MINI PROJECT REPORT
ON
ONLINE PARKING RESERVATION

Submitted by

ABHISHEK JAISWAL
(171500008)
KALEEM ULLAH SIDDIQUI
(171500151)
KARTIK SRIVASTAVA
(171500156)
SANDEEP KUMAR RAI
(171500288)

Department of Computer Engineering & Applications
Institute of Engineering & Technology

GLA University
Mathura- 281406, INDIA

We hereby declare that the work which is being presented in the Mid Term Report "ONLINE PARKING RESERVATION using JAVA in partial fulfillment of the requirements for Mini Project is an authentic record of my team work carried under the supervision of Mandeep Singh(Technical Trainer),GLA UNIVERSITY.

Name of Team Members:

Abhishek Jaiswal (171500008)

Kaleem Ullah Siddiqui(171500151)

Kartik Srivastava(171500156)

Sandeep Kumar Rai(171500288)

Course: B.Tech

Year: 3rd

Semester: 5th

SYNOPSIS

Student Information:

Name: Abhishek Jaiswal Kaleem Ullah Siddiqui Kartik Srivastava Sandeep Kumar Rai	University Roll No. 171500008 171500151 171500156 171500288
Mobile: 8173041954 7897186526 9565030271 9598164671	Email: abhishek.jaiswal_cs17@gla.ac.in kaleem.siddiqui_cs17@gla.ac.in kartik.srivastava_cs17@gla.ac.in sandeep.rai_cs17@gla.ac.in

Information about Industry/Organization:

Industry/Organization Name with full Address	GLA University,Mathura Uttar Pradesh ,281406
Contact Person	Name & Designation : Mandeep Singh(Technical Trainer) Email : mandeep.singh@gla.ac.in

Project Information:

Title Of Project/Training/Task	Online Parking Reservation Using Java
Role & Responsibility	
Technical Details	Hardware Requirements: Laptop Software Requirements: JAVA , Mysql
Training Implementation Details	Partial Implemented

Summary of the Training Work:

Online parking reservation that is designed to make it easier for people to book parking spaces online. Our online parking reservation system to reserve parking spaces in the immediate parking, additional services will increase our website by enabling customers to go online and reserve their favourite space to park their car. As they need, and to set the period of availability can add many types of vehicle seats as administrator. It is designed to make it easier for people to book parking spaces online.

Availability and prices can add up for a period of several vehicle types as vehicle parking space reservation system administrators as they need. In today parking lots there are no standard system to check for parking spaces. The proposed project is a smart parking booking system that provides institute students an easy way of reserving a parking space online using web portal.

It overcomes the problem of finding a parking space in universities/institute areas that unnecessary consume time.

ACKNOWLEDGEMENT

It gives me a great sense of pleasure to present the report of the Industrial Training undertaken before B. Tech. Third Year. This training in itself is an acknowledgement to the inspiration, drive and technical assistance contributed to it by many individuals. This project would never have seen the light of the day without the help and guidance that we have received.

Our heartiest thanks to Mandeep Singh,Technical Trainer, Department of CEA, GLA University for providing me with an encouraging platform to develop this project, which thus helped me in shaping our abilities towards a constructive goal.

We owe special debt of gratitude Mandeep Singh, for his constant support and guidance throughout the course of our work. His sincerity, thoroughness and perseverance have been a constant source of inspiration for me. He has showered me with all his extensively experienced ideas and insightful comments at virtually all stages of the project & has also taught us about the latest industry-oriented technologies.

I also do not like to miss the opportunity to acknowledge the contribution of all authorities of the organization for their kind guidance and cooperation during the development of our project.

ABSTRACT

Online Car Parking Reservation System is designed to make it easier for people to book parking spaces online. Our online parking reservation system will instantly enhance your website by enabling customers to reserve parking spaces, buy extra services and pay online from home or on the go. Car Park Booking System admins can add as many car space types as they need, set availability periods and prices, launch promos and discounts.

Contents

INTRODUCTION

This Web Application mainly deals with the parking slots in the buildings like clubs, hotels, malls and many more.

In this Web Application we can access the information of parking slots in the building where is free. By finding the empty space the user is able to block the slot.

This application also provides information about the user like Car No, License No. and mobile number the administration will able to notify the user if there is any problem.

After selecting the empty space in the parking slot the user is able to pay the amount and confirm his/her booking.

LIST OF MODULES

(i)Admin/ Guard Module

(ii)User Module

SOFTWARE ENVIRONMENT

Java Technology

Java technology is both a programming language and a platform.

The Java Programming Language

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.

You can think of Java byte codes as the machine code instructions for the Java Virtual Machine (JVM). Every Java interpreter, whether it's a development tool or a Web browser that can run applets, is an implementation of the JVM. 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 JVM. That means that as long as a computer has a JVM, the same program written in the Java programming language can run on Windows 2000, a Solaris workstation, or on an iMac.

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 next section, What Can Java Technology Do? Highlights what functionality some of the packages in the Java API provide.

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.

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.

How does the API support all these kinds of programs? It does so with packages of software components that provides a wide range of functionality.

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 (JDBC™): 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.

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 Java™ 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.

- Distribute software more easily: You can upgrade applets easily from a central server. Applets take advantage of the feature of allowing new classes to be loaded "on the fly," without recompiling the entire program.

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. There is a 16-bit and a 32-bit version of this program and each

maintains a separate list of ODBC data sources.

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.

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 goals that were set for JDBC are important. They will give you some insight as to why certain classes and functionalities behave the way they do. The eight 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 MS Access database.

Java ha 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

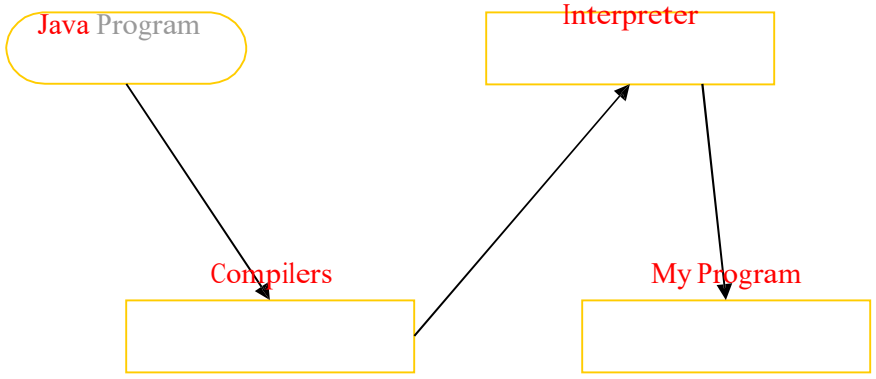
Interpreted multithreaded

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.



You can think of Java byte codes as the machine code instructions for the Java Virtual Machine (JVM). Every Java interpreter, whether it's a Java development tool or a Web browser that can run Java applets, is an implementation of the JVM. The JVM can also be implemented in hardware.

Java byte codes help make "write once, run anywhere" possible. You can compile your Java program into byte codes on my platform that has a Java compiler. The byte codes can then be run any implementation of the Java VM. For example, the same Java program can run Windows NT, Solaris, and Macintosh.

Networking

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

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.

JFree Chart

JFreeChart is a free 100% Java chart library that makes it easy for developers to display professional quality charts in their applications. JFreeChart's extensive feature set includes:

A consistent and well-documented API, supporting a wide range of chart types;

A flexible design that is easy to extend, and targets both server-side and client-side applications;

Support for many output types, including Swing components, image files (including PNG and JPEG), and vector graphics file formats (including PDF, EPS and SVG);

JFreeChart is "open source" or, more specifically, free software. It is distributed under the terms of the GNU Lesser General Public Licence (LGPL), which permits use in proprietary applications.

1. Map Visualizations

Charts showing values that relate to geographical areas. Some examples include: (a) population density in each state of the United States, (b) income per capita for each country in Europe, (c) life expectancy in each country of the world. The tasks in this project include:

Sourcing freely redistributable vector outlines for the countries of the world, states/provinces in particular countries (USA in particular, but also other areas);

Creating an appropriate dataset interface (plus default implementation), a rendered, and integrating this with the existing XYPlot class in JFreeChart;

Testing, documenting, testing some more, documenting some more.

2. Time Series Chart Interactivity

Implement a new (to JFreeChart) feature for interactive time series charts --- to display a separate control that shows a small version of ALL the time series data, with a sliding "view" rectangle that allows you to select the subset of the time series data to display in the main chart.

3. Dashboards

There is currently a lot of interest in dashboard displays. Create a flexible dashboard mechanism that supports a subset of JFreeChart chart types (dials, pies, thermometers, bars, and lines/time series) that can be delivered easily via both Java Web Start and an applet.

4. Property Editors

The property editor mechanism in JFreeChart only handles a small subset of the properties that can be set for charts. Extend (or reimplement) this mechanism to provide greater end-user control over the appearance of the charts.

J2ME (Java 2 Micro edition):-

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

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 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 The following graphic depicts the relationship between the different virtual machines, configurations, and profiles. It also draws a parallel with the J2SE API and its Java virtual machine. While the J2SE virtual machine is generally referred to as a JVM, the J2ME virtual machines, KVM and CVM, are subsets of JVM. Both KVM and CVM can be thought of as a kind of Java virtual machine -- it's just that they are shrunken versions of the J2SE JVM and are specific to J2ME.

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 preverification 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:

- Keep it simple. Remove unnecessary features, possibly making those features a separate, secondary application.

- 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. You should manage the memory efficiently yourself by setting object references to null when you are finished with them. Another way to reduce run-time memory is to use lazy instantiation, only allocating objects on an as-needed basis. Other ways of reducing overall and peak memory use on small devices are to release resources quickly, reuse objects, and avoid exceptions.

4. Configurations 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 16-bit 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 KJava 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.

A skeleton profile upon which you can create your own profile, the Foundation Profile, is available for CDC.

Profile 1: KJava

KJava is Sun's proprietary profile and contains the KJava API. The KJava profile is built on top of the CLDC configuration. The KJava virtual machine, KVM, accepts the same byte codes and class file format as the classic J2SE virtual machine. KJava contains a Sun-specific API that runs on the Palm OS. The KJava API has a great deal in common with the J2SE Abstract Windowing Toolkit (AWT). However, because it is not a standard J2ME package, its main package is com.sun.kjava. We'll learn more about the KJava API later in this tutorial when we develop some sample applications.

Profile 2: MIDP

MIDP is geared toward mobile devices such as cellular phones and pagers. The MIDP, like KJava, is built upon CLDC and provides a standard run-time environment that allows new applications and services to be deployed dynamically on end user devices. MIDP is a common, industry-standard profile for mobile

devices that is not dependent on a specific vendor. It is a complete and supported foundation for mobile application

development. MIDP contains the following packages, the first three of which are core CLDC packages, plus three MIDP-specific packages.

* java.lang

* java.io

* java.util

* javax.microedition.io

* javax.microedition.lcdui

* javax.microedition.midlet

* javax.microedition.rms

6. SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the

Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

TYPES OF TESTS

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive

processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

6.1 Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

6.2 Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

6.3 Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

SYSTEM TESTING

TESTING METHODOLOGIES

The following are the Testing Methodologies:

- Unit Testing.
- Integration Testing.
- User Acceptance Testing.
- Output Testing.
- 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 tests each module individually and then each module is module is integrated with a main module and tested for functionality.

OTHER TESTING METHODOLOGIES

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.

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.

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 messages. 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.

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.

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.

System Design:

UML Diagrams

Unified Modeling Language (UML) is a standard language for writing software blueprints. UML can be used for visualizing, specifying, constructing, documenting the artifacts of a software-intensive system. The Unified Modeling Language will result in lower overall costs, more reliable and efficient software, and a better relationship with all parties involved. Software documented with UML can be modified much more efficiently.

UML is a notation that resulted from the unification of Object Modeling Technique and Object Oriented Software Technology .UML has been designed for broad range of application. Hence, it provides constructs for a broad range of systems and activities.

1. Use case diagrams

Use cases are used during requirements elicitation and analysis to represent the functionality of the system. Use cases focus on the behavior of the system from the external point of view. The actors are outside the boundary of the system, whereas the use cases are inside the boundary of the system.

2. Class diagrams

Class diagrams to describe the structure of the system. Classes are abstractions that specify the common structure and behavior of a set Class diagrams describe the system in terms of objects, classes, attributes, operations and their associations.

3. Sequence diagrams

Sequence diagrams are used to formalize the behavior of the system and to visualize the communication among objects. They are useful for identifying additional objects that participate in the use cases. A Sequence diagram represents the interaction that take place among these objects.

4. State Chart diagrams

State chart diagrams describe the behavior of an individual object as a number of states and transitions between these states. A state represents a particular set of values for an object. The sequence diagram focuses on the messages exchanged between objects, the state chart diagrams focuses on the transition between states.

5. Activity diagrams

An activity diagram describes a system in terms of activities. Activities are statesthat represents the execution of a set of operations. Activity diagrams are similar to flowchart diagram and data flow

Use case Diagram

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral 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.

Sequence diagram

UML sequence diagrams model the flow of logic within your system in a visual manner, enabling you both to document and validate your logic, and are commonly used for both analysis and design purposes. Sequence diagrams are the most popular UML artifact for dynamic modeling, which focuses on identifying the behavior within your system.

The following sequences of steps are involved in the system.

- The user has to login the website .
- If the user has login valid then he can process the data.
- Generate Sessions
- Apply clustering on generated sessions data sets .
- Finally we result from the clustering data .

Figure 4.6 Sequence Diagram

4.6.6Activity diagram

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

Figure 4.9 Activity Diagram

CLASS DIAGRAM;

Testing:

SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the

Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

TYPES OF TESTS

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process

performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot "see" into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.

- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

SYSTEM TESTING

TESTING METHODOLOGIES

The following are the Testing Methodologies:

- Unit Testing.
- Integration Testing.
- User Acceptance Testing.
- Output Testing.
- 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 tests each module individually and then each module is module is integrated with a main module and tested for functionality.

OTHER TESTING METHODOLOGIES

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.

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.

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 messages. 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.

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.

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.

CODE:

Accept.jsp

```
<%@page import="java.sql.*"%>
<%@page import="databaseconnection.databaseconn"%>
<%@ page session="true" %>
<%@page import="Mail.Mail"%>
    <%
          String sno = request.getParameter("slatno");
              int sno1 = Integer.parseInt(sno);
    try {
           Connection con1 = databaseconn.getconnection();



            String msg = "Sir/Madam slot no: "+sno1+" is allocated for You";

                String email=request.getParameter("email");
                    String vehicleno=request.getParameter("vehicleno");
                //String str = "Booked";
                Statement st1 = con1.createStatement();
                String query1 ="update vehicletable set status='Booked' where
vehicleno='"+vehicleno+"' ";
                st1.executeUpdate (query1);



                     Mail m= new Mail();
                    m.secretMail(msg, email, email);
                    System.out.println("Done.....................");

                 //   String query2 ="update vehicletable set verify='mail sent' where
vehicleno='"+vehicleno+"' ";
                //st1.executeUpdate (query2);
                    con1.close();
                response.sendRedirect("viewrequest.jsp?m1=success");
```

```
}
        catch(Exception e)
        {
            out.println(e.getMessage());
        }

    %>
```

Adminhome.jsp

```
<%@page import="java.sql.*"%>
<%@page import="databaseconnection.databaseconn"%>
<%@ page session="true" %>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Vehicle Parking</title>
    <style>
    .home-bg {
    background: white url("img/v1.jpg") no-repeat scroll;
    overflow: hidden;
    color: whitesmoke;

    background-size: cover;
}
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
  overflow: hidden;
  background-color: #333;
}
```

```
li {
  float: left;
}

li a {
  display: block;
  color: white;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
}

li a:hover:not(.active) {
  background-color: #111;
}

.active {
  background-color:
    #4CAF50;



}
</style>
   </head>


  <body class="home-bg">
     <header class="main-header">
        <h1 style="text-align:center;"><i style="color:blueviolet">Vehicle Parking
System</i></h1>
   </header>
<ul>
  <li><a class="active" href="adminhome.jsp">Home</a></li>
  <li><a class="" href="viewusers.jsp">View_Users</a></li>
  <li><a class="" href="viewrequest.jsp">View_Booking_Request</a></li>
  <li><a class="" href="adminlogin.jsp">Logout</a></li>
</ul>
```

```
</body>
</html>
```

Adminlogin.jsp

```
<!DOCTYPE html>

<html>
   <head>
      <title>Vehicle Parking</title>
      <meta charset="UTF-8">
      <meta name="viewport" content="width=device-width, initial-scale=1.0">
      <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
integrity="sha384-
ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZ
w1T" crossorigin="anonymous">
      <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
integrity="sha384-
q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo
" crossorigin="anonymous">
      </script>
   <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"
integrity="sha384-
UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHND
z0W1" crossorigin="anonymous">
  </script>
  <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"
integrity="sha384-
JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM"
crossorigin="anonymous">
  </script>
```

```
  </head>
    <style>
        .frm{
            background-color: activecaption;
        }
         .b2 {
             background-image:url("img/v9.jpg");
             background-repeat: no-repeat;
             background-size: 100% 500%;

         }
         .login {
    text-align: center;
    width: 400px;
    box-sizing: border-box;
    padding: 30px;
    background: lightseagreen;
    border-radius: 0 80px;
}
.s1{
  font-size: 150%;
}

    </style>
    <body class="b2">
      <nav class="navbar navbar-dark bg-dark">
       <ul class="nav nav-pills">
          <li class="nav-item">
             <a class="nav-link active" href="index.html">Home</a>
          </li>
          <li class="nav-item">
             <a class="nav-link" href="adminlogin.jsp">Adminlogin</a>
          </li>
          <li class="nav-item">
       </ul>
   <form class="form-inline my-2 my-lg-0">
```

```
  <input class="form-control mr-sm-2" type="search" placeholder="Search" aria-
label="Search">
     <button class="btn btn-outline-success my-2 my-sm-0"
type="submit">Search</button>


   </form>
 <!-- Navbar content -->
</nav>
   <br><br>
   <center class="s1">
      <div class="login">
      <h1 style="color:lightcoral"> Authorized One!</h1>
         <i style="color:brown">
      <br>
```

```
  <form action="adminloginact.jsp" method="post">
   <div style="width:300px" class="frm">
     <div  class="form-group">
      <label for="exampleDropdownFormEmail2">Username</label>
        <input type="text" name="username" class="form-control"
id="exampleDropdownFormEmail2" placeholder="Username">
    </div>
  <div class="form-group">
    <label for="exampleDropdownFormPassword2">Password</label>
    <input type="password" name="password" class="form-control"
id="exampleDropdownFormPassword2" placeholder="Password">
  </div>
  <div class="form-group">

     <input type="checkbox" class="form-check-input" id="dropdownCheck2">
     <label class="form-check-label" for="dropdownCheck2">
       Remember me
     </label>
    </div>
  </div>


  <button type="submit" class="btn btn-primary">Sign in</button><br>
 </div>
</center>
       </body>
      <br><br>
```

```
        <footer style="background-color: black">
            <center><a href="">Developed by:GLAU </a></center>
</footer>
</html>
```

Adminloginact.jsp

```
<%
    String username = request.getParameter("username");
    String password = request.getParameter("password");


    try{

          if ((username.equals("admin") ) && (password.equals("admin")))
```

