**CHAPTER 1**

**INTRODUCTION**

**1.1 Overview**

Recent advancement in the automobile industry has opted many people to use their own vehicle for travelling. This has increased effect on car ownership. But to park all these cars in the major metro cities is quite tedious and difficult. Parking problems are becoming ubiquitous and ever growing at an alarming rate in every major city. Lot of research and development is being done all over the world to implement better and smarter parking management mechanisms. Widespread use of wireless technologies paired with the recent advances in wireless applications for parking, manifests that digital data dissemination could be the key to solve emerging parking problems. There are numerous methods of detecting cars in a car park such as Magnetic sensors, Microwave Radar, Ultrasonic sensors and image processing.

Most car parks today are not run efficiently. This means that on busy days drivers may take a long time driving around a car park in order to find a free parking space. A recent survey says that during rush hours in most of the big cities, the traffic generated by cars searching for parking spaces takes up to 40% of the total traffic. Implementing the proposed system will help to resolve the growing problem of traffic congestion, wasted time, wasting money, excessive usage of paper and help provide better public service, reduce car emissions and pollution, improve city visitor experience, increase parking utilization, and prevent unnecessary capital investments. The system does this by providing more efficient and effective parking enforcement.

The term ‘Smart Parking’, refer to the technology which allows people to dynamically reserve and make payment for a parking slot, whereas a ‘System’ is a set of computer component or orderliness. By general understanding, a Smart Parking System is a combination of intelligent actions using a combination of hardware, software and data communications to efficiently manage a large car parking area. This parking system can be deployed where image processing is used to take the picture of the car number plate in order to extract the car number and store it in the database. This system concentrates on parking area problems or to be precise, parking area management.

**1.1.1 Image Processing**

In implementation of Smart Parking and path Retrieval System, Image Processing plays a very crucial role in management of parking. Image processing is used to extract the number from the car’s number plate image which is taken at the entrance of the parking area. The number will be stored in the database and automatically stored in the user’s car list. There are many methodologies and techniques used for image processing which are explained in this section. Morphological technique has been used in the proposed system.

**License Plate Recognition System based on Image Processing Using Labview**

A License plate recognition (LPR) system is one kind of an intelligent transport system and is of considerable interest because of its potential applications in highway electronic toll collection and traffic monitoring systems. A lot of work has been done regarding LPR systems for Korean, Chinese, European and US license plates that generated many commercial products. However, little work has been done for Indian license plate recognition systems*.* The purpose of this paper is to develop a real time application which recognizes license plates from cars at a gate, for example at the entrance of a parking area or a border crossing*.* The system, based on regular PC with video camera, catches video frames which include a visible car license plate and processes them. Once a license plate is detected, its digits are recognized, displayed on the user interface or checked against a database. The focus is on the design of algorithms used for extracting the license plate from a single image, isolating the characters of the plate and identifying the individual characters. The proposed system has been implemented using Vision Assistant 8.2.1 & Labview 11.0 the recognition of about 98% vehicles shows that the system is quite efficient.

**Vehicular Number Plate Recognition Using Edge Detection and Characteristic Analysis of National Number Plates**

VNPR using edge detection and character analysis propose a framework that uses a camera installed at roadside to detect the vehicle number plate. A typical video component requires several adjustments after image has been stored such as enhancement of image, localization of number plate, separating each character and recognition of each character. In this system it extracts the vehicle number plate from our image recognizes it based on the characteristics of number plates in different countries. It uses Sobel edge detection for plate localization, template matching and fuzzy logic for recognition. It makes use of the characteristics of the vehicle number sequences to further enhance performance. Vehicle number plate recognition can be used to decrease human effort by making systems automatic.

**Automatic car number plate detection using morphological Image Processing**

Automatic car number plate detection using morphological image processing describes a detection method of car number plates on digital images. The method inputs the gray tone image of a car and extracts from it the characters on the number plate. The proposed method makes use of the morphological operators: morphological reconstruction, filters based on it and top-hat operators. The method can be applied to access control systems supervising the car traffic in the restricted areas.

It describes an image processing method which allows detecting a car number plate on the image presenting a car entering the supervised area. The method is intended to be a first part of the identification process which consists also of a second part - the recognition of detected characters. Automatic car number plates’ detection is performed by means of mathematical morphology. It makes use of an intelligent filtering of the input image based on a set of filters removing unnecessary image elements, but preserving the position and shape of characters of the car number plate.

**Number Plate Recognition Using an Improved Segmentation**

The NPR (Number Plate Recognition) using is a system designed to help in recognition of number plates of vehicles. This system is designed for the purpose of the security system. This system is based on the image processing system. This system helps in the functions like detection of the number plates of the vehicles, processing them and using processed data for further processes like storing, allowing vehicle to pass or to reject vehicle. NPR is an image processing technology which uses number (license) plate to identify the vehicle. The objective is to design an efficient automatic authorized vehicle identification system by using the vehicle number plate. The system is implemented on entrance for security control of a highly restricted area like military zones or area around top government offices e.g. Parliament, Supreme Court etc. The developed system first captures the vehicle image. Vehicle number plate region is extracted using the image segmentation in an image. Optical character recognition technique is used for the character recognition. The resulting data is then used to compare with the records on a database. The system is implemented and simulated in Matlab, and it performance is tested on real image. It is observed from the experiment that the developed system successfully detects and recognize the vehicle number plate on real images.

The number plate recognition systems are today used in various traffic and security applications, such as parking, access and border control, or tracking of stolen cars. In entrance gate, number plates are used to identify the vehicles. When a vehicle enters an input gate, number plate is automatically recognized and stored in database and black-listed number is not given permission. When a vehicle later exits the place through the gate, number plate is recognized again and paired with the first-one stored in the database and it is taken a count. Automatic number plate recognition systems can be used in access control. For example, this technology is used in many companies to grant access only to vehicles of authorized personnel.

**Steps for Image Processing Techniques**

Following are the steps that are used while detecting a car’s number plate using the previously explained techniques

1. Contrast enhancement
2. Background cleaning
3. Plate area detection
4. Extraction of characters
5. **Contrast Enhancement**: - Due to variable lighting conditions the contrast is often not as high as necessary. An original image and the result of contrast enhancement are shown in fig below using morphological technique.
6. **Background cleaning**: - Its aims at removing the most of unnecessary areas on the image. The important requirement is a preservation of shapes of the letters of images, so that they can be correctly recognized in further steps of treatment. This feature was the motivation of applying the reconstruction filters in both top-hats methods involved in morphological technique
7. **Plate area detection**:-The result of combination of two top-hats contains not only letters of number plate but also lot of other unwanted image elements. This process has two phases: Rough detection and Filtering. The resulting image contains several light areas. The area of number plate is noticeably lighter than the other light fragments.
8. **Extraction of characters**: - The last step is extraction of single characters from the number plate. It must detect what kind of plate is currently being processed. To do this, two images results from computed in the second step are analyzed then binarized which contains only characters from number plate further extraction of single character is done and documented.

**1.2 Problem Statement**

According to a recent survey, searching for a vacant parking space in a metropolitan area during rush hours, the traffic generated takes up to 40% of the total traffic which also leads to time consumption and air pollution.

The current smart parking or parking guidance systems only obtain the availability information of parking spaces from deployed sensor networks. But in our smart parking system, it will help people to pre-booking the parking slot in the parking area so that space will be reserved for the particular car. The system will capture the image of the number plate and it also provides the path to the allotted slot and using the same, we can also retrieve the location of the car.

**1.3 Objectives**

The objective is to reduce congestion in parking space as well as to reduce traffic outside the parking area.

* Providing reservation facility for the people who want to pre-book the parking space
* Reservation is based on parking slot numbers, so that wireless sensors will be eliminated where its installations are costly
* Providing graphical information of path to users in ease finding of their vehicles
* Arrival and exit of vehicle will be monitored by image processing techniques by capturing Image of number plate
* To reduce paper usage in parking areas

**1.4 Limitations**

* Internet connectivity is a must
* Maintenance cost is comparatively high
* Font sizes and font styles may vary for different cars

**1.5 Literature Survey**

Searching for a vacant parking space in a metropolitan area is the daily concern for most drivers, and it is time consuming. It commonly results more traffic congestion and air pollution by constantly cruising in certain area only for an available parking space. For instance, a recent survey shows that during rush hours in most big cities, the traffic generated by cars searching for parking spaces takes up to 40% of the total traffic. To alleviate such traffic congestion and improve the convenience for drivers, many smart parking systems aiming to satisfy the involved parties (e.g., parking service providers and drivers) have been deployed. This chapter presents different approaches and the related work done in the field of Smart Parking.

To improve the parking facilities by the introduction of new smart parking systems that would reduce the empty parking space searching time [1]. Most of the recent parking technologies relies on intrusive sensor to detect empty parking space and did not specifically guide patrons to a specific parking lot. Therefore here, they have proposed the implementation of Smart Parking Systems using Image Processing Techniques, Wireless Sensor network and Shortest Path Algorithm in order to help patrons finding vacant Parking spaces. The pre-existing Security surveillances (CCTV’s) will be used as Sensing Nodes to identify the vacant parking space. The image captured will be processes through the RabbitCore MicroController and the processed data will be transmitted via Zigbee to a central computer to store and update the occupancy status of a available parking spaces vacancies in the database. The system will be automatically assigned a space to patrons using A-star (A\*) shortest path algorithm based on the point of nearest entrance of building. The patron will be guided to the specified location by referring to variable massage sign and map printed on parking ticket. RFID technology will also be used uniquely tag the reserved ID to the database. The information on the ID will be used to remind patrons of their parking location during payment upon leaving.

To cope with the ever growing problem of traffic management and parking management this technique proposes an advance solution for managing and monitoring free parking space and automated guidance for user to park the car. Managing of car parking system using wireless sensor network aims at implementing smarter and better parking guidance mechanism which reduces significantly vehicle travel time and parking time [2]. In this system all the Infrared sensor nodes (IR sensor) sense the status of the car space and accordingly transfer the information to the AVR controller. Accordingly AVR sensor sense the status of car parking space and displays the information on the LED screen for the user, thereby reducing the time for the driver to find vacant empty space and almost reduce the chances of entering into the unusual space which might lead into the traffic jam.

Smart Parking Systems obtain information about available parking spaces, process it and then place the car at a certain position [3]. A prototype of the parking assistance system based on the proposed architecture was constructed here. The adopted hardware, software, and implementation solutions in this prototype construction are described in this technique. The effective circular design is introduced here having rack-pinion special mechanism which is used to lift and place the car in the certain position. The design of rack pinion mechanism is also simulated using AUTODESK INVENTOR and COMSOL software.

A multi-level car parking is essentially a building with number of floors or layers for the cars to be parked. The different levels are accessed through interior or exterior ramps. An automated car parking has mechanized lifts which transport the car to the different levels at a certain position. Therefore, these car parks need less building volume and less ground space and thus save on the cost of the building. Car parking is an issue of significance both at the local and at the strategic level of planning. Here a circular arrangement parking model is introduced where the cars are lifted from the base. The base can rotate circularly and a rack-pinion mechanism is set on the base which is used to lift the cars to the parking chamber at different level. Here the parking system is designed for two levels and the angle of basement is about 300 to 400. Different study and research had been carried out in this regard. But it is difficult to find the total study for the complete structure of an automated multilevel parking system. Rather, most of the research work was related to assist the car for parking in the specified position with the utilization of RFID systems or other networking systems.

The main objectives of the car parking system are- to design and fabricate a multi-level car parking system and to design and fabricate a cost-effective model, to develop a fully

automated control system and to prevent illegally parked vehicles.

Intelligent Parking Management System Based on Image Processing a secure parking reservation system using Global System for Mobile communications (GSM) technology has been performed [4]. It helps the drivers from facing the problem that always occurs at the car park, such as time being wasted in searching for the available parking spaces and keep on circling the parking area until they found an empty parking spot. This problem usually occurs in urban areas, where number of vehicles is higher as compared to the availability of parking spaces. In this proposed system there are two modules have been developed such as parking lot monitoring and security reservation modules. For the parking lot monitoring module, the layout animation is used to display the parking lot status. In security reservation module, the users need to send SMS to reserve the parking lot and use the password provided to enter and exit the parking lot. Our main contribution is the system has more security than other previous research with the used of password to enter and exit.

With the increase of economic behavior and the upgrade of living standard, the ratio of people in India who own automobiles and motorcycles have recently increased giving a boost to Metropolitan Traffic. Therefore, parking issues will be a big challenge to facilitate traffic network and ensure urban life quality. Searching for parking space in most metropolitan areas, especially during the rush hours, is difficult for drivers. The difficulty arises from not knowing where the available spaces may be at that time; even if known, many vehicles may pursue very limited parking spaces to cause serious traffic congestion. A Reservation-based Smart Parking System designs and implements a prototype of Smart Parking System based on Reservation (SPSR) that allows drivers to effectively find and reserve the vacant parking spaces [5]. By periodically learning the parking status from the host parking database management in parking lots, the reservation service is affected by the change of physical parking status. The drivers are allowed to access this cyber-physical system with their personal communication devices. Furthermore, we study state-of-the-art parking policies in smart parking systems and compare their performance. The experiment results show that the proposed reservation-based parking policy has the potential to simplify the operations of parking systems, as well as alleviate traffic congestion caused by parking searching.

Smart parking based on reservation proposes a smart parking system to solve the problem of unnecessary time consumption in finding parking spot in commercial car park areas [6]. A parking reservation system is developed in such a way that users book their parking spots through short message services (SMS). The SMS sent will be processed by a wireless communication instrumentation device called micro-RTU (Remote Terminal Unit). This micro-RTU will reply the confirmation of booking by giving the details of reservation like password and lot number. The password will be used to enter the parking area and valid for a certain period of time. The system is fully automated with the use of the Peripheral Interface Controller (PIC). This microcontroller is capable in storing information of empty parking spaces, provide passwords as well as allowing or denying access to the parking area. A prototype of a car park system has been designed to demonstrate the capability of the proposed work. The demonstration has proven the capability of the system to reserve the parking, gain entry to the parking area and hence eliminates the hassle of searching empty parking lots.

The number of personal vehicles usage is increasing manifold. People prefer personal vehicles to commute than depend on public transportation. Finding a parking space in most metropolitan areas, especially during the rush hours, is difficult for drivers. Due to this there is a need to provide sufficient parking places coupled with plenty of slots to help the user park his vehicle safely, also to ensure the user does not end up parking on non-parking area and cause discomfort to pedestrian. The idea behind Android Application for Vehicle Parking System: Park Me is to help the user analyze area’s where parking is available and number of slots free in that area [7]. Additionally, four hours prior to his expected arrival, the user can pre-book a slot in the area he desires if it is available. This will help reduce the load on the administrator as his physical work reduces drastically and user can search the parking slot through Android Application. Payment services are made available using Google Wallet, so the user is required to own a credit card or debit card. “Park Me” Application relieves the user from the hassle of manually searching and waiting for empty slots to park the vehicle.

Number Plates Recognition (NPR) has since found wide commercial applications, making its research prospects challenging and scientifically interesting. A complete NPR system functions by vz steps, license plate; localization, sizing and orientation, normalization, character recognitions and geometric analysis. This paper is a review of NPR preliminary stages; it explains number plate localization, sizing and orientations as well as normalizations sections of the Number Plates Detection and Recognition-Tanzania Case study. MATLAB R2012b is employed in these processes. The input incorporated includes front and rear photographic images of vehicles, for proximity and simulation purposes the ample angle of image is 90 degree +-15. The captured image is converted to gray scale, binarized and edge detection algorithms are used to enhance edges. The output of this stage provides the input feature extraction, segmentation and recognitions. The main goal of project is to study, analyze and design an efficient and optimized algorithm(s) for detection and recognition of Tanzanian cars by their plate numbers [8]-[12]. The algorithm(s) will provide accurate and timely alphanumeric recognition performance. The application of the algorithm(s) will facilitate automation capturing and recognition of vehicle plate number, additionally the license plate characters will be captured with information available in respective database, example Tanzania Revenue Authority database or parking lot database.

**1.6 Organization of the Report**

The organization is as follows. Chapter 1 introduces basic concepts. Chapter 2 enumerates the literature survey. Chapter 3 discusses the design of architecture. Chapter 4 discusses the implementation. Chapter 5 emphasizes on testing. Conclusion and future enhancement is stated in Chapter 6. Glossary provides the keywords and Acronyms gives the expansion and Bibliography gives the papers referred respectively.

**CHAPTER 2**

**ANALYSIS**

**2.1 Existing System**

**2.1.1 Description**

The current parking guidance systems obtain the availability of parