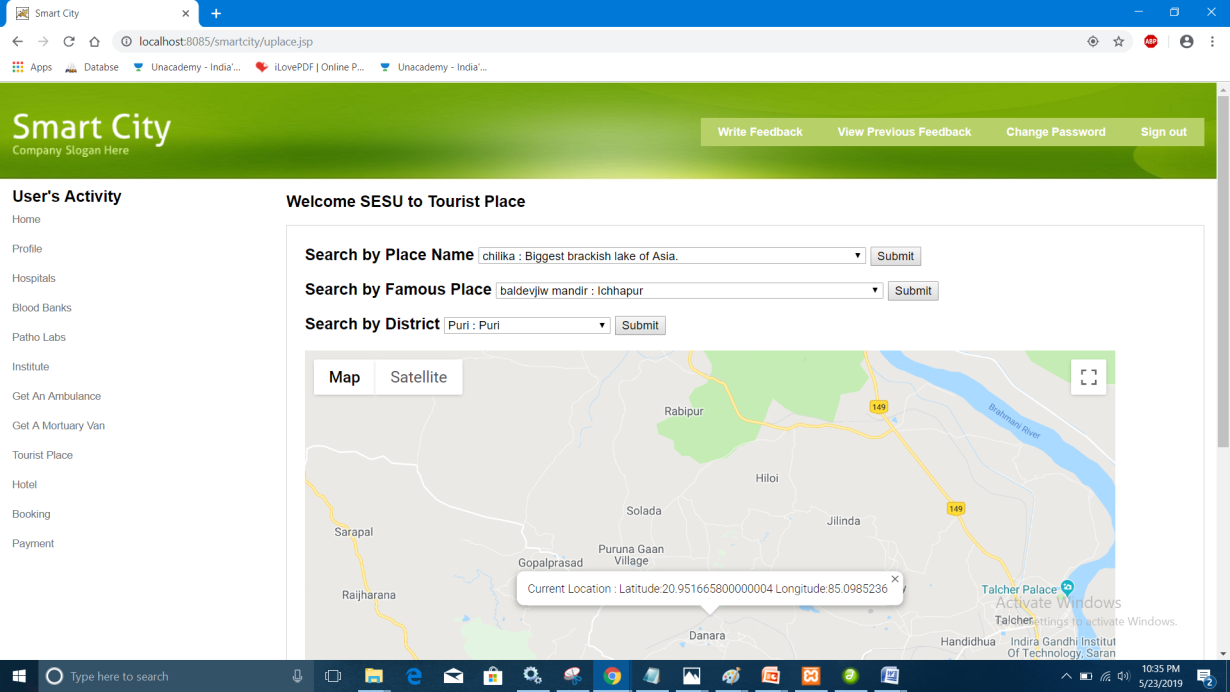


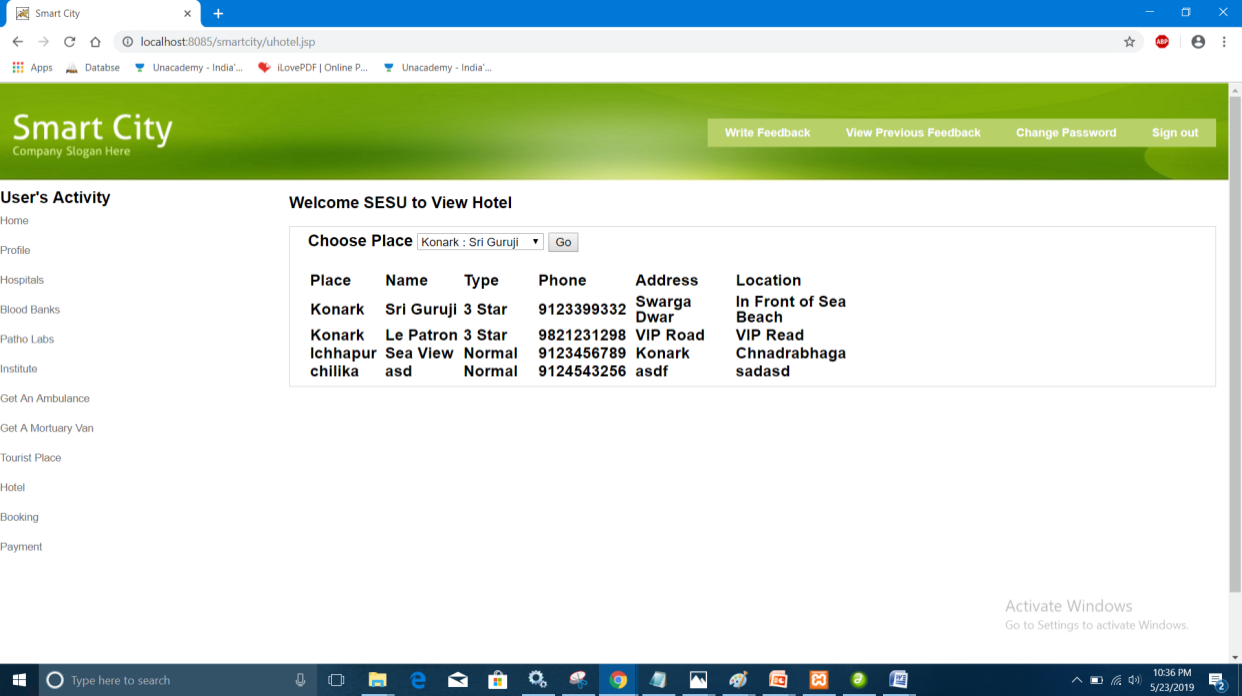
****

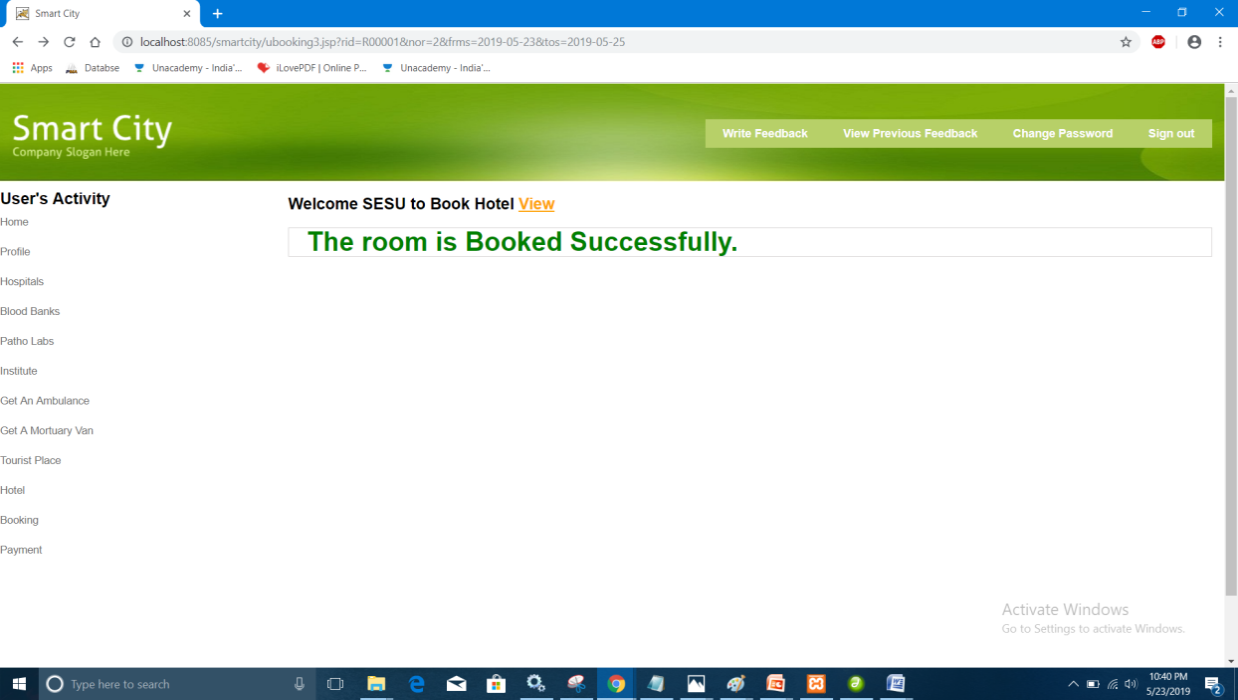
****

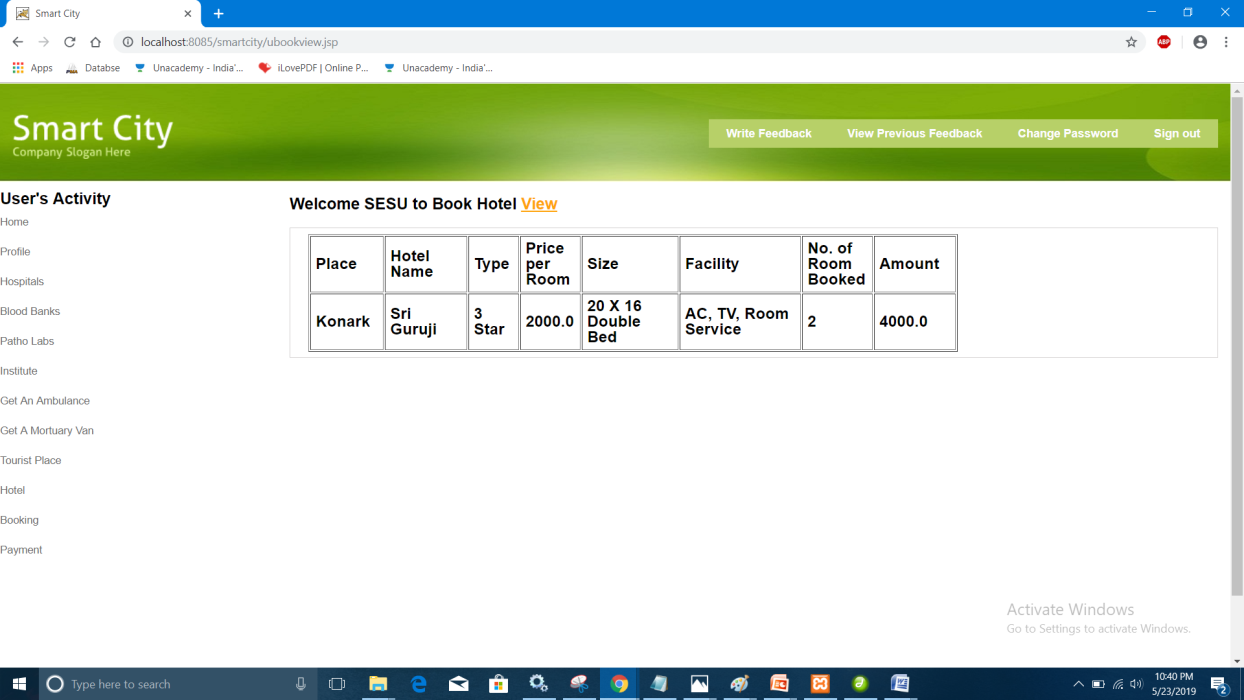
****

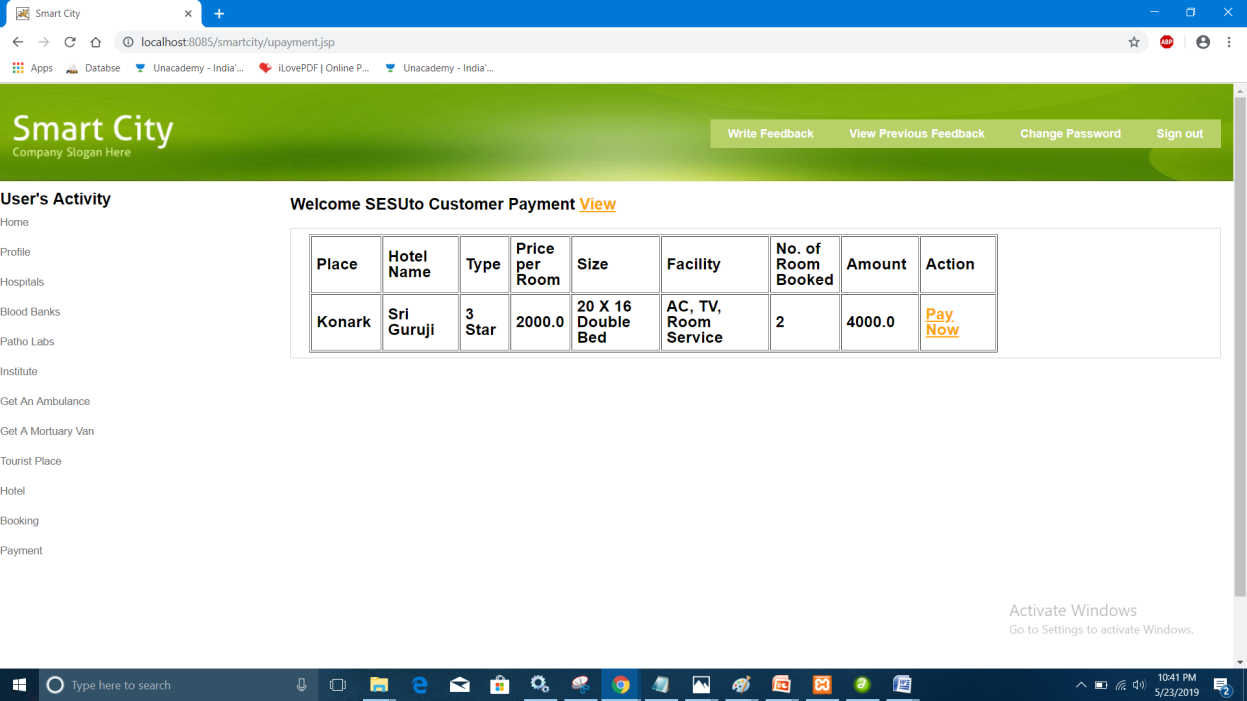
****

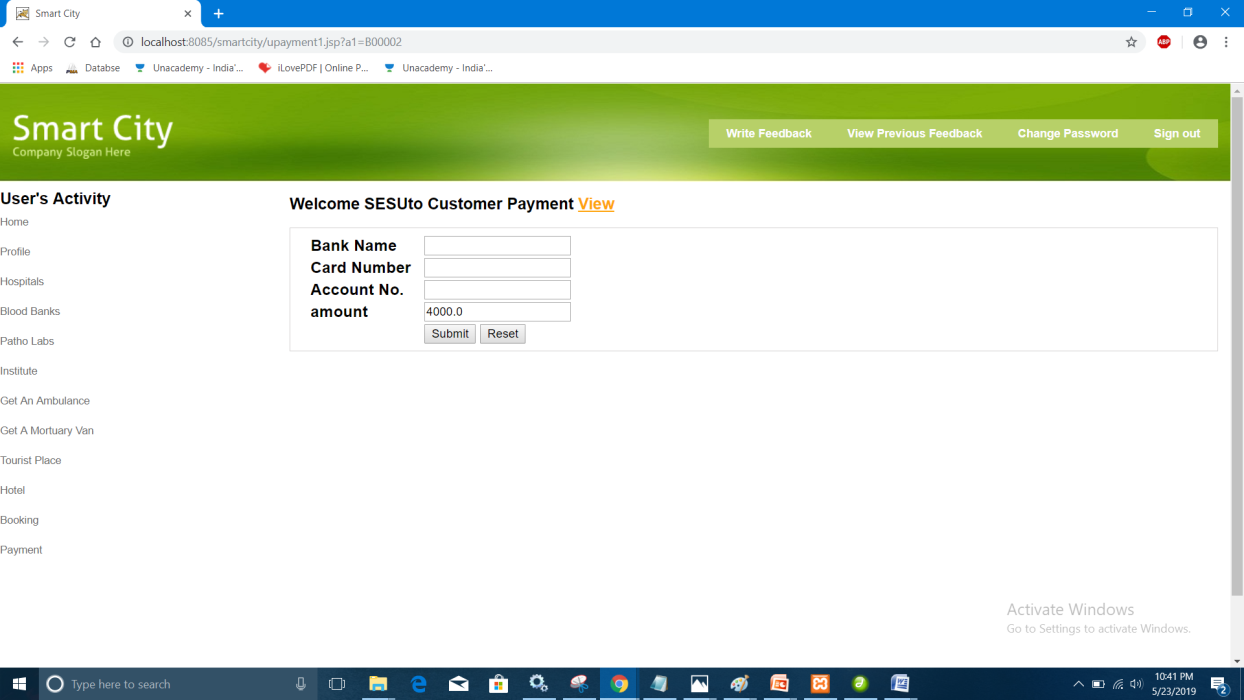
****

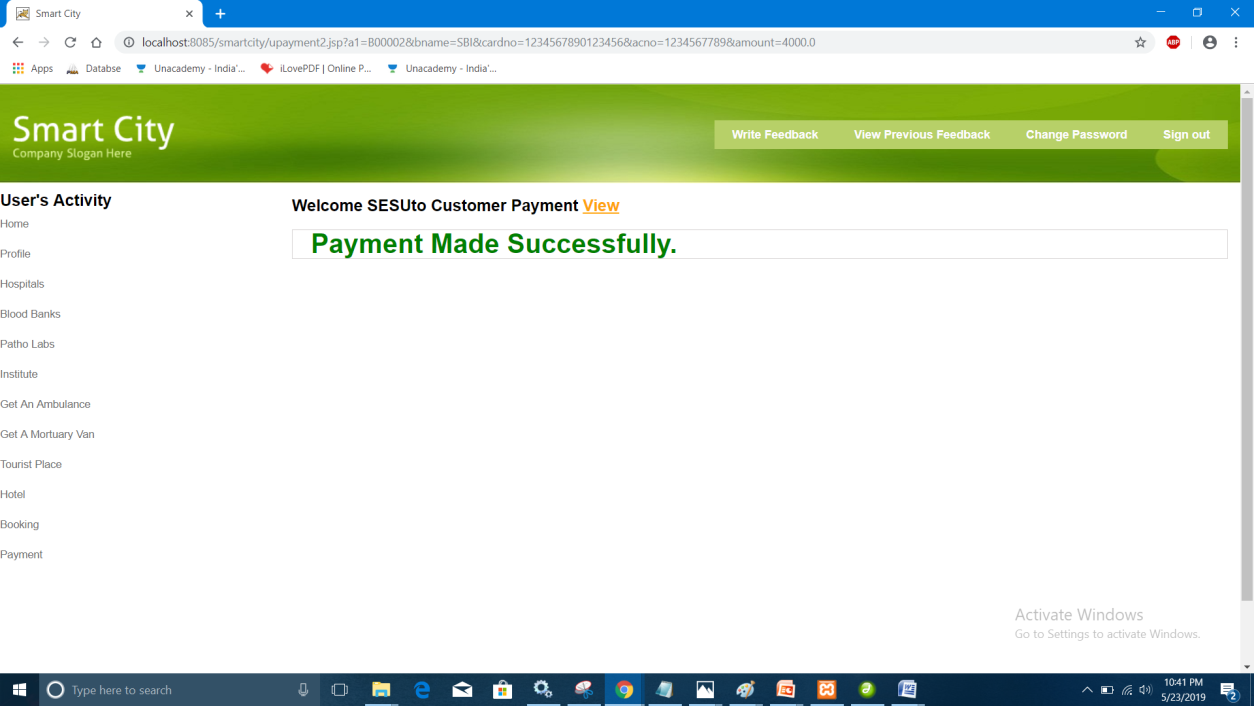
****

****

****

****

****

****

**SOURCE CODE**

Index.jsp

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<title>Smart City</title>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<link href="css/style.css" rel="stylesheet" type="text/css" />

<link rel="stylesheet" type="text/css" href="css/coin-slider.css" />

<script type="text/javascript" src="js/cufon-yui.js"></script>

<script type="text/javascript" src="js/cufon-aller.js"></script>

<script type="text/javascript" src="js/jquery-1.4.2.min.js"></script>

<script type="text/javascript" src="js/script.js"></script>

<script type="text/javascript" src="js/coin-slider.min.js"></script>

<style type="text/css">

<!--

.style3 {

font-size: large;

color: #990000;

}

.style4 {

font-size: large;

color: #0000FF;

}

.style6 {

color: #000000;

font-weight: bold;

}

-->

</style>

</head>

<body>

<div class="main">

<div class="header">

<div class="header\_resize">

<div class="logo">

<h1><a href="support.jsp">SMART CITY</a></h1>

</div>

<div class="menu\_nav">

<jsp:include page="topmenu.jsp" />

</div>

<div class="clr"></div>

<div class="slider">

<div id="coin-slider">

<img src="images/image.jpg" width="1280" height="322" alt="" />

</div>

<div class="clr"></div>

</div>

<div class="clr"></div>

</div>

</div>

<div class="content">

<div class="content\_resize">

<div class="templatemo\_rcol\_sectionwithborder">

<div id="sidebar">

<table width="1121">

<tr><td width="371">

<form name="f1" method="get" action="login.jsp">

<table width="327">

<tr><td width="170"><span class="style3">Sign In</span>.. &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp; <a href="userreg.jsp" class="style4">Sign Up</a></td>

</tr>

<tr><td ><input type="text" name="t1" size="25" placeholder="Enter User Name" /></td></tr>

<tr><td ><input type="password" name="p1" size="25" placeholder="Enter Password" /></td></tr>

<tr><td > <input type="submit" value="Submit" /> &nbsp;&nbsp;&nbsp; <input type="reset" value="Reset" /></td></tr>

<tr><td ></td></tr>

<tr><td><a href="forget.jsp">forget password</a></td></tr>

</table>

</form>