Online Parking Reservation Software Envionment

```
    {

                response.sendRedirect("adminhome.jsp?msg=Success");
        }
       else
        {
           response.sendRedirect("adminlogin.jsp?msg1=LoginFail");
        }
     }
    catch(Exception e)
    {
       System.out.println("Error in userlogin.jsp"+e.getMessage());
    }
%>
```

Forgetact.jsp

```
<!DOCTYPE html>

<html>
   <head>
      <title>Vehicle Parking</title>
      <meta charset="UTF-8">
      <meta name="viewport" content="width=device-width, initial-scale=1.0">
      <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
integrity="sha384-
ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZ
w1T" crossorigin="anonymous">
      <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
integrity="sha384-
q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo
" crossorigin="anonymous">
      </script>
   <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"
integrity="sha384-
UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHND
z0W1" crossorigin="anonymous">
  </script>
  <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"
integrity="sha384-
JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM"
crossorigin="anonymous">
  </script>

   </head>
   <style>
      .frm{
         background-color: activecaption;
      }
       .b2 {
```

```
      background-image:url("img/v9.jpg");
            background-repeat: no-repeat;
            background-size: 100% 500%;
           }
         .login {
    text-align: center;
    width: 400px;
    box-sizing: border-box;
    padding: 40px;
    background: rgb(84, 86, 99);
    border-radius: 0 110px;
}
.s1{
  font-size: 150%;
}
 </style>
    <body class="b2">
        <header style="background-color:lightgrey"><h1 style="text-align:
center;padding: 30px;"><i>Vehicle Parking<i></h1></header>
       <nav class="navbar navbar-dark bg-dark">
        <ul class="nav nav-pills">
           <li class="nav-item">
              <a class="nav-link active" href="index.html">Home</a>
           </li>
           <li class="nav-item">
        </ul>
   <form class="form-inline my-2 my-lg-0">
      <input class="form-control mr-sm-2" type="search" placeholder="Search" aria-
label="Search">
      <button class="btn btn-outline-success my-2 my-sm-0"
type="submit">Search</button>
    </form>
 <!-- Navbar content -->
</nav>
    <br><br>
```

```
<center class="s1">


<div class="login">
<h1 style="color:lightcoral"> Forget Password</h1>
<i style="color:brown">
<br>
<form action="forgetpass.jsp" method="post">
<div style="width:300px" class="frm">

<div class="form-group">
<label for="exampleDropdownFormemail">Enter Email id:</label>



<input type="email" name="email" class="form-control"
id="exampleDropdownFormPassword2" placeholder="Enter email id">
</div>

</div>


<button type="submit" class="btn btn-primary">Submit</button><br>

</div>
</center>
</body>
</html>
```

Forgetpass.jsp

```
<%@page import="Mail.Mail"%>
<%@page import="java.sql.*"%>
<%@page import="databaseconnection.databaseconn"%>
<%@ page session="true" %>
<%
String umail = request.getParameter("email");
```

```
try{

      //String msg = "mail sent succesfull";
 Connection con = databaseconn.getconnection();
   Statement st = con.createStatement();
   String str = "SELECT password FROM user where email='"+umail+"'";
  ResultSet rs= st.executeQuery(str);
  //String password= rs.getString("password");
  if(rs.next())
   {
     // String msg = "password is '"+password+"' succesfull";
        String password= rs.getString("password");
         String msg = "password is '"+password+"' succesfully";
           Mail m= new Mail();
                m.secretMail(msg, umail, umail);
            response.sendRedirect("userlogin.jsp?m1=success");
   }

 }
catch(Exception e)
{
   System.out.println("Error in userlogin.jsp"+e.getMessage());
}

%>
```

Reject.jsp

```
<%@page import="java.sql.*"%>
<%@page import="databaseconnection.databaseconn"%>
<%@page session="true"%>
<%@page import="Mail.Mail"%>
    <%

    try {
            String msg = "Sir/Madam Sorry slot is not Available";
                Connection con1 =databaseconn.getconnection();
               String email=request.getParameter("email");
                   String vehicleno=request.getParameter("vehicleno");



                   Mail m= new Mail();
                   m.secretMail(msg, email, email);


               con1.close();
               response.sendRedirect("viewrequest.jsp?m2=success");



        }
        catch(Exception e)
        {
             out.println(e.getMessage());
        }

    %>
```

Requestact.jsp

```
<%@page import="java.sql.*"%>
<%@page import="databaseconnection.databaseconn"%>
<%@page session="true"%>
```

```
<%
    String username=request.getParameter("uname");
    String vehicleno=request.getParameter("vehicleno");
    String type=request.getParameter("type");
    String date=request.getParameter("date");
     String umail=request.getParameter("umail");
    String status="pending";
    int slatno=0;


 try
    {
    Connection con=databaseconn.getconnection();
 //   Statement st=con.createStatement();
    PreparedStatement ps=con.prepareStatement("insert into vehicletable
values(?,?,?,?,?,?,?)");
    ps.setString(1,vehicleno);
    ps.setString(2,type);
    ps.setString(3,username);
    ps.setString(4,date);
    ps.setString(5,umail);
    ps.setString(6,status);
    ps.setInt(7,slatno);
    int i= ps.executeUpdate();
    if(i>0)
    {

     response.sendRedirect("userlogin.jsp?m2=Reguest_sended");
    }
    else
    {
      response.sendRedirect("usserhome.jsp?m3=Booking_failed");
    }
    }
     catch(Exception e)
            {
               out.println(e.getMessage());
            }
```

```
%>
```

Userhome.jsp

```
<%@page import="java.sql.*"%>
<%@page import="databaseconnection.databaseconn"%>
<%@ page session="true" %>
<!DOCTYPE html>

<html>
   <head>
      <title>Online Parking Reservation</title>
      <meta charset="UTF-8">
      <meta name="viewport" content="width=device-width, initial-scale=1.0">
      <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
integrity="sha384-
ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZ
w1T" crossorigin="anonymous">


 <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-
q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo
" crossorigin="anonymous">
      </script>
   <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"
integrity="sha384-
UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHND
z0W1" crossorigin="anonymous">
  </script>
  <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"
integrity="sha384-
JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM"
crossorigin="anonymous">
  </script>
```

```
</head>
  <%
          String uname= session.getAttribute("username").toString();
           %>
  <style>
     .cl{
        background: red;
     }

     .b2 {
         background-image:url("img/v9.jpg");
         background-repeat: no-repeat;
         background-size: 100% 500%;

     }
      .login {
   text-align: center;
   width: 600px;
   box-sizing: border-box;
  padding: 30px;
   background: lightseagreen;
   border-radius: 0 80px;
}
.s1{
 font-size: 150%;
}
   </style>
   <br>
    <body class="b2">


     <nav class="navbar navbar-dark bg-dark">
      <ul class="nav nav-pills">
        <li class="nav-item">
          <a class="nav-link active" href="userhome.html">Home</a>
```

```
  </li>

          <li class="nav-item">
             <a class="nav-link" href="userlogin.jsp">Logout</a>
          </li>
          <li class="nav-item">
        </ul>
   <form class="form-inline my-2 my-lg-0">
     <input class="form-control mr-sm-2" type="search" placeholder="Search" aria-
label="Search">
     <button class="btn btn-outline-success my-2 my-sm-0"
type="submit">Search</button>
    </form>
 <!-- Navbar content -->
</nav>

      <br><br>
       <center class="s1">
          <div class="login">
            <h1>Welcome User!</h1>
    <div class="container">
          <div class="container-fluid" style="width:900px">

          <div style="width:300px" class="c1">
                    <form action="requestact.jsp" method="post">
          <center>
      <table>
          <%
       String username= session.getAttribute("username").toString();
       %>
       <tr>
          <td>
              <label for="exampleDropdownFormPassword1">UserName:</label>
        </td>
        <td>
        <input type="text" name="uname" value="<%=username%>">
```

```
        </td>
        <tr>

         <tr>
          <div class="form-group" class="c1">
             <td>
             <label for="exampleDropdownFormPassword1">VehicleNo:</label>
             </td>
             <td>
                <input type="text" name="vehicleno" class="form-control"
id="exampleDropdownFormPassword1" placeholder="VehicleNo">
             </td>
```

```
 </div>
        </tr>
        <tr>
         <div class="form-group">
          <td>  <label
for="exampleDropdownFormEmail1">TypeOfthevehicle:</label></td>
          <td>
           <select name="type">
               <option value="select">--Select_vehicle--</option>
              <option value="2wheeler">Two_Wheeler</option>
              <option value="4wheeler">Four_Wheeler</option>
           </select>
          </td>
         </div>
          </tr>
          <tr>
             <div class="form-group">
             <td>  <label for="exampleDropdownFormEmail1">Select
Date:</label></td>
              <td>
               <input type="date" name="date" class="form-control"
id="exampleDropdownFormPassword1" placeholder="Username">
             </td>
              </div>
          </tr
          <tr>
             <div class="form-group">

    <td>  <label for="exampleDropdownFormEmail1">EmailId:</label></td><br>
              <td>
               <input type="email" name="umail" class="form-control"
id="exampleDropdownFormPassword1" placeholder="Usermail">
             </td>
              </div>
          </tr>
           </table><br>
          <button type="submit" class="btn btn-primary">Submit</button>
```

```
</form>
          </div>
       </div>
          </div>
        </div>

              </center>

       <br><br>
        <footer style="background-color: black">
          <marquee> <center><a href="">Developed by:GLAU
</a></center></marquee>
</footer>
        </body>
```

```
</html>
```

Userlogin.jsp

```
<!DOCTYPE html>

<html>
   <head>
      <title>Vehicle Parking</title>
      <meta charset="UTF-8">
      <meta name="viewport" content="width=device-width, initial-scale=1.0">
      <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
integrity="sha384-
ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZ
w1T" crossorigin="anonymous">
      <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
integrity="sha384-
q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo
" crossorigin="anonymous">
      </script>
   <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"
integrity="sha384-
UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHND
z0W1" crossorigin="anonymous">
  </script>
  <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"
integrity="sha384-
JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM"
crossorigin="anonymous">
  </script>

   </head>
   <style>
```

```
.b2 {
        background-image:url("img/v9.jpg");
        background-repeat: no-repeat;
        background-size: 100% 500%;

    }

.login {
   text-align: center;
   width: 400px;
   box-sizing: border-box;
    padding: 30px;
   background: lightseagreen;
   border-radius: 0 80px;
}
```

```
.s1{
  font-size: 150%;
}

    </style>
    <br>
     <body class="b2">

        <header style="background-color:"><h1 style="text-align: center;padding:
20px;"><i><marquee>Vehicle Parking</marquee><i></h1></header>
       <nav class="navbar navbar-dark bg-dark">
        <ul class="nav nav-pills">
           <li class="nav-item">
              <a class="nav-link active" href="index.html">Home</a>
           </li>
           <li class="nav-item">
              <a class="nav-link" href="userlogin.jsp">Userlogin</a>
           </li>
           <li class="nav-item">
        </ul>
   <form class="form-inline my-2 my-lg-0">
```

```
<input class="form-control mr-sm-2" type="search" placeholder="Search" aria-
label="Search">
     <button class="btn btn-outline-success my-2 my-sm-0"
type="submit">Search</button>
   </form>
 <!-- Navbar content -->
</nav>

       <center class="s1">
       <br><br>
       <div class="login">
          <h1><i> Already Member!</i></h1><br>

         <div class="container">
          <div class="container-fluid" style="width:900px">

          <div style="width:300px" class="frm">
             <form action="userloginact.jsp" method="post">
          <center>
          <div class="form-group">
             <label for="exampleDropdownFormPassword1">User Name</label>
             <input type="text" name="name" class="form-control"
id="exampleDropdownFormPassword1" placeholder="Username" required="">
          </div>
          <div class="form-group">
             <label for="exampleDropdownFormEmail1">Password</label>
             <input type="password" name="psw"class="form-control"
id="exampleDropdownFormEmail1" placeholder="Password">
```

```
    </div>

          <button type="submit" class="btn btn-primary">Sign in</button>     <a
href="forgetact.jsp" class="btn btn-
primary"><botton>Forgotpassword</button></a>
       </form>

       <a class="dropdown-item" href="userregistration.jsp">New customer here?
Sign up</a>

      </div>
         </div>
       </div>
  </center>
                <br>
                <br><br>
       <footer style="background-color: black">
          <center><a href="">Developed by:GLAU </a></center>
</footer>
       </body>
</html>
```

Userloginact.jsp

```
<%@page import="java.sql.*"%>
<%@page import="databaseconnection.databaseconn"%>
<%@ page session="true" %>
<%
   String username = request.getParameter("name");
   String password = request.getParameter("psw");


   try{


    Connection con = databaseconn.getconnection();
       Statement st = con.createStatement();
```

```
  ResultSet rs = st.executeQuery("select * from user where name ='"+username+"'
 and password='"+password+"'");
       if(rs.next())
        {
            session.setAttribute("username",username);
                response.sendRedirect("userhome.jsp?msg=Success");
        }
       else
        {
           response.sendRedirect("userlogin.jsp?msg1=LoginFail");
              }
     }t


   catch(Exception e)
    {
       System.out.println("Error in userlogin.jsp"+e.getMessage());
    }
%>
```

Userregistration.jsp

```
<%@page import="java.sql.*"%>
<%@page import="databaseconnection.databaseconn"%>
<%@ page session="true" %>
<%
    String username = request.getParameter("name");
    String password = request.getParameter("psw");


    try{

     Connection con = databaseconn.getconnection();
        Statement st = con.createStatement();
        ResultSet rs = st.executeQuery("select * from user where name
='"+username+"' and password='"+password+"'");
       if(rs.next())
        {
```

```
session.setAttribute("username",username);
                response.sendRedirect("userhome.jsp?msg=Success");
       }
       else
       {
          response.sendRedirect("userlogin.jsp?msg1=LoginFail");
             }
    }
    catch(Exception e)
    {
       System.out.println("Error in userlogin.jsp"+e.getMessage());
    }
%>
```

Userregistrationact.jsp

```
<%@page import="java.sql.*"%>
<%@page import="databaseconnection.databaseconn"%>
<%@page session="true"%>
<%
   String username=request.getParameter("username");
```

```
   String password=request.getParameter("password");
    String uno=request.getParameter("uniqueno");
    String email=request.getParameter("umail");
    String address=request.getParameter("address");
    String phoneno=request.getParameter("phoneno");
     try
    {
    Connection con=databaseconn.getconnection();
 //   Statement st=con.createStatement();
    PreparedStatement ps=con.prepareStatement("insert into user
values(?,?,?,?,?,?)");
    ps.setString(1,username);
    ps.setString(2,password);
    ps.setString(3,uno);
  ps.setString(4,phoneno);
```

```
    ps.setString(5,email);
      ps.setString(6,address);
    int i= ps.executeUpdate();
    if(i>0)
    {

     response.sendRedirect("userlogin.jsp?m2=Registred_sucessful");

    }
    else
    {
      response.sendRedirect("registration.jsp?m3=login_failed");
    }
    }
     catch(Exception e)
            {
               out.println(e.getMessage());
            }
%>
```

Viewrequest.jsp

```
<%@page import="java.sql.*"%>
<%@page import="databaseconnection.databaseconn"%>
<%@ page session="true" %>
<!DOCTYPE html>
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title>Vehicle Parking System</title>
```

```
<style>
    .s1{
  font-size: 150%;
}
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
  overflow: hidden;
  background-color: #333;
}

li {
  float: left;
}

li a {
  display: block;
  color: white;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
}
li a:hover:not(.active) {
  background-color: #111;
}
.active {
    background-color:
      #4CAF50;
}
.b1{
     background-image: url("img/s2.jpg");
     background-repeat: no-repeat;
     background-size:  cover;
}
</style>
    </head>
```

```
<body class="b1">
        <header class="main-header">
            <h1 style="text-align:center;"><i style="color:blueviolet">Vehicle Parking
  System</i></h1>
      </header>
  <ul>
    <li><a class="" href="adminhome.jsp">Home</a></li>
    <li><a class="active" href="viewrequest.jsp">View_BookingRequest</a></li>
     <li><a class="" href="adminlogin.jsp">Logout</a></li>
  </ul>
```

```
center class="s1">
        <h1 style="text-align: center"><u>User Booking_ Requests</u></h1>
  <t<able border="1" width="80%">
  <tr>

  <th>Vehicleno</th>
  <th>V_type</th>
  <th>Name</th>
  <th>Date</th>
  <th>Email</th>
  <th>Status</th>
  <th>Slotno</th>
  <th>Verify</th>


  </tr>

  <%
  try{
  Connection con= databaseconn.getconnection();
  Statement st=con.createStatement();
  String sql ="select * from vehicletable where status='pending'";
  ResultSet rs= st.executeQuery(sql);
  while(rs.next()){
  %>
   <tr>
  <td><%=rs.getString("vehicleno") %></td>
  <td><%=rs.getString("vehicletype") %></td>
  <td><%=rs.getString("username") %></td>
  <td><%=rs.getString("date") %></td>
  <td><%=rs.getString("email") %></td>
  <td><%=rs.getString("status") %></td>
  <td><%=rs.getString("slatno") %></td>
  <td><a
  href="accept.jsp?vehicleno=<%=rs.getString(1)%>&slatno=<%=rs.getString(7)%>
  &email=<%=rs.getString(5)%>">Accept</a></td>
```

```
<td><a
href="reject.jsp?email=<%=rs.getString(5)%>&vehicleno=<%=rs.getString(1)%>"
>Reject</a></td>
</tr>
<%
}

con.close();
} catch (Exception e) {
e.printStackTrace();
```

```
}
%>
</table>
</center>
</body>
</html>
```

Viewusers.jsp

```
<%@page import="java.sql.*"%>
<%@page import="databaseconnection.databaseconn"%>
<%@ page session="true" %>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Vehicle Parking System</title>
<style>
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
  overflow: hidden;
  background-color: #333;
}
li {
  float: left;
}
li a {
  display: block;
  color: white;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
}
```

```
li a:hover:not(.active) {
  background-color: #111;
}

.active {
  background-color:
    #4CAF50;
```

```
}
  .b1{
     background-image: url("img/s3.jpg");
     background-repeat: no-repeat;
     background-size:  cover;
  }
  .s1{
    font-size: 150%;
  }
  </style>
     </head>


    <body class="b1" >
        <header class="main-header">
           <h1 style="text-align:center;"><i style="color:blueviolet">Vehicle Parking
  System</i></h1>
     </header>

    <ul>
    <li><a class="active" href="adminhome.jsp">Home</a></li>
    <li><a class="" href="viewusers.jsp">View_Users</a></li>
    <li><a class="" href="viewrequest.jsp">View_Booking_Request</a></li>
    <li><a class="" href="adminlogin.jsp">Logout</a></li>
  </ul>
     <center class="s1">
      <div class="b2">
        <h1 style="text-align: center"><i>User Information</i></h1>
  <table border="1" width="80%">
  <tr>
  <th>Name</th>
  <th>AadharId</th>
  <th>Phone</th>
  <th>Email</th>
  <th>Address</th>
</tr>
```

```
<%
try{
Connection con= databaseconn.getconnection();
Statement st=con.createStatement();
String sql ="select * from user";
ResultSet rs= st.executeQuery(sql);
while(rs.next()){
%>
 <tr>
<td><%=rs.getString("name") %></td>
<td><%=rs.getString("adharid") %></td>
<td><%=rs.getString("phoneno") %></td>
<td><%=rs.getString("email") %></td>
<td><%=rs.getString("address") %></td>
</tr>

<%
}

con.close();
} catch (Exception e) {
e.printStackTrace();
}
%>
</table>
</center>

</body>
</html>
```

Mail.java

```
package Mail;

import java.util.Properties;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.PasswordAuthentication;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;

/**
 *
 * @author java2
 */
public class Mail {

    public static boolean secretMail(String msg, String userid, String to) {
        Properties props = new Properties();
        props.put("mail.smtp.host", "smtp.gmail.com");
        props.put("mail.smtp.socketFactory.port", "465");
        props.put("mail.smtp.socketFactory.class",
```

```
        "javax.net.ssl.SSLSocketFactory");
    props.put("mail.smtp.auth", "true");
    props.put("mail.smtp.port", "465");
    // Assuming you are sending email from localhost
    Session session = Session.getInstance(props,
          new javax.mail.Authenticator() {
             protected PasswordAuthentication getPasswordAuthentication() {
```



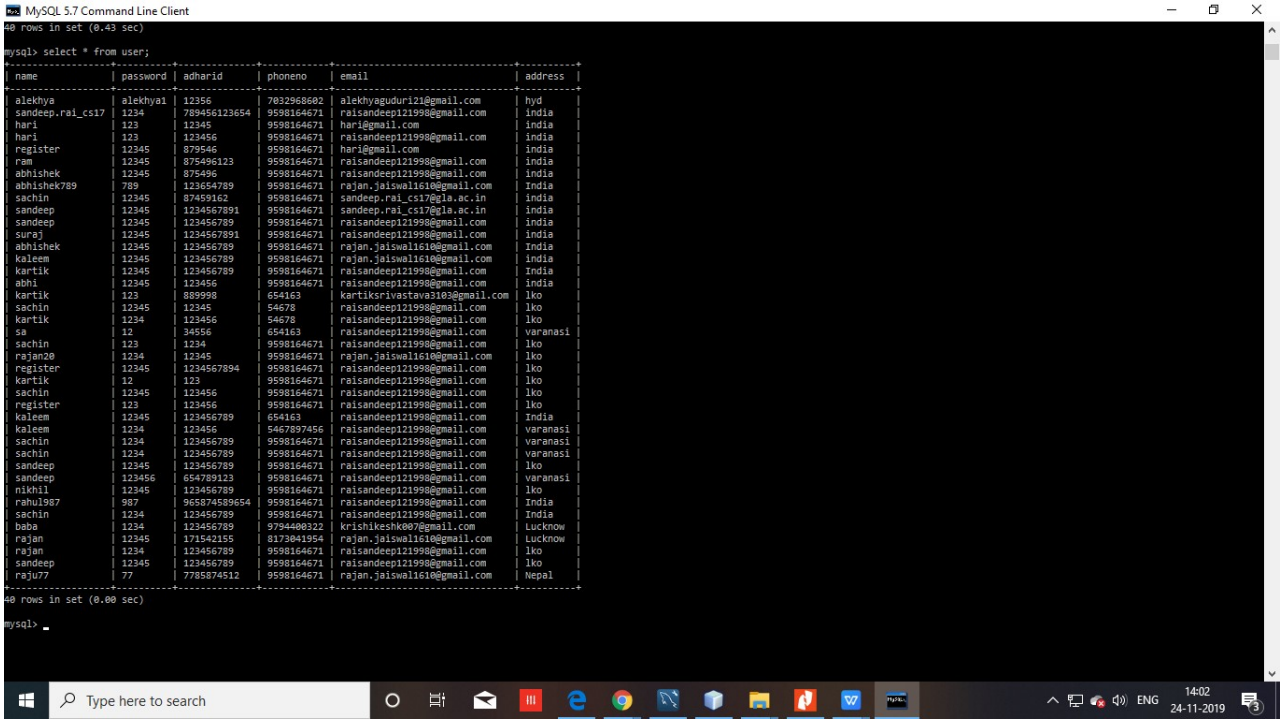
```
                return new
PasswordAuthentication("projects9876@gmail.com","projects123");
             }
          });

    System.out.println("Message   " + msg);
    try {
       Message message = new MimeMessage(session);
       message.setFrom(new InternetAddress(userid));
       message.setRecipients(Message.RecipientType.TO,
       InternetAddress.parse(to));
       message.setSubject("Message ");
       message.setText(msg)
    Transport.send(message);
    System.out.println("Done");
       System.out.println("Done");
       return true;

    } catch (MessagingException e) {
       System.out.println(e);
       e.printStackTrace();
       return false;
       // throw new RuntimeException(e);
    }
 }}
```

Databaseconn.java

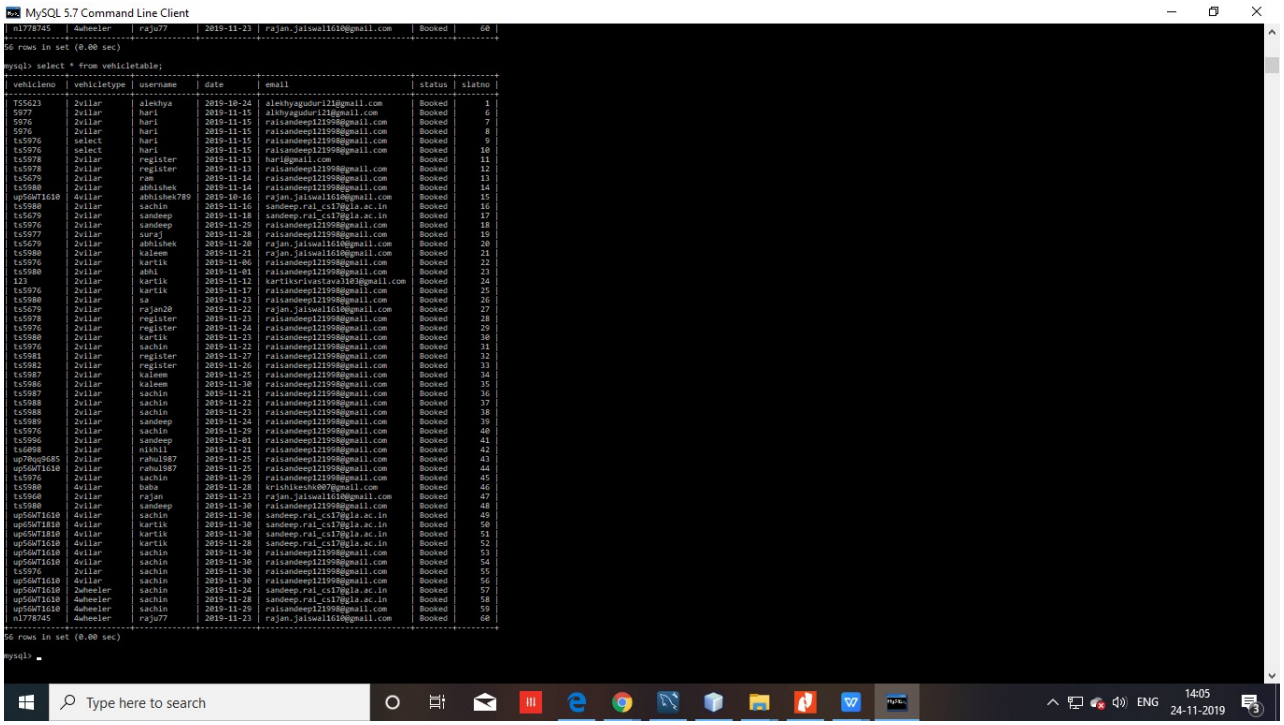
```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package databaseconnection;
import java.sql.Connection;
import java.sql.*;



public class databaseconn
{

    static Connection con;
     public static Connection getconnection()
     {
      try
     {
      Class.forName("com.mysql.jdbc.Driver");


con=DriverManager.getConnection("jdbc:mysql://localhost:33061/vihicleparking",
"root","1234");
     }
      catch(Exception e)
       {
            System.out.println(e);

       }
    return con;
     }
}
```

Screen Shorts:

DATABASE:

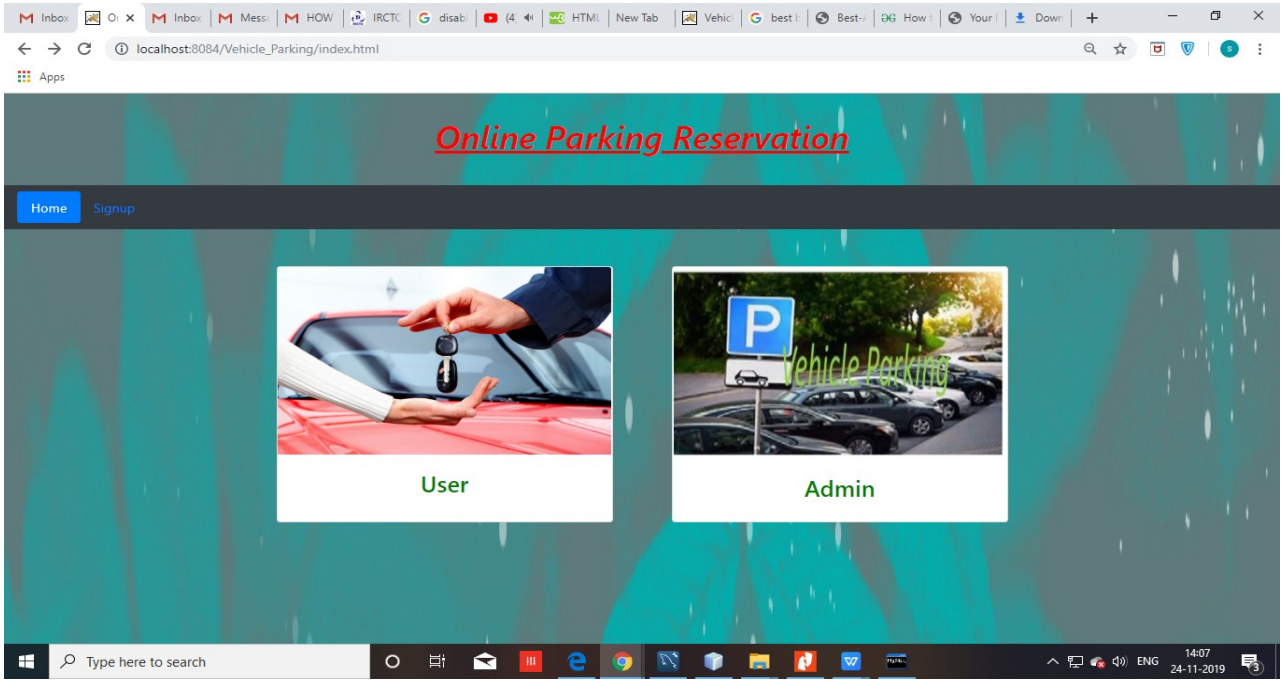


```
MySQL 5.7 Command Line Client
40 rows in set (0.43 sec)

mysql> select * from user;
+------------------+----------+---------------+------------+-----------------------------+---------+
| name             | password | adharid       | phoneno    | email                       | address |
+------------------+----------+---------------+------------+-----------------------------+---------+
| alekhya          | alekhya1 | 12356         | 7032968602 | alekhyagudur121@gmail.com   | hyd     |
| sandeep.rai_cs17 | 1234     | 789456123654  | 9598164671 | raisandeep121998@gmail.com  | india   |
| hari             | 123      | 12345         | 9598164671 | hari@gmail.com              | india   |
| hari             | 123      | 123456        | 9598164671 | raisandeep121998@gmail.com  | india   |
| register         | 12345    | 879546        | 9598164671 | hari@gmail.com              | india   |
| ram              | 12345    | 875496123     | 9598164671 | raisandeep121998@gmail.com  | india   |
| abhishek         | 12345    | 875496        | 9598164671 | raisandeep121998@gmail.com  | india   |
| abhishek789      | 789      | 123654789     | 9598164671 | rajan.jaiswal1610@gmail.com | India   |
| sachin           | 12345    | 87459162      | 9598164671 | sandeep.rai_cs17@gla.ac.in  | india   |
| sandeep          | 12345    | 1234567891    | 9598164671 | sandeep.rai_cs17@gla.ac.in  | india   |
| sandeep          | 12345    | 123456789     | 9598164671 | raisandeep121998@gmail.com  | india   |
| suraj            | 12345    | 1234567891    | 9598164671 | raisandeep121998@gmail.com  | india   |
| abhishek         | 12345    | 123456789     | 9598164671 | rajan.jaiswal1610@gmail.com | India   |
| kaleem           | 12345    | 123456789     | 9598164671 | rajan.jaiswal1610@gmail.com | india   |
| kartik           | 12345    | 123456789     | 9598164671 | raisandeep121998@gmail.com  | India   |
| abhi             | 12345    | 123456        | 9598164671 | raisandeep121998@gmail.com  | india   |
| kartik           | 123      | 889998        | 654163     | kartiksrivastava3103@gmail.com | lko  |
| sachin           | 12345    | 12345         | 54678      | raisandeep121998@gmail.com  | lko     |
| kartik           | 1234     | 123456        | 54678      | raisandeep121998@gmail.com  | lko     |
| sa               | 12       | 34556         | 654163     | raisandeep121998@gmail.com  | varanasi |
| sachin           | 123      | 1234          | 9598164671 | raisandeep121998@gmail.com  | lko     |
| rajan20          | 1234     | 12345         | 9598164671 | rajan.jaiswal1610@gmail.com | lko     |
| register         | 12345    | 1234567894    | 9598164671 | raisandeep121998@gmail.com  | lko     |
| kartik           | 12       | 123           | 9598164671 | raisandeep121998@gmail.com  | lko     |
| sachin           | 12345    | 123456        | 9598164671 | raisandeep121998@gmail.com  | lko     |
| register         | 123      | 123456        | 9598164671 | raisandeep121998@gmail.com  | lko     |
| kaleem           | 12345    | 123456789     | 654163     | raisandeep121998@gmail.com  | India   |
| kaleem           | 1234     | 123456        | 5467897456 | raisandeep121998@gmail.com  | varanasi |
| sachin           | 1234     | 123456789     | 9598164671 | raisandeep121998@gmail.com  | varanasi |
| sachin           | 1234     | 123456789     | 9598164671 | raisandeep121998@gmail.com  | varanasi |
| sandeep          | 12345    | 123456789     | 9598164671 | raisandeep121998@gmail.com  | lko     |
| sandeep          | 123456   | 654789123     | 9598164671 | raisandeep121998@gmail.com  | varanasi |
| nikhil           | 12345    | 123456789     | 9598164671 | raisandeep121998@gmail.com  | lko     |
| rahul987         | 987      | 9658745896654 | 9598164671 | raisandeep121998@gmail.com  | India   |
| sachin           | 1234     | 123456789     | 9598164671 | raisandeep121998@gmail.com  | India   |
| baba             | 1234     | 123456789     | 9794400322 | krishikeshk007@gmail.com    | Lucknow |
| rajan            | 12345    | 171542155     | 8173041954 | rajan.jaiswal1610@gmail.com | Lucknow |
| rajan            | 1234     | 123456789     | 9598164671 | raisandeep121998@gmail.com  | lko     |
| sandeep          | 12345    | 123456789     | 9598164671 | raisandeep121998@gmail.com  | lko     |
| raju77           | 77       | 7785874512    | 9598164671 | rajan.jaiswal1610@gmail.com | Nepal   |
+------------------+----------+---------------+------------+-----------------------------+---------+
40 rows in set (0.00 sec)

mysql>
```

