**ONLINE PARKING SYSTEM**

* 1. **Project Description**

The main objective of this project is to design a solution for overcoming the parking issues that exist in public places such as malls, multiplexes etc. especially on weekends. The aim is to achieve this by using the concept of Internet of Things. Where in an Android Application is created for the customer, whose details are constantly updated by the hardware/server at the location. The features include unique identification for each vehicle, display of available parking slots on the mobile application, possibility of making reservations for the same, maintenance of a database (for the management).

* + 1. **Project overview**

Online Parking Manager is a concept where technological solutions are designed to overcome these issues. They can be avoided if a person is enabled to book the parking slots in advance. The cost-effective technology used in this project enables customers to check if parking slots are available at a specific location such as a mall or a movie theatre and also to book an available slot for a specific period for their vehicle using the same. This can be done on a mobile phone using an android application. This application can be managed by an administrator at the malls and multiplexes. The customer information and the booking details are stored one continuously-updating central server. The payment for the booked slot can be done via deduction of the amount from the customer's account which c

an be refilled online itself. The vehicle can be parked at the booked slot after verification, for which hardware is used. This project primarily makes finding parking systems more efficient and less time-consuming, and also eliminates the need of a large staff to be employed for such purposes at these places, thus increasing the profit margins.

* + 1. **Impact of user**

This can be done on a mobile phone using an android application. This application can be managed by an administrator at the malls and multiplexes.

The customer information and the booking details are stored one continuously-updating central server.

# 1.1.3 Software Hardware Requirements:

Front End: XTML

Language: C#

Back end: SQL

Operating system WINDOWS 7

PROCESSOR INTEL COREI3

# Hardware Requirements:

Hard Disk: 80GB

RAM : 4GB

Accessories Mouse and Keyboard

**2.Literature** [survey](http://www.blurtit.com/q876299.html)

Literature [survey](http://www.blurtit.com/q876299.html) is the most important step in software development process. Before developing the tool it is necessary to determine the time factor, economy n company strength. Once these things r satisfied, ten next steps are to determine which operating system and language can be used for developing the tool. Once the [programmers](http://www.blurtit.com/q876299.html) start building the tool the programmers need lot of external support. This support can be obtained from senior programmers, from [book](http://www.blurtit.com/q876299.html) or from websites. Before building the system the above consideration r taken into account for developing the proposed system.

The lack of benchmarks for OSN short text classification makes problematic the development of a reliable comparative analysis. However, an indirect comparison of our method can be done with work that show similarities or complementary aspects with our solution. A study that responds to these characteristics is proposed in , where a classification of incoming tweets into five categories is described. Similarly to our approach, messages are very short and represented in the learning framework with both internal, content-based and contextual properties. In particular, the features considered in are BoW, Author Name, plus eight document properties features. Qualitatively speaking, the results of the analysis conducted in on the representative power of the three type of features tallied in general with our conclusions: contextual features are found to be very discriminative and BoW considered alone does not reach a satisfactory performance. Best numerical results obtained in our work are comparable with those obtained in . Limiting to accuracy index, which is the only metric used in , our results are slightly inferior, but this result must be interpreted considering the following aspects. First of all, we use a much smaller set of preclassified data (1,266 versus 5,407), and this is an advantage over the tweets classification considering the efforts in manually preclassifying messages with an acceptable level of consistency. Second, the classes we considered have a higher degree of vagueness, since their semantics is closely linked to subjective interpretation. A second work provides weak conditions for a comparative evaluation. The authors deal with short text classification using a statistical model, named Prediction by Partial Matching (PPM), without feature engineering. However, their study is oriented to text containing complex terminology and prove the classifier on medical texts from Newsgroups, clinical texts, and Reuters- These differences may lower the level of reliability in comparison. In addition, we observe that the performance reported in is strongly affected by the data set used in the evaluation. If we consider results in obtained on clinical texts our classifier with the best results of Prec. 0.76, Recall 0.59, is considerably higher than PPM classifier (Prec. 0.36, Recall 0.42). It has a comparable behavior, if we consider the averaged performance on three Reuters subsets (Prec. 0.74, Recall 0.63) and slightly inferior when considering the newsgroups data set (Prec. 0.96, Recall 0.84). 6.3 Overall Performance and Discussion In order to provide an overall assessment of how effectively the system applies a FR, we look again at Table 2. This table allows us to estimate the Precision and Recall of our FRs, since values reported in Table 2 have been computed for FRs with content specification component set to ðC; 0:5Þ, where C 2 \_. Let us suppose that the system applies a given rule on a certain message. As such, Precision reported in Table 2 is the probability that the decision taken on the considered message (that is, blocking it or not) is actually the correct one. In contrast, Recall has to be interpreted as the probability that, given a rule that must be applied over a certain message, the rule is really enforced. Let us now discuss, with some examples, the results presented in Table 2, which reports Precision and Recall values. The second column of Table 2 represents the Precision and the Recall value computed for FRs with ðNeutral; 0:5Þ content constraint. In contrast, the fifth column stores the Precision and the Recall value computed for FRs with ðV ulgar; 0:5Þ constraint. Results achieved by the content-based specification component, on the first-level classification, can be considered good enough and reasonably aligned with those obtained by well-known information filtering techniques .

**2.1Existing System**

Nowadays, a problem of finding parking slots for vehicles is observed in malls mostly on public holidays and weekends. It has been observed that a considerable amount of time gets wasted while trying to find a vacant spot at large public parking spaces, and if no parking slots are available, then the parking has to be done elsewhere, thus resulting in more time being wasted.

**2.2Proposed System:**

The customer information and the booking details are stored one continuously-updating central server. The payment for the booked slot can be done via deduction of the amount from the customer's account which can be refilled online itself. The vehicle can be parked at the booked slot after verification, for which hardware is used. This project primarily makes finding parking systems more efficient and less time-consuming, and also eliminates the need of a large staff to be employed for such purposes at these places, thus increasing the profit margins.

**2.3FEASIBILITY STUDY**

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

* ECONOMICAL FEASIBILITY
* TECHNICAL FEASIBILITY
* SOCIAL FEASIBILITY

**ECONOMICAL FEASIBILITY**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

### TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

**SOCIAL FEASIBILITY**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

**3.Developing Tools**

## 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 (Java VM). Every Java interpreter, whether it’s a development tool or a Web browser that can run applets, is an implementation of the Java VM. Java byte codes help make “write once, run anywhere” possible. You can compile your program into byte codes on any platform that has a Java compiler. The byte codes can then be run on any implementation of the Java VM. That means that as long as a computer has a Java VM, the same program written in the Java programming language can run on Windows 2000, a Solaris workstation, or on an iMac.



### 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 JavaBeansTM, can plug into existing component architectures.
* **Object serialization**: Allows lightweight persistence and communication via Remote Method Invocation (RMI).
* **Java Database Connectivity (JDBCTM)**: Provides uniform access to a wide range of relational databases.

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



## How Will Java Technology Change My Life?

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

* **Get started quickly**: Although the Java programming language is a powerful object-oriented language, it’s easy to learn, especially for programmers already familiar with C or C++.
* **Write less code**: Comparisons of program metrics (class counts, method counts, and so on) suggest that a program written in the Java programming language can be four times smaller than the same program in C++.
* **Write better code**: The Java programming language encourages good coding practices, and its garbage collection helps you avoid memory leaks. Its object orientation, its JavaBeans component architecture, and its wide-ranging, easily extendible API let you reuse other people’s tested code and introduce fewer bugs.
* **Develop programs more quickly**: Your development time may be as much as twice as fast versus writing the same program in C++. Why? You write fewer lines of code and it is a simpler programming language than C++.
* **Avoid platform dependencies with 100% Pure Java**: You can keep your program portable by avoiding the use of libraries written in other languages. The 100% Pure JavaTM Product Certification Program has a repository of historical process manuals, white papers, brochures, and similar materials online.
* **Write once, run anywhere**: Because 100% Pure Java programs are compiled into machine-independent byte codes, they run consistently on any Java platform.
* **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.

1. ***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.

1. ***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.
2. ***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.

1. ***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.

1. ***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.

1. ***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.

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.

**Java Program**

**Compilers**

**Interpreter**

**My Program**

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

**What is a Java Web Application?**

A Java web application generates interactive web pages containing various types of markup language (HTML, XML, and so on) and dynamic content. It is typically comprised of web components such as JavaServer Pages (JSP), servlets and JavaBeans to modify and temporarily store data, interact with databases and web services, and render content in response to client requests.

Because many of the tasks involved in web application development can be repetitive or require a surplus of boilerplate code, web frameworks can be applied to alleviate the overhead associated with common activities. For example, many frameworks, such as JavaServer Faces, provide libraries for templating pages and session management, and often promote code reuse.