</td>

<td width="7"></td>

<td width="312">

<span class="style3">Organisation Sign Up </span>

<form name="f2" method="get" action="organsignup.jsp">

<table>

<tr>

<td><span class="style6">Type of Organisation</span></td>

<td>

<select name="o">

<option value="Hospital">Hospital</option>

<option value="Bloodbank">Bloodbank</option>

<option value="Path Lab">Path Lab</option>

<option value="Burn Unit">Burn Unit</option>

<option value="Trauma Center">Trauma Center</option>

</select> </td>

</tr>

<tr><td></td><td><input type="submit" value="Submit" />

</td></tr>

</table>

</form>

</td>

<td width="13"></td>

<td width="394">

<span class="style3">Ambulance/ Mortuary Van Sign Up</span>

<form name="f3" method="get" action="vehiclesignup.jsp">

<table>

<tr>

<td><span class="style6">Type of Organisation</span></td>

<td>

<select name="o">

<option value="Ambulance">Ambulance</option>

<option value="Mortuary Van">Mortuary Van</option>

</select> </td>

</tr>

<tr><td></td><td><input type="submit" value="Submit" />

</td></tr>

</table>

</form>

</td></tr>

</table>

<div class="clr"></div>

</div></div>

</div>

</div>

<div class="fbg">

<div class="fbg\_resize">

<div class="col c1">

<h2><span>Image</span> Gallery</h2>

<a href="#"><img src="images/gal1.jpg" width="75" height="75" alt="" class="gal" /></a> <a href="#"><img src="images/gal2.jpg" width="75" height="75" alt="" class="gal" /></a> <a href="#"><img src="images/gal3.jpg" width="75" height="75" alt="" class="gal" /></a> <a href="#"><img src="images/gal4.jpg" width="75" height="75" alt="" class="gal" /></a> <a href="#"><img src="images/gal5.jpg" width="75" height="75" alt="" class="gal" /></a> <a href="#"><img src="images/gal6.jpg" width="75" height="75" alt="" class="gal" /></a> </div>

<div class="col c2">

<h2><span>Services</span> Overview</h2>

<p>Curabitur sed urna id nunc pulvinar semper. Nunc sit amet tortor sit amet lacus sagittis posuere cursus vitae nunc.Etiam venenatis, turpis at eleifend porta, nisl nulla bibendum justo.</p>

<ul class="fbg\_ul">

<li><a href="#">Lorem ipsum dolor labore et dolore.</a></li>

<li><a href="#">Excepteur officia deserunt.</a></li>

<li><a href="#">Integer tellus ipsum tempor sed.</a></li>

</ul>

</div>

<div class="col c3">

<h2><span>Contact</span> Us</h2>

<p>Nullam quam lorem, tristique non vestibulum nec, consectetur in risus. Aliquam a quam vel leo gravida gravida eu porttitor dui.</p>

<p class="contact\_info"> <span>Address:</span>Odisha University Of Agriculture And Technology , BBSR<br />

<span>Telephone:</span> +123-1234-5678<br />

<span>FAX:</span> +458-4578<br />

<span>Others:</span> +301 - 0125 - 01258<br />

<span>E-mail:</span> <a href="#">mail@yoursitename.com</a> </p>

</div>

<div class="clr"></div>

</div>

</div>

<div class="footer">

<jsp:include page="footer.jsp" />

</div>

</div>

</body>

</html>

Login.jsp

<%@ include file="dbcon.jsp" %>

<%! String un="", ty="", ud=""; %>

<%

String s1=request.getParameter("t1");

String s2=request.getParameter("p1");

try{

ud="";

ty="";

un="";

ResultSet rs=st.executeQuery("select \* from admin where logname='"+s1+"' and pwd='"+s2+"'");

if(rs.next()){

ud=rs.getString(2);

ty="Admin";

un=s1;

}else {

rs=st.executeQuery("select \* from login where logname='"+s1+"' and pwd='"+s2+"' and status='Y'");

if(rs.next()){

ud=rs.getString(1);

ty=rs.getString(4);

un=s1;

}else

ty="";

}

con.close();

if( !ty.equals("")){

session.removeAttribute("uid");

session.removeAttribute("type");

session.removeAttribute("name");

session.setAttribute("uid", ud);

session.setAttribute("type", ty);

session.setAttribute("name", s1);

}

}catch(Exception e){ out.print("error : "+ e); }

//out.print(ty);

if(ty.equalsIgnoreCase("admin")){ %>

<jsp:forward page="adminhome.jsp" />

<% }else if(ty.equalsIgnoreCase("member")){ %>

<jsp:forward page="membhome.jsp" />

<% }else if(ty.equalsIgnoreCase("user")){ %>

<jsp:forward page="userhome.jsp" />

<% }else { %>

<jsp:forward page="index.jsp" >

<jsp:param name="a1" value="error" />

</jsp:forward>

<% } %>

Ahotel.jsp

<%@ include file="dbcon.jsp" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<title>Smart City</title>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<link href="css/style1.css" rel="stylesheet" type="text/css" />

<link rel="stylesheet" type="text/css" href="css/coin-slider.css" />

<script type="text/javascript" src="js/cufon-yui.js"></script>

<script type="text/javascript" src="js/cufon-aller.js"></script>

<script type="text/javascript" src="js/jquery-1.4.2.min.js"></script>

<script type="text/javascript" src="js/script.js"></script>

<script type="text/javascript" src="js/coin-slider.min.js"></script>

<style type="text/css">

<!--

.style1 {

font-size: large;

font-weight: bold;

color: #000000;

}

-->

</style>

</head>

<body>

<div class="main">

<div class="header">

<div class="header\_resize">

<div class="logo">

<h1><a href="about.jsp">Smart City <small>Company Slogan Here</small></a></h1>

</div>

<div class="menu\_nav">

<jsp:include page="admintop.jsp" />

</div>

<div class="clr"></div>

<div class="clr"></div>

</div>

</div>

<div class="content">

<div class="content\_resize">

<div class="mainbar style1">

<br />

Welcome <%=((String) session.getAttribute("name")).toUpperCase() %> to View Hotel

<div class="article">

</div>

<h5>

<%

try{

ResultSet rs= st.executeQuery("select \* from hotel a, place b where a.pid=b.pid");

%>

<table width="937">

<tr style="border:double #6666FF">

<td width="97" height="25">Place</td>

<td width="122">Name</td>

<td width="116">Type</td>

<td width="151">Phone</td>

<td width="156">Address</td>

<td width="198">Location</td>

<td width="65">Action</td>

</tr>

<% while(rs.next()) { %>

<tr style="border:double #6666FF">

<td><%=rs.getString(9)%></td>

<td><%=rs.getString(4)%></td>

<td><%=rs.getString(3)%></td>

<td><%=rs.getString(5)%></td>

<td><%=rs.getString(6)%></td>

<td><%=rs.getString(7)%></td>

<td><a href="ahotel1.jsp?a1=<%=rs.getString(1)%>" class="style1"><font color="green">Edit</font></a></td>

</tr>

<% } %>

</table>

<%

con.close();

}catch(Exception e){ out.print(e); }

%>

</h5>

</div>

<div class="sidebar">

<div class="gadget">

<jsp:include page="adminleft.jsp" />

</div>

</div>

<br /><br /><br /><br /><br /><br /><br /><br />

<br /><br /><br /><br /><br /><br /><br /><br />

<br /><br /><br /><br />

<div class="clr"></div>

</div>

</div>

</div>

<div class="footer">

<jsp:include page="footer.jsp" />

</div>

</div>

</body>

</html>

Aambula.jsp

<%@ include file="dbcon.jsp"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<title>Smart City</title>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<link href="css/style1.css" rel="stylesheet" type="text/css" />

<link rel="stylesheet" type="text/css" href="css/coin-slider.css" />

<script type="text/javascript" src="js/cufon-yui.js"></script>

<script type="text/javascript" src="js/cufon-aller.js"></script>

<script type="text/javascript" src="js/jquery-1.4.2.min.js"></script>

<script type="text/javascript" src="js/script.js"></script>

<script type="text/javascript" src="js/coin-slider.min.js"></script>

<style type="text/css">

<!--

.style1 {

font-size: large;

font-weight: bold;

color: #000000;

}

-->

</style>

<link rel="stylesheet" type="text/css" href="tabcontent.css" />

<meta charset='utf-8'>

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="stylesheet" href="styles.css">

<script type="text/javascript" src="tabcontent.js"></script>

<style>

.myButton {

-moz-box-shadow: 0px 1px 0px 0px #fff6af;

-webkit-box-shadow: 0px 1px 0px 0px #fff6af;

box-shadow: 0px 1px 0px 0px #fff6af;

background:-webkit-gradient(linear, left top, left bottom, color-stop(0.05, #ffec64), color-stop(1, #ffab23));

background:-moz-linear-gradient(top, #ffec64 5%, #ffab23 100%);

background:-webkit-linear-gradient(top, #ffec64 5%, #ffab23 100%);

background:-o-linear-gradient(top, #ffec64 5%, #ffab23 100%);

background:-ms-linear-gradient(top, #ffec64 5%, #ffab23 100%);

background:linear-gradient(to bottom, #ffec64 5%, #ffab23 100%);

filter:progid:DXImageTransform.Microsoft.gradient(startColorstr='#ffec64', endColorstr='#ffab23',GradientType=0);

background-color:#ffec64;

-moz-border-radius:6px;

-webkit-border-radius:6px;

border-radius:6px;

border:1px solid #ffaa22;

display:inline-block;

cursor:pointer;

color:#333333;

font-family:Arial;

font-size:15px;

font-weight:bold;

padding:7px 44px;

text-decoration:none;

text-shadow:0px 1px 0px #ffee66;

}

.myButton:hover {

background:-webkit-gradient(linear, left top, left bottom, color-stop(0.05, #ffab23), color-stop(1, #ffec64));

background:-moz-linear-gradient(top, #ffab23 5%, #ffec64 100%);

background:-webkit-linear-gradient(top, #ffab23 5%, #ffec64 100%);

background:-o-linear-gradient(top, #ffab23 5%, #ffec64 100%);

background:-ms-linear-gradient(top, #ffab23 5%, #ffec64 100%);

background:linear-gradient(to bottom, #ffab23 5%, #ffec64 100%);

filter:progid:DXImageTransform.Microsoft.gradient(startColorstr='#ffab23', endColorstr='#ffec64',GradientType=0);

background-color:#ffab23;

}

.myButton:active {

position:relative;

top:1px;

}

#map {

width: 920px;

height: 600px;

}

.style12 {font-size: 12px}

</style>

<script type="text/javascript" src='js/jquery-1.10.0.min.js'></script>

</head>

<body>

<div class="main">

<div class="header">

<div class="header\_resize">

<div class="logo">

<h1><a href="about.jsp">Smart City <small>Company Slogan Here</small></a></h1>

</div>

<div class="menu\_nav">

<jsp:include page="topmenu.jsp" />

</div>

<div class="clr"></div>

<div class="clr"></div>

</div>

</div>

<div class="content">

<div class="content\_resize">

<div class="mainbar style1">

<br />

<h4> Welcome <%=((String) session.getAttribute("name")).toUpperCase() %> to View Ambulance's Around You !</h4> <div class="article">

<form method="post" action="ambula1.jsp">

<%

try{

ResultSet rs=st.executeQuery("select a.\* from vehicle a,login b where a.vid=b.id and b.status='y' and a.availability='Y' and a.type='Ambulance'");

%>

<table>

<tr><td><span class="style12">Choose Ambulance</span></td>

<td></td>

<td><span class="style12">

<select name="em">

<% while(rs.next()){%>

<option value="<%=rs.getString(1)%>"><%=rs.getString(2)+" "%></option>

<%}%>

</select>

</span> </td><td></td><td></td><td></td><td></td>

<td><input type="submit" class="myButton" value="Book Ambulance"></td><td></td><td></td><td></td>

</tr>

</table>

<button class="sort-table asc">Sort Nearest First</button>

<button class="sort-table desc">Sort Farthest First</button>

<%}catch(Exception ee){}%>

</form>

<%

try{

ResultSet rs=st.executeQuery("select a .\* from vehicle a,login b where a.vid=b.id and b.status='y' and a.type='Ambulance' and a.availability='Y' limit 15");

%>

<form method="post" name="latilongiform" action="">

<table width="923" height="65" border="1" cellpadding="5" id="loctable">

<thead>

<tr>

<td width="110"><span class="style12">Vehicle Number</span></td>

<td width="30"><span class="style12">Model</span></td>

<td width="40"><span class="style12">Owner</span></td>

<td width="40"><span class="style12">Driver Name</span></td>

<td width="68"><span class="style12">Booking Number</span></td>

<td width="99"><span class="style12">City</span></td>

<td width="94"><span class="style12">District</span></td>

<td width="120"><span class="style12">State</span></td>

<td width="120"><span class="style12">Timing</span></td>

<td width="100"><span class="style12">Base Price</span></td>

<td width="100"><span class="style12">Per K.m Price</span></td>

<td width="100"><span class="style12">Distance</span></td>

</tr>

</thead>

<tbody>

<%

int i=0;

while(rs.next()){ %>

<tr>

<td height="29">

<span class="style12">

<input type="hidden" name="lati" id='lati\_<%=i%>' value="<%=rs.getString("lati")%>"/>

<input type="hidden" name="longi" id='longi\_<%=i%>' value="<%=rs.getString("longi")%>"/>

<%=rs.getString(2)%></span></td>

<td><span class="style12"><%=rs.getString(3)%></span></td>

<td><span class="style12"><%=rs.getString(4)%></span></td>

<td><span class="style12"><%=rs.getString(5)%></span></td>

<td><span class="style12"><%=rs.getString(6)%></span></td>

<td><span class="style12"><%=rs.getString(7)%></span></td>

<td><span class="style12"><%=rs.getString(8)%></span></td>

<td><span class="style12"><%=rs.getString(9)%></span></td>

<td><span class="style12"><%=rs.getString(10)%></span></td>

<td><span class="style12"><%=rs.getString(11)%></span></td>

<td><span class="style12"><%=rs.getString(12)%></span></td>

<td id="distance\_<%=i%>"></td>

</tr>

<%

i++;

}

%>

</tbody>

</table>

</form>

<%

con.close();

}catch(Exception ee) { }

%>

<div id="map"></div>

<script>

var currentPositionLat=20.2860587;

var currentPositionLng=85.8345398;

var latis = document.latilongiform.lati;

var longis = document.latilongiform.longi;

function initMap() {

var baseLatLong = {lat: 20.2860587, lng: 85.8345398};

var map = new google.maps.Map(document.getElementById('map'), {

zoom: 12,

center: baseLatLong

});

/\* START :: AJAX call for Saved Location \*/

$.ajax({

url : 'get-saved-locations.jsp',

data:{type:'Hospital'},

dataType: 'json',

success : function(json) {

//console.log(JSON.stringify(json));

$(json).each(function(key,value){

var savedLatLong = new google.maps.LatLng(value.lat, value.lng)

//console.log(savedLatLong);

//console.log(baseLatLong);

var contentString = '<div id="content">'+

'<div id="siteNotice">'+

'</div>'+

'<h1 id="firstHeading" class="firstHeading">'+value.name+'</h1>'+

'<div id="bodyContent">'+

'<p><b>Latitude: '+value.lat+'</b></p>' +

'<p><b>Longitude: '+value.lng+'</b></p>' +

'</div>'+

'</div>';

var infowindow = new google.maps.InfoWindow({

content: contentString

});

var marker = new google.maps.Marker({

position: savedLatLong,

map: map,

title: value.name

});

marker.addListener('click', function() {

infowindow.open(map, marker);

});

});

}

});

/\* END :: AJAX call for Saved Location \*/

// START :: Current Location

var infoWindow1 = new google.maps.InfoWindow({map: map});

// Try HTML5 geolocation.

if (navigator.geolocation) {

navigator.geolocation.getCurrentPosition(function(position) {

var pos = {

lat: position.coords.latitude,

lng: position.coords.longitude

};

var currentPosition = new google.maps.LatLng(pos.lat, pos.lng);

//var otherPoint = new google.maps.LatLng(baseLatLong.lat, baseLatLong.lng);

currentPositionLat=currentPosition.lat();

currentPositionLng=currentPosition.lng();

//var distance = calcDistance(currentPosition , otherPoint);

if(latis!=null)

{

for(var i=0;i<latis.length;i++)

{

var lati = latis[i].value;

var longi = longis[i].value;

var otherPoint = new google.maps.LatLng(lati, longi);

var distance = calcDistance(currentPosition , otherPoint);

//document.getElementById("distance\_"+i).innerHTML

$("#distance\_"+i).html(distance);

}

}

sortTable();

//alert(calcDistance(currentPosition , otherPoint));

console.log(pos);

infoWindow1.setPosition(pos);

infoWindow1.setContent('Current Location : Latitude:'+pos.lat + " Longitude:"+ pos.lng);

map.setCenter(pos);

}, function() {

handleLocationError(true, infoWindow1, map.getCenter());

});

} else {

// Browser doesn't support Geolocation

handleLocationError(false, infoWindow1, map.getCenter());

}

//END :: Current Location

}

function placeMarker(map, location) {

var marker = new google.maps.Marker({

position: location,

map: map

});

//console.log(location);

var contentData = '<div id="content">'+

'<div id="siteNotice">'+

'</div>'+

'<h1 id="firstHeading" class="firstHeading">Location</h1>'+

'<div id="bodyContent">'+

'<p><b>Latitude: '+location.lat()+'</b></p>' +

'<p><b>Longitude: '+location.lng()+'</b></p>' +

'</div>'+

'</div>';

var infowindow2 = new google.maps.InfoWindow({

content: contentData

});

marker.addListener('click', function() {

infowindow2.open(map, marker);

});

}

//Function for Current Location

function handleLocationError(browserHasGeolocation, infoWindow, pos) {

infoWindow.setPosition(pos);

infoWindow.setContent(browserHasGeolocation ?

'Error: The Geolocation service failed.' :

'Error: Your browser doesn\'t support geolocation.');

}

//calculates distance between two points in km's

function calcDistance(p1, p2){

return (google.maps.geometry.spherical.computeDistanceBetween(p1, p2) / 1000).toFixed(2);

}

/\*

var rad = function(x) {

return x \* Math.PI / 180;

};

var getDistance = function(p1, p2) {

var R = 6378137; // Earth’s mean radius in meter

var dLat = rad(p2.lat() - p1.lat());

var dLong = rad(p2.lng() - p1.lng());

var a = Math.sin(dLat / 2) \* Math.sin(dLat / 2) +

Math.cos(rad(p1.lat())) \* Math.cos(rad(p2.lat())) \*

Math.sin(dLong / 2) \* Math.sin(dLong / 2);

var c = 2 \* Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));

var d = R \* c;

return d; // returns the distance in meter

};\*/

function sortTable(){

$table = $('#loctable'), // cache the target table DOM element

$rows = $('tbody > tr', $table); // cache all rows from the target table body

$rows.sort(function(a, b) {

var keyA = $('td',a).text();

var keyB = $('td',b).text();

//var keyA = $(a).children('td').eq(10).text();

//var keyB = $(b).children('td').eq(10).text();

return (keyA < keyB) ? 1 : 0; // A bigger than B, sorting ascending

});

$rows.each(function(index, row){

$table.append(row); // append rows after sort

});

/\* //var rows = $('#loctable tbody tr').get();

var $tbody = $('#loctable tbody');

var rows = $tbody.find('tr');

rows.sort(function(a, b) {

var A = $(a).children('td').eq(10).text();

var B = $(b).children('td').eq(10).text();

console.log(A);

console.log(B);

if(A < B) {

return -1;

}

if(A > B) {

return 1;

}

return 0;

});

$.each(rows, function(index, row) {

$('#loctable').children('tbody').append(row);

});\*/

}

$(function(){

$('.sort-table').click(function(e) {

e.preventDefault(); // prevent default button click behaviour

var sortAsc = $(this).hasClass('asc'), // ASC or DESC

$table = $('#loctable'), // cache the target table DOM element

$rows = $('tbody > tr', $table); // cache all rows from the target table body

$rows.sort(function(a, b) {

var keyA = $('td',a).text();

var keyB = $('td',b).text();

if (sortAsc) {

return (keyA > keyB) ? 1 : 0; // A bigger than B, sorting ascending

} else {

return (keyA < keyB) ? 1 : 0; // B bigger than A, sorting descending

}

});

$rows.each(function(index, row){

$table.append(row); // append rows after sort

});

});

});

</script>

<script async defer

src="https://maps.googleapis.com/maps/api/js?key=AIzaSyDBRwuvgr8Thn-kajeeHeMwr6speR549AI&libraries=geometry&callback=initMap">

</script>

</div>

</div>

<div class="sidebar">

<div class="gadget">

</div>

</div>

<br /><br /><br /><br /><br /><br /><br /><br />

<br /><br /><br /><br /><br /><br /><br /><br />

<br /><br /><br /><br />

<div class="clr"></div>

</div>

</div>

</div>

<div class="footer">

<jsp:include page="footer.jsp" />

</div>

</div>

</body></html>

Aplace.jsp

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<title>Smart City</title>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<link href="css/style1.css" rel="stylesheet" type="text/css" />

<link rel="stylesheet" type="text/css" href="css/coin-slider.css" />

<script type="text/javascript" src="js/cufon-yui.js"></script>

<script type="text/javascript" src="js/cufon-aller.js"></script>

<script type="text/javascript" src="js/jquery-1.4.2.min.js"></script>

<script type="text/javascript" src="js/script.js"></script>

<script type="text/javascript" src="js/coin-slider.min.js"></script>

<style type="text/css">

<!--

.style1 {

font-size: large;

font-weight: bold;

color: #000000;

}

.style5 {font-size: 14px}

-->

</style>

</head>

<body>

<div class="main">

<div class="header">

<div class="header\_resize">

<div class="logo">

<h1><a href="about.jsp">Smart City <small>Company Slogan Here</small></a></h1>

</div>

<div class="menu\_nav">

<jsp:include page="admintop.jsp" />

</div>

<div class="clr"></div>

<div class="clr"></div>

</div>

</div>

<div class="content">

<div class="content\_resize">

<div class="mainbar style1">

<br />

Welcome <%=((String) session.getAttribute("name")).toUpperCase() %> to Place Entry <a href="aplaceview.jsp">View</a> <br />

<div class="article">

<form action="aplace1.jsp" method="post">

<table>

<tr><td width="278"><span class="style8 style5">Place Name</span></td>

<td width="225"><input type="text" name="place" required /></td></tr>

<tr><td><span class="style8 style5">Area(City Name or nearest City Name)</span></td>

<td><input type="text" name="area" required /></td></tr>

<tr><td><span class="style8 style5">District</span></td>

<td><input type="text" name="dist" required /></td></tr>

<tr><td><span class="style8 style5">State</span></td>

<td><input type="text" name="state" required /></td></tr>

<tr><td><span class="style8 style5">URL</span></td>

<td><input type="text" name="url" required /></td></tr>

<tr><td><span class="style8 style5">Famous for</span></td>

<td><textarea name="famfor" rows="6" cols="60" required></textarea></td></tr>

<tr><td><span class="style8 style5">Description(Within 1000 letters)</span></td>

<td><textarea name="descr" rows="6" cols="60" required></textarea></td></tr>

<tr><td></td>

<td><input type="submit" value="Submit" /> <input type="reset" value="Reset" /></td></tr>

</table>

</form>

</div>

</div>

<div class="sidebar">

<div class="gadget">

<jsp:include page="adminleft.jsp" />

</div>

</div>

<br /><br /><br /><br /><br /><br /><br /><br />

<br /><br /><br /><br /><br /><br /><br /><br />

<br /><br /><br /><br />

<div class="clr"></div>

</div>

</div>

</div>

<div class="footer">

<jsp:include page="footer.jsp" />

</div>

</div>

</body>

</html>

eblood.jsp

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<title>Smart City</title>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<link href="css/style1.css" rel="stylesheet" type="text/css" />

<link rel="stylesheet" type="text/css" href="css/coin-slider.css" />

<script type="text/javascript" src="js/cufon-yui.js"></script>

<script type="text/javascript" src="js/cufon-aller.js"></script>

<script type="text/javascript" src="js/jquery-1.4.2.min.js"></script>

<script type="text/javascript" src="js/script.js"></script>

<script type="text/javascript" src="js/coin-slider.min.js"></script>

<style type="text/css">

<!--

.style1 {

font-size: large;

font-weight: bold;

color: #000000;

}

.style2 {font-size: 14}

-->

</style>

</head>

<body>

<div class="main">

<div class="header">

<div class="header\_resize">

<div class="logo">

<h1><a href="about.jsp">Smart City <small>Company Slogan Here</small></a></h1>

</div>

<div class="menu\_nav">

<jsp:include page="emptop.jsp" />

</div>

<div class="clr"></div>

<div class="clr"></div>

</div>

</div>

<div class="content">

<div class="content\_resize">

<div class="mainbar style1">

<br />

Welcome <%=((String) session.getAttribute("name")).toUpperCase() %> to Blood Bank Details View ! <div class="article">

<table>

<tr>

<td width="153" height="34"><a href="ebldavl.jsp" class="myButton style2">Blood Availability</a></td>

<td width="211"><a href="ebldreq.jsp" class="myButton style2">Pending Blood Donation</a></td>

<td width="243"><a href="ebldreq1.jsp" class="myButton style2">Blood Donation History</a></td>

</tr>

</table>

</div>

</div>

<div class="sidebar">

<div class="gadget">

<jsp:include page="empleft.jsp" />

</div>

</div>

<br /><br /><br /><br /><br /><br /><br /><br />

<br /><br /><br /><br /><br /><br /><br /><br />

<br /><br /><br /><br />

<div class="clr"></div>

</div>

</div>

</div>

<div class="footer">

<jsp:include page="footer.jsp" />

</div>

</div>

</body>

</html>

**CONCLUSION**

Smart City people can get a host of services at their fingertip after implementation of this application by all organizations. It would keep people updated with the information and would specially keep them aware of the prices charged to them for the services. Organisations can also get a platform through this application to enhance their business. Smart technology is Smart Living which the Smart City Deserves.

# **REFERENCES:**

## Books Referred

1. H.M.DIETEL.P.J.DIETEL, Java How to Program, PhL, Second Edition
2. Gray Cornett, Horstmann, CoreJava, Sunsoft Press, 1996.
3. Patrick Naughton & Herbert Schildt, Java : The Complete Reference, Tata McGraw-Hill, Macrch 1997.
4. Grady Booch, Object Oriented Analysis and Design with Applications, the Benjimin/Cummings, 1994.

## Web Sites Referred

<http://java.sun.com>

<https://developers.google.com>

\*\*\*\*\*\*