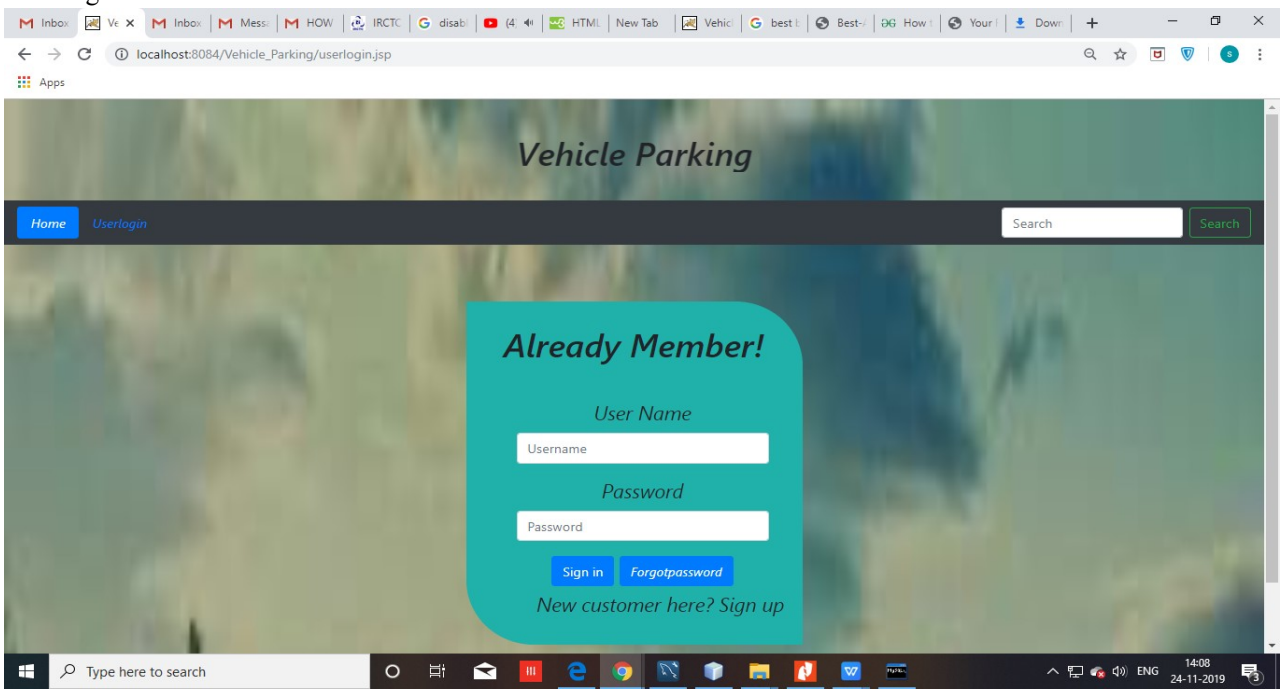


```
MySQL 5.7 Command Line Client
| n1778745 | 4wheeler | raju77 | 2019-11-23 | rajan.jaiswal1610@gmail.com | Booked | 60 |
+----------+----------+--------+------------+-----------------------------+--------+----+
56 rows in set (0.00 sec)

mysql> select * from vehicletable;
+----------+-------------+----------+------------+-----------------------------+--------+--------+
| vehicleno | vehicletype | username | date       | email                       | status | slatno |
+----------+-------------+----------+------------+-----------------------------+--------+--------+
| TS5623   | 2vilar      | alekhya  | 2019-10-24 | alekhyagudur121@gmail.com   | Booked |      1 |
| 5977     | 2vilar      | hari     | 2019-11-15 | alkhyagudur121@gmail.com    | Booked |      6 |
| 5976     | 2vilar      | hari     | 2019-11-15 | raisandeep121998@gmail.com  | Booked |      7 |
| 5976     | 2vilar      | hari     | 2019-11-15 | raisandeep121998@gmail.com  | Booked |      8 |
| ts5976   | select      | hari     | 2019-11-15 | raisandeep121998@gmail.com  | Booked |      9 |
| ts5976   | select      | hari     | 2019-11-15 | raisandeep121998@gmail.com  | Booked |     10 |
| ts5978   | 2vilar      | register | 2019-11-13 | hari@gmail.com              | Booked |     11 |
| ts5978   | 2vilar      | register | 2019-11-13 | raisandeep121998@gmail.com  | Booked |     12 |
| ts5679   | 2vilar      | ram      | 2019-11-14 | raisandeep121998@gmail.com  | Booked |     13 |
| ts5980   | 2vilar      | abhishek | 2019-11-14 | raisandeep121998@gmail.com  | Booked |     14 |
| up56WT1610 | 4vilar    | abhishek789 | 2019-10-16 | rajan.jaiswal1610@gmail.com | Booked |  15 |
| ts5980   | 2vilar      | sachin   | 2019-11-16 | sandeep.rai_cs17@gla.ac.in  | Booked |     16 |
| ts5679   | 2vilar      | sandeep  | 2019-11-18 | sandeep.rai_cs17@gla.ac.in  | Booked |     17 |
| ts5976   | 2vilar      | sandeep  | 2019-11-29 | raisandeep121998@gmail.com  | Booked |     18 |
| ts5977   | 2vilar      | suraj    | 2019-11-28 | raisandeep121998@gmail.com  | Booked |     19 |
| ts5679   | 2vilar      | abhishek | 2019-11-20 | rajan.jaiswal1610@gmail.com | Booked |     20 |
| ts5980   | 2vilar      | kaleem   | 2019-11-21 | rajan.jaiswal1610@gmail.com | Booked |     21 |
| ts5976   | 2vilar      | kartik   | 2019-11-06 | raisandeep121998@gmail.com  | Booked |     22 |
| ts5980   | 2vilar      | abhi     | 2019-11-01 | raisandeep121998@gmail.com  | Booked |     23 |
| 123      | 2vilar      | kartik   | 2019-11-12 | kartiksrivastava3103@gmail.com | Booked |  24 |
| ts5976   | 2vilar      | kartik   | 2019-11-17 | raisandeep121998@gmail.com  | Booked |     25 |
| ts5980   | 2vilar      | sa       | 2019-11-23 | raisandeep121998@gmail.com  | Booked |     26 |
| ts5679   | 2vilar      | rajan20  | 2019-11-22 | rajan.jaiswal1610@gmail.com | Booked |     27 |
| ts5978   | 2vilar      | register | 2019-11-23 | raisandeep121998@gmail.com  | Booked |     28 |
| ts5976   | 2vilar      | register | 2019-11-24 | raisandeep121998@gmail.com  | Booked |     29 |
| ts5980   | 2vilar      | kartik   | 2019-11-23 | raisandeep121998@gmail.com  | Booked |     30 |
| ts5976   | 2vilar      | sachin   | 2019-11-22 | raisandeep121998@gmail.com  | Booked |     31 |
| ts5981   | 2vilar      | register | 2019-11-27 | raisandeep121998@gmail.com  | Booked |     32 |
| ts5982   | 2vilar      | register | 2019-11-26 | raisandeep121998@gmail.com  | Booked |     33 |
| ts5987   | 2vilar      | kaleem   | 2019-11-25 | raisandeep121998@gmail.com  | Booked |     34 |
| ts5986   | 2vilar      | kaleem   | 2019-11-30 | raisandeep121998@gmail.com  | Booked |     35 |
| ts5987   | 2vilar      | sachin   | 2019-11-21 | raisandeep121998@gmail.com  | Booked |     36 |
| ts5988   | 2vilar      | sachin   | 2019-11-22 | raisandeep121998@gmail.com  | Booked |     37 |
| ts5988   | 2vilar      | sachin   | 2019-11-23 | raisandeep121998@gmail.com  | Booked |     38 |
| ts5989   | 2vilar      | sandeep  | 2019-11-24 | raisandeep121998@gmail.com  | Booked |     39 |
| ts5976   | 2vilar      | sachin   | 2019-11-29 | raisandeep121998@gmail.com  | Booked |     40 |
| ts5996   | 2vilar      | sandeep  | 2019-12-01 | raisandeep121998@gmail.com  | Booked |     41 |
| ts6098   | 2vilar      | nikhil   | 2019-11-21 | raisandeep121998@gmail.com  | Booked |     42 |
| up70qq9685 | 2vilar    | rahul987 | 2019-11-25 | raisandeep121998@gmail.com  | Booked |     43 |
| up56WT1610 | 2vilar    | rahul987 | 2019-11-25 | raisandeep121998@gmail.com  | Booked |     44 |
| ts5976   | 2vilar      | sachin   | 2019-11-29 | raisandeep121998@gmail.com  | Booked |     45 |
| ts5980   | 4vilar      | baba     | 2019-11-28 | krishikeshk007@gmail.com    | Booked |     46 |
| ts5960   | 2vilar      | rajan    | 2019-11-23 | rajan.jaiswal1610@gmail.com | Booked |     47 |
| ts5980   | 2vilar      | sandeep  | 2019-11-30 | raisandeep121998@gmail.com  | Booked |     48 |
| up56WT1610 | 4vilar    | sachin   | 2019-11-30 | sandeep.rai_cs17@gla.ac.in  | Booked |     49 |
| up65WT1810 | 4vilar    | kartik   | 2019-11-30 | sandeep.rai_cs17@gla.ac.in  | Booked |     50 |
| up65WT1810 | 4vilar    | kartik   | 2019-11-30 | sandeep.rai_cs17@gla.ac.in  | Booked |     51 |
| up56WT1610 | 4vilar    | kartik   | 2019-11-28 | sandeep.rai_cs17@gla.ac.in  | Booked |     52 |
| up56WT1610 | 4vilar    | sachin   | 2019-11-30 | raisandeep121998@gmail.com  | Booked |     53 |
| up56WT1610 | 4vilar    | sachin   | 2019-11-30 | raisandeep121998@gmail.com  | Booked |     54 |
| ts5976   | 2vilar      | sachin   | 2019-11-30 | raisandeep121998@gmail.com  | Booked |     55 |
| up56WT1610 | 4vilar    | sachin   | 2019-11-30 | raisandeep121998@gmail.com  | Booked |     56 |
| up56WT1610 | 2wheeler  | sachin   | 2019-11-24 | sandeep.rai_cs17@gla.ac.in  | Booked |     57 |
| up56WT1610 | 4wheeler  | sachin   | 2019-11-28 | sandeep.rai_cs17@gla.ac.in  | Booked |     58 |
| up56WT1610 | 4wheeler  | sachin   | 2019-11-29 | raisandeep121998@gmail.com  | Booked |     59 |
| n1778745 | 4wheeler | raju77 | 2019-11-23 | rajan.jaiswal1610@gmail.com | Booked | 60 |
+----------+----------+--------+------------+-----------------------------+--------+----+
56 rows in set (0.00 sec)

mysql>
```

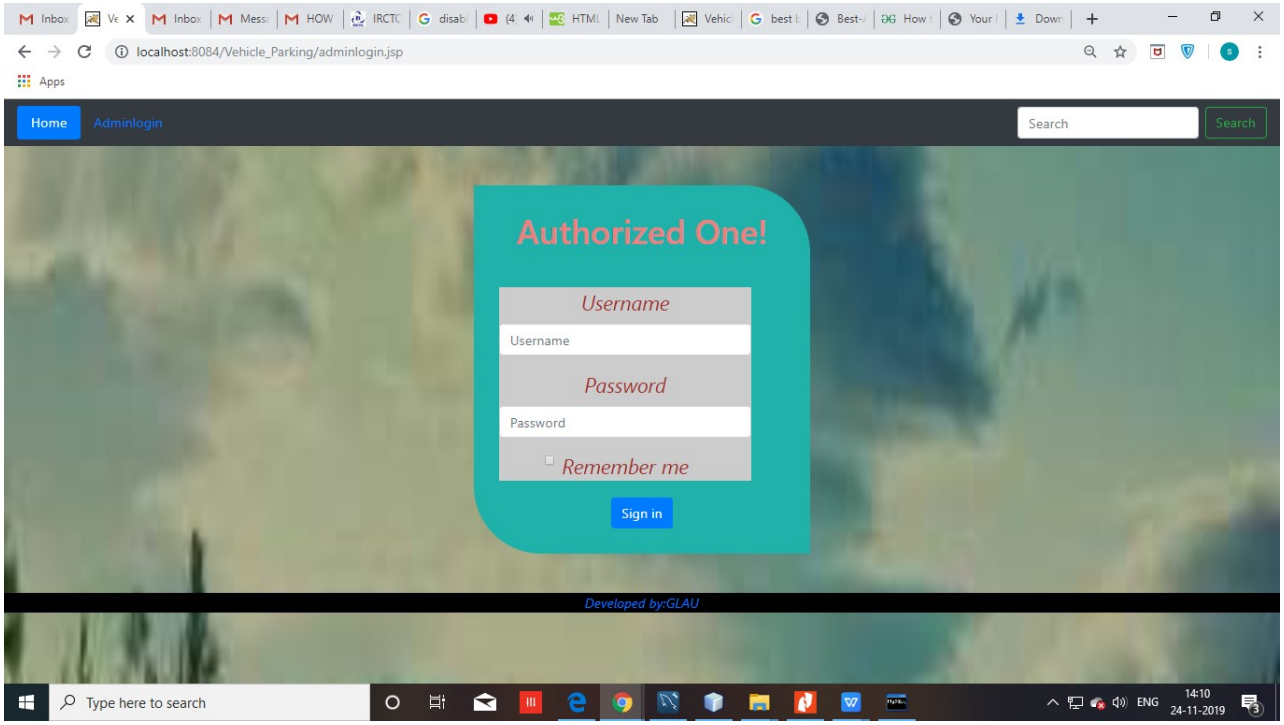
Home:



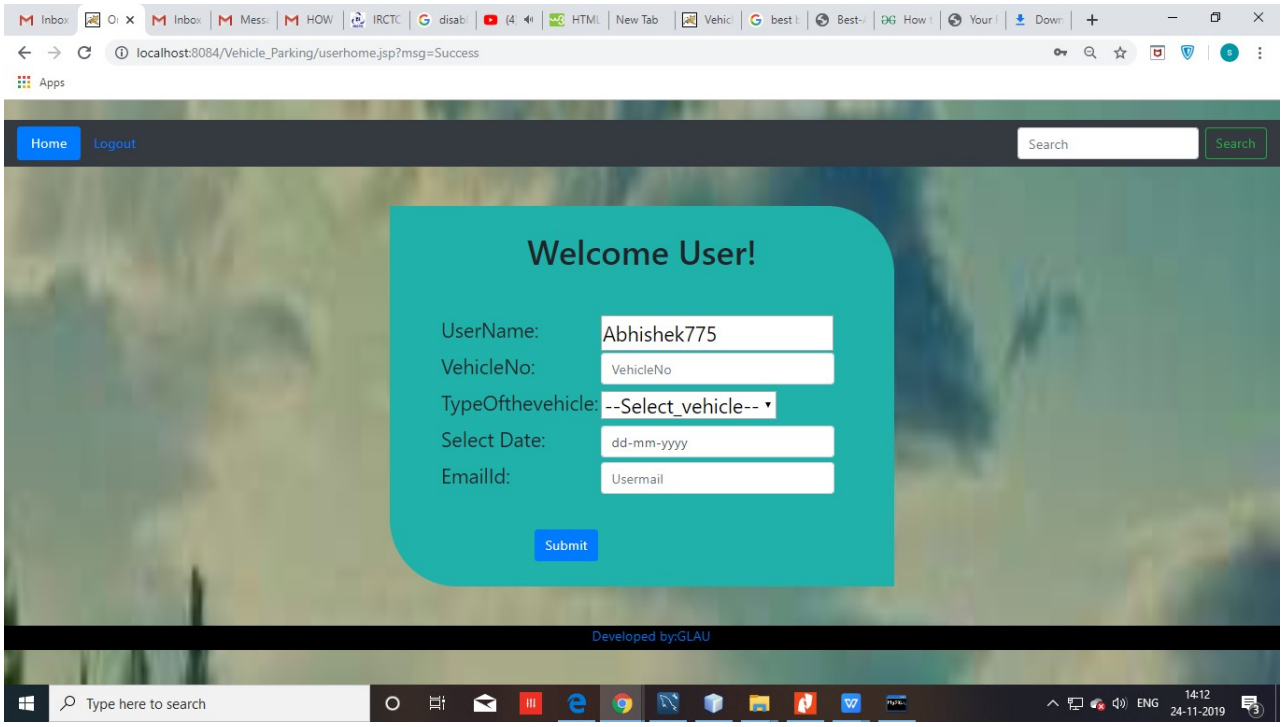
User login:



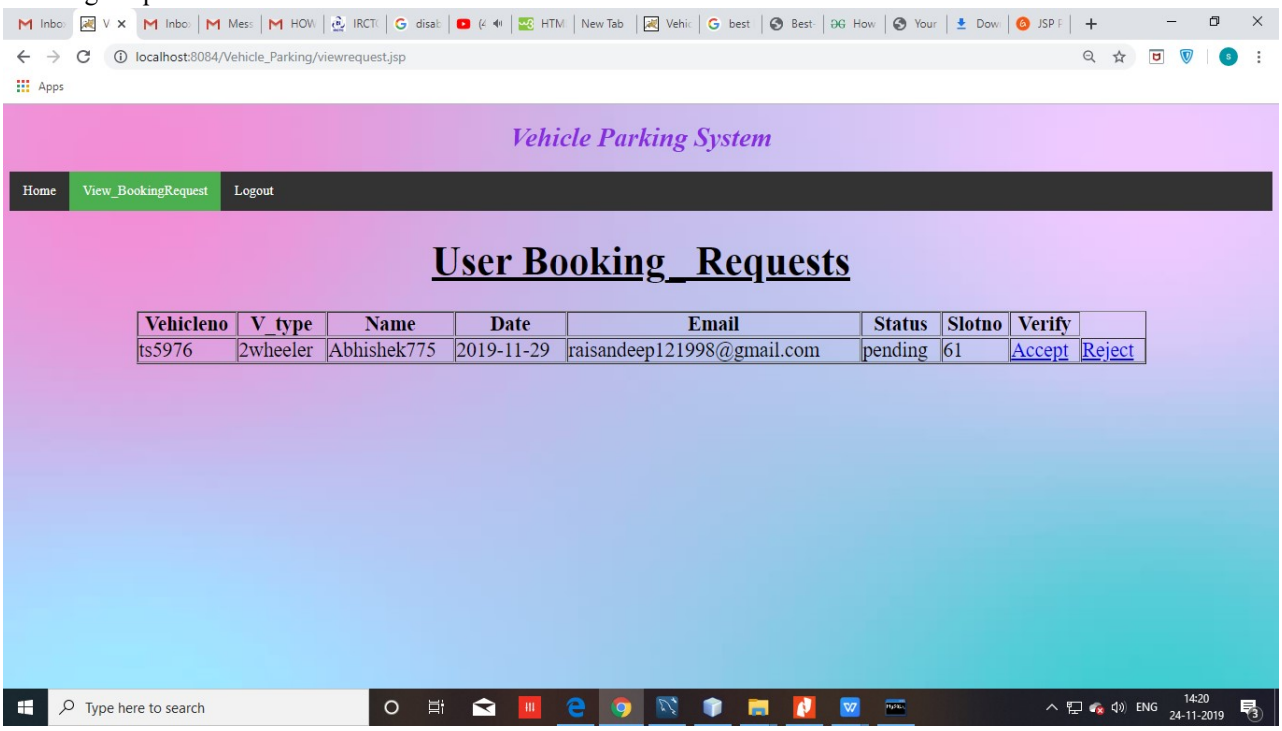
Admin Login



User Home:

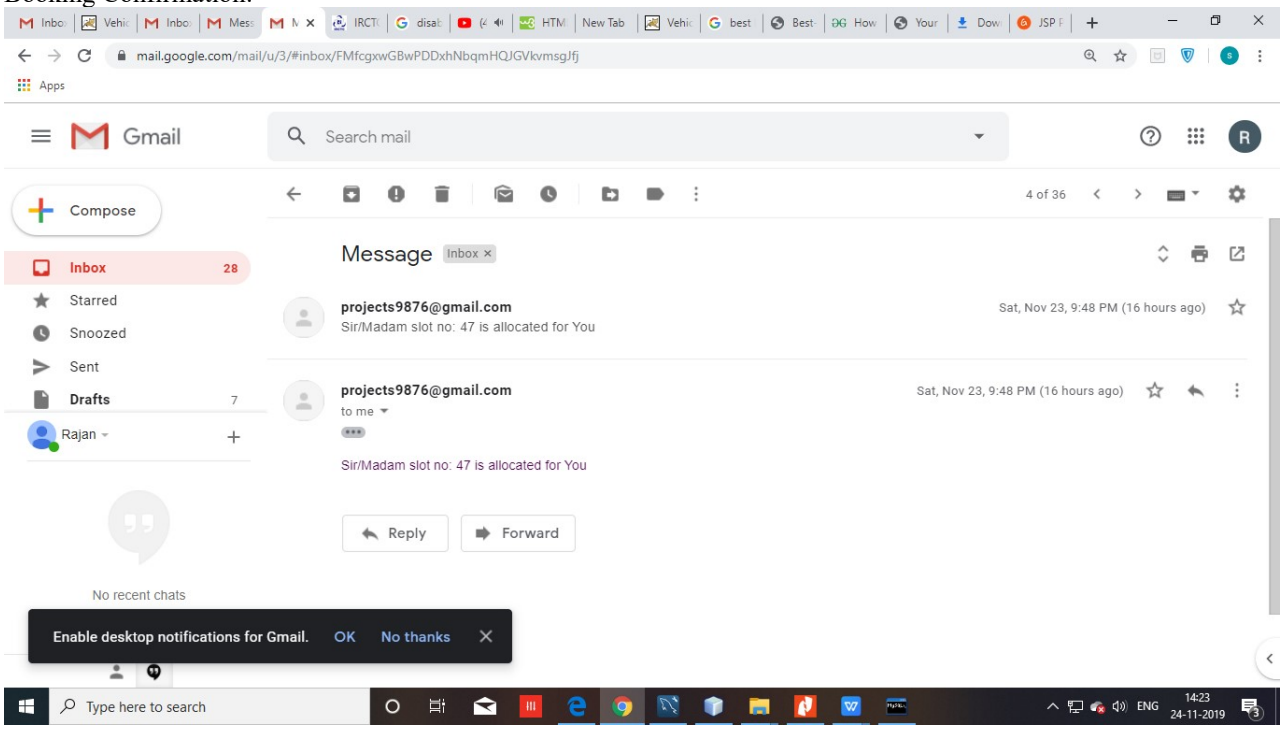


Booking Request



:

Booking Confirmation:



BIBLOGRAPHY

1. www.geeksforgeeks.com
2. www.javatpoint.com