**What is Java EE?**

Java EE (Enterprise Edition) is a widely used platform containing a set of coordinated technologies that significantly reduce the cost and complexity of developing, deploying, and managing multi-tier, server-centric applications. Java EE builds upon the Java SE platform and provides a set of APIs (application programming interfaces) for developing and running portable, robust, scalable, reliable and secure server-side applications.

Some of the fundamental components of Java EE include:

* Enterprise JavaBeans (EJB): a managed, server-side component architecture used to encapsulate the business logic of an application. EJB technology enables rapid and simplified development of distributed, transactional, secure and portable applications based on Java technology.
* Java Persistence API (JPA): a framework that allows developers to manage data using object-relational mapping (ORM) in applications built on the Java Platform.

**JavaScript and Ajax Development**

JavaScript is an object-oriented scripting language primarily used in client-side interfaces for web applications. Ajax (Asynchronous JavaScript and XML) is a Web 2.0 technique that allows changes to occur in a web page without the need to perform a page refresh. JavaScript toolkits can be leveraged to implement Ajax-enabled components and functionality in web pages.

**Web Server and Client**

Web Server is a software that can process the client request and send the response back to the client. For example, Apache is one of the most widely used web server. Web Server runs on some physical machine and listens to client request on specific port.

A web client is a software that helps in communicating with the server. Some of the most widely used web clients are Firefox, Google Chrome, Safari etc. When we request something from server (through URL), web client takes care of creating a request and sending it to server and then parsing the server response and present it to the user.

**HTML and HTTP**

Web Server and Web Client are two separate softwares, so there should be some common language for communication. HTML is the common language between server and client and stands for **H**yper**T**ext **M**arkup **L**anguage.

Web server and client needs a common communication protocol, HTTP (**H**yper**T**ext **T**ransfer **P**rotocol) is the communication protocol between server and client. HTTP runs on top of TCP/IP communication protocol.

Some of the important parts of HTTP Request are:

* **HTTP Method** – action to be performed, usually GET, POST, PUT etc.
* **URL** – Page to access
* **Form Parameters** – similar to arguments in a java method, for example user,password details from login page.

Sample HTTP Request:

|  |  |
| --- | --- |
| 1  2  3 | GET /FirstServletProject/jsps/hello.jsp HTTP/1.1  Host: localhost:8080  Cache-Control: no-cache |

Some of the important parts of HTTP Response are:

* **Status Code** – an integer to indicate whether the request was success or not. Some of the well known status codes are 200 for success, 404 for Not Found and 403 for Access Forbidden.
* **Content Type** – text, html, image, pdf etc. Also known as MIME type
* **Content** – actual data that is rendered by client and shown to user.

**MIME Type or Content Type**: If you see above sample HTTP response header, it contains tag “Content-Type”. It’s also called MIME type and server sends it to client to let them know the kind of data it’s sending. It helps client in rendering the data for user. Some of the mostly used mime types are text/html, text/xml, application/xml etc.

### 

### Understanding URL

URL is acronym of Universal Resource Locator and it’s used to locate the server and resource. Every resource on the web has it’s own unique address. Let’s see parts of URL with an example.

**http://localhost:8080/FirstServletProject/jsps/hello.jsp**

**http://** – This is the first part of URL and provides the communication protocol to be used in server-client communication.

**localhost** – The unique address of the server, most of the times it’s the hostname of the server that maps to unique IP address. Sometimes multiple hostnames point to same IP addresses and web server virtual host takes care of sending request to the particular server instance.

**8080** – This is the port on which server is listening, it’s optional and if we don’t provide it in URL then request goes to the default port of the protocol. Port numbers 0 to 1023 are reserved ports for well known services, for example 80 for HTTP, 443 for HTTPS, 21 for FTP etc.

**FirstServletProject/jsps/hello.jsp** – Resource requested from server. It can be static html, pdf, JSP, servlets, PHP etc.

### 

### Why we need Servlet and JSPs?

Web servers are good for static contents HTML pages but they don’t know how to generate dynamic content or how to save data into databases, so we need another tool that we can use to generate dynamic content. There are several programming languages for dynamic content like PHP, Python, Ruby on Rails, Java Servlets and JSPs.

Java Servlet and JSPs are server side technologies to extend the capability of web servers by providing support for dynamic response and data persistence.

### 

### Web Container

Tomcat is a web container, when a request is made from Client to web server, it passes the request to web container and it’s web container job to find the correct resource to handle the request (servlet or JSP) and then use the response from the resource to generate the response and provide it to web server. Then web server sends the response back to the client.

When web container gets the request and if it’s for servlet then container creates two Objects HTTPServletRequest and HTTPServletResponse. Then it finds the correct servlet based on the URL and creates a thread for the request. Then it invokes the servlet service() method and based on the HTTP method service() method invokes doGet() or doPost() methods. Servlet methods generate the dynamic page and write it to response. Once servlet thread is complete, container converts the response to HTTP response and send it back to client.

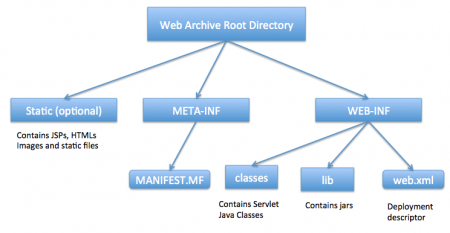
Some of the important work done by web container are:

* **Communication Support** – Container provides easy way of communication between web server and the servlets and JSPs. Because of container, we don’t need to build a server socket to listen for any request from web server, parse the request and generate response. All these important and complex tasks are done by container and all we need to focus is on our business logic for our applications.
* **Lifecycle and Resource Management** – Container takes care of managing the life cycle of servlet. Container takes care of loading the servlets into memory, initializing servlets, invoking servlet methods and destroying them. Container also provides utility like JNDI for resource pooling and management.
* **Multithreading Support** – Container creates new thread for every request to the servlet and when it’s processed the thread dies. So servlets are not initialized for each request and saves time and memory.
* **JSP Support** – JSPs doesn’t look like normal java classes and web container provides support for JSP. Every JSP in the application is compiled by container and converted to Servlet and then container manages them like other servlets.
* **Miscellaneous Task** – Web container manages the resource pool, does memory optimizations, run garbage collector, provides security configurations, support for multiple applications, hot deployment and several other tasks behind the scene that makes our life easier.

### 

### Web Application Directory Structure

Java Web Applications are packaged as Web Archive (WAR) and it has a defined structure. You can export above dynamic web project as WAR file and unzip it to check the hierarchy. It will be something like below image.

[](http://www.journaldev.com/wp-content/uploads/2013/08/WAR-directory-structure.png)

### Deployment Descriptor

**web.xml** file is the deployment descriptor of the web application and contains mapping for servlets (prior to 3.0), welcome pages, security configurations, session timeout settings etc.

Thats all for the java web application startup tutorial, we will explore Servlets and JSPs more in future posts.

INTRODUCTION TO SQL SERVER

### MySQL:

MySQL, the most popular Open Source SQL database management system, is developed, distributed, and supported by Oracle Corporation.

The MySQL Web site (http://www.mysql.com/) provides the latest information about MySQL software.

* **MySQL is a database management system.**

A database is a structured collection of data. It may be anything from a simple shopping list to a picture gallery or the vast amounts of information in a corporate network. To add, access, and process data stored in a computer database, you need a database management system such as MySQL Server. Since computers are very good at handling large amounts of data, database management systems play a central role in computing, as standalone utilities, or as parts of other applications.

* **MySQL databases are relational.**

A relational database stores data in separate tables rather than putting all the data in one big storeroom. The database structures are organized into physical files optimized for speed. The logical model, with objects such as databases, tables, views, rows, and columns, offers a flexible programming environment. You set up rules governing the relationships between different data fields, such as one-to-one, one-to-many, unique, required or optional, and “pointers” between different tables. The database enforces these rules, so that with a well-designed database, your application never sees inconsistent, duplicate, orphan, out-of-date, or missing data.

The SQL part of “MySQL” stands for “Structured Query Language”. SQL is the most common standardized language used to access databases. Depending on your programming environment, you might enter SQL directly (for example, to generate reports), embed SQL statements into code written in another language, or use a language-specific API that hides the SQL syntax.

SQL is defined by the ANSI/ISO SQL Standard. The SQL standard has been evolving since 1986 and several versions exist. In this manual, “SQL-92” refers to the standard released in 1992, “SQL:1999” refers to the standard released in 1999, and “SQL:2003” refers to the current version of the standard. We use the phrase “the SQL standard” to mean the current version of the SQL Standard at any time.

* **MySQL software is Open Source.**

Open Source means that it is possible for anyone to use and modify the software. Anybody can download the MySQL software from the Internet and use it without paying anything. If you wish, you may study the source code and change it to suit your needs. The MySQL software uses the GPL (GNU General Public License), http://www.fsf.org/licenses/, to define what you may and may not do with the software in different situations. If you feel uncomfortable with the GPL or need to embed MySQL code into a commercial application, you can buy a commercially licensed version from us. See the MySQL Licensing Overview for more information (http://www.mysql.com/company/legal/licensing/).

* **The MySQL Database Server is very fast, reliable, scalable, and easy to use.**

If that is what you are looking for, you should give it a try. MySQL Server can run comfortably on a desktop or laptop, alongside your other applications, web servers, and so on, requiring little or no attention. If you dedicate an entire machine to MySQL, you can adjust the settings to take advantage of all the memory, CPU power, and I/O capacity available. MySQL can also scale up to clusters of machines, networked together.

You can find a performance comparison of MySQL Server with other database managers on our benchmark page.

MySQL Server was originally developed to handle large databases much faster than existing solutions and has been successfully used in highly demanding production environments for several years. Although under constant development, MySQL Server today offers a rich and useful set of functions. Its connectivity, speed, and security make MySQL Server highly suited for accessing databases on the Internet.

* **MySQL Server works in client/server or embedded systems.**

The MySQL Database Software is a client/server system that consists of a multi-threaded SQL server that supports different backends, several different client programs and libraries, administrative tools, and a wide range of application programming interfaces (APIs).

We also provide MySQL Server as an embedded multi-threaded library that you can link into your application to get a smaller, faster, easier-to-manage standalone product.

* **A large amount of contributed MySQL software is available.**

MySQL Server has a practical set of features developed in close cooperation with our users. It is very likely that your favorite application or language supports the MySQL Database Server.

The official way to pronounce “MySQL” is “My Ess Que Ell” (not “my sequel”), but we do not mind if you pronounce it as “my sequel” or in some other localized way.

**4.System Design**

Diagrams:

****

****

****

**3 SYSTEM DESIGN**

**3.1Data Flow Diagram**

Register Page

Login

User Home

Park here

view

Add feedback

View parking

Check avaliblity

A Admin

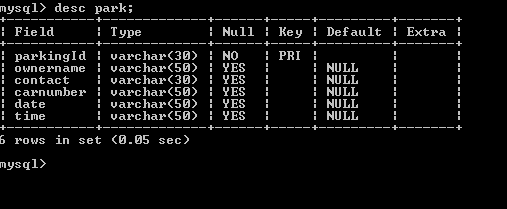
View customers

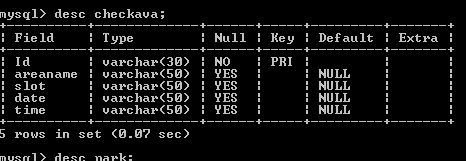
Add status

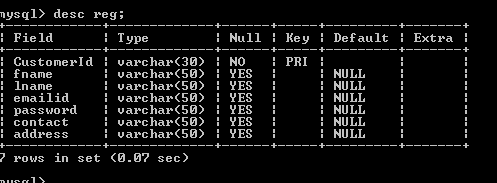
Add parking ststua

****

Database design:







### 5.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.

**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.

Source Code

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">

<title>admin

</title>

<style type="text/css">

.style1

{

width: 100%;

font-weight: bold;

border: 6px solid white;

background-color: yellowgreen;

}

.style2

{

font-size: xx-large;

color: white;

}

.style3

{

width: 60%;

border: 4px solid PINK;

}

</style>

</head>

<body background="images/car (1).gif">

<form NAME="HOME" ACTION="HOMEPAGE" METHOD="GET">

<div>

<table cellpadding="10" cellspacing="0" width="100%" class="style1">

<tr>

<td class="style2">

<center>Admin Contol</center></td>

</tr>

</table>

</div>

<hr color="red" width="100%" size="8px"/>

<table align="left" cellpadding="6" cellspacing="4" width="100%" bgcolor="skyblue" >

<tr>

<td>

<a href="customerdetails.jsp">ViewUsers</a>

</td>

<td>

<a href="ava.jsp">Add Availability</a>

</td>

<td>

<a href="pkde.jsp">Parking Details</a>

</td>

<td>

<a href="feeds.jsp">ViewFeedbacks</a>

</td>

<td>

<a href="d1.jsp">Delete Parking</a>

</td>

<td>

<a href="home.html">Logout</a>

</td>

</table><br>

<img src="">

</table>

</form>

</body>

</html>

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>

<%@ page import="java.sql.\*" %>

<%@ page import="java.io.\*" %>

<!DOCTYPE html>

<html lang="en">

<head>

<title>car</title>

<body background="images/1 (4).jpg">

<table border="2" cellpadding="2" cellspacing="0" bgcolor="skyblue" width="40%">

<tr><td><center><font size="+5" color="GREEN">CustomerInfo</center></td>

</tr>

<a href="adminhome.jsp">BACK</a>

<hr size="10" color="green"></hr>

<%

try {

String connectionURL = "jdbc:mysql://localhost:3306/parking";

Connection connection = null;

Statement statement = null;

ResultSet rs = null;

Class.forName("com.mysql.jdbc.Driver").newInstance();

connection = DriverManager.getConnection(connectionURL, "root", "root");

statement = connection.createStatement();

String QueryString = "SELECT \* from reg";

rs = statement.executeQuery(QueryString);

%>

<%

while (rs.next()) {

%>

<table bordercolor="black" border="5" bgcolor="lightyellow" cellpadding="0"

cellspacing="2" WIDTH="40%">

<TR>

<TD><font color="green"></b><center>CustomerId</center></td>

<td><font color="green"></b><center>Fname</center></td>

<td><font color="green"></b><center>Lname</center></td>

<TD><font color="green"></b><center>emailid</center></td>

<td><font color="green"></b><center>Contact</center></td>

<td><font color="green"></b><center>address</center></td>

</TR>

<tr><td></b><center><%=rs.getString(1)%></center>

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

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

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

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

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

</tr>

<% } %>

<%

// close all the connections.

rs.close();

statement.close();

connection.close();

} catch (Exception ex) {

%>

<font size="+3" color="red"></b>

<%

out.println("Unable to connect to database.");

}

%>

</TABLE><TABLE>

<TR>

<TD><FORM ACTION="welcome\_to\_database\_query.jsp" method="get" >

</TD>

</TR>

</body>

</TABLE>

</font>

</body>

</html>

<!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->

<script src="js/jquery-2.1.1.min.js"></script>

<!-- Include all compiled plugins (below), or include individual files as needed -->

<script src="js/bootstrap.min.js"></script>

<script src="js/wow.min.js"></script>

<script src="js/jquery.easing.1.3.js"></script>

<script src="js/jquery.bxslider.min.js"></script>

<script src="js/jquery.isotope.min.js"></script>

<script src="js/fancybox/jquery.fancybox.pack.js"></script>

<script src="js/functions.js"></script>

<script>

wow = new WOW(

{

} )

.init();

</script>

</html>

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>

<%@ page import="java.sql.\*" %>

<%@ page import="java.io.\*" %>

<!DOCTYPE html>

<html lang="en">

<head>

<title>car</title>

<body background="images/1 (4).jpg">

<table border="2" cellpadding="2" cellspacing="0" bgcolor="skyblue" width="40%">

<tr><td><center><font size="+5" color="GREEN">CustomerInfo</center></td>

</tr>

<a href="adminhome.jsp">BACK</a>

<hr size="10" color="green"></hr>

<%

try {

String connectionURL = "jdbc:mysql://localhost:3306/parking";

Connection connection = null;

Statement statement = null;

ResultSet rs = null;

Class.forName("com.mysql.jdbc.Driver").newInstance();

connection = DriverManager.getConnection(connectionURL, "root", "root");

statement = connection.createStatement();

String QueryString = "SELECT \* from reg";

rs = statement.executeQuery(QueryString);

%>

<%

while (rs.next()) {

%>

<table bordercolor="black" border="5" bgcolor="lightyellow" cellpadding="0"

cellspacing="2" WIDTH="40%">

<TR>

<TD><font color="green"></b><center>CustomerId</center></td>

<td><font color="green"></b><center>Fname</center></td>

<td><font color="green"></b><center>Lname</center></td>

<TD><font color="green"></b><center>emailid</center></td>

<td><font color="green"></b><center>Contact</center></td>

<td><font color="green"></b><center>address</center></td>

</TR>

<tr><td></b><center><%=rs.getString(1)%></center>

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

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

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

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

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

</tr>

<% } %>

<%

// close all the connections.

rs.close();

statement.close();

connection.close();

} catch (Exception ex) {

%>

<font size="+3" color="red"></b>

<%

out.println("Unable to connect to database.");

}

%>

</TABLE><TABLE>

<TR>

<TD><FORM ACTION="welcome\_to\_database\_query.jsp" method="get" >

</TD>

</TR>

</body>

</TABLE>

</font>

</body>

</html>

<!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->

<script src="js/jquery-2.1.1.min.js"></script>

<!-- Include all compiled plugins (below), or include individual files as needed -->

<script src="js/bootstrap.min.js"></script>

<script src="js/wow.min.js"></script>

<script src="js/jquery.easing.1.3.js"></script>

<script src="js/jquery.bxslider.min.js"></script>

<script src="js/jquery.isotope.min.js"></script>

<script src="js/fancybox/jquery.fancybox.pack.js"></script>

<script src="js/functions.js"></script>

<script>

wow = new WOW(

{

} )

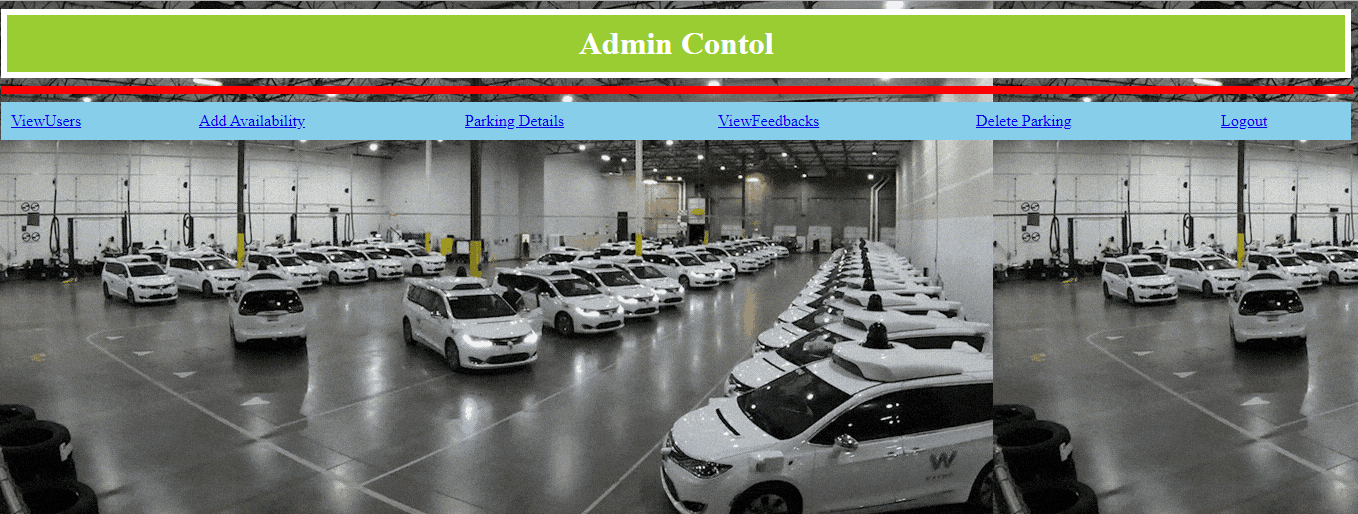
.init();

</script>

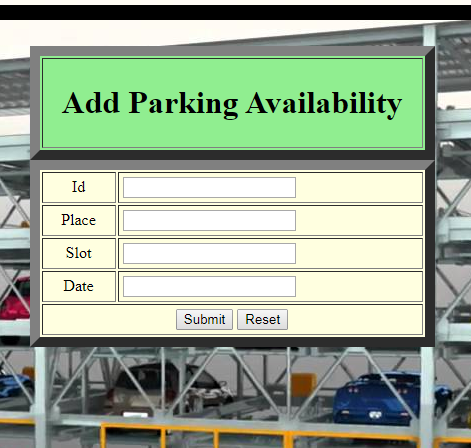
</html>

**6.Screen Shots**

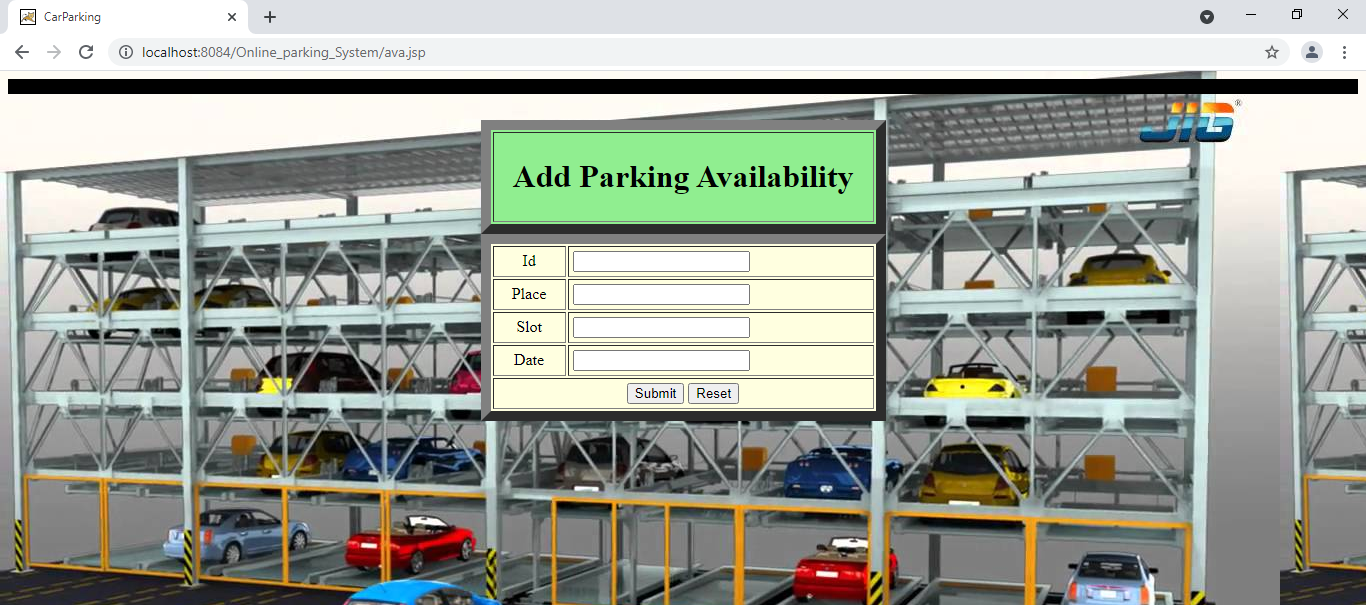
Screen shots











**7.FUTURE ENHANCEMENT**

To make an event successful event manager needs different service provider like Sound systems services, Lighting providers, Canteen services, stage construction and so on. In present system event company have to do all management work manually. They keep all payment information on papers. There is no system to check the past expenses on any event. To do this they have to check payment register and this task is very time consuming and tiresome. Keeping all these problem in mind we have developed this system. This system helps the event management company to manage their paper work online and they can also retrieve report of last event they have completed.

**8.Conclusion:**

The aim is to achieve this by using the concept of Internet of Things (IoT), wherein an java sApplication is created for the customer, whose details are constantly updated by the hardware/server at the location. The features include unique identification for each vehicle, display of available parking slots on the mobile application, possibility of making reservations for the same, maintenance of a database.

This project primarily makes finding parking systems more efficient and less time-consuming, and also eliminates the need of a large staff to be employed for such purposes at these places, thus increasing the profit margins.

**9.Bibilography**

**1**.J. Gubbi, R. Buyya, S. Marusic, M. Palaniswami, "Internet of things (iot): A vision architectural elements and future directions", *Future Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1645-1660, Sep. 2013.

2. O. Bello, S. Zeadally, "Intelligent device-to-device communication in the internet of things", IEEE Systems Journal, vol. 10, no. 3, pp. 1172-1182, Sept 2016.

3. L. Atzori, A. Iera, G. Morabito, "The internet of things: A survey", *Comput. Netw.*, vol. 54, no. 15, pp. 2787-2805, Oct. 2010.

4. A. Ahmad, S. Phon-Amnuaisuk, Low cost parking space management system, vol. 5, pp. 385-393, 2016.

5. E. Sifuentes, O. Casas, R. Pallas-Areny, "Wireless magnetic sensor node for vehicle detection with optical wake-up", IEEE Sensors Journal IEEE. IEEE, 2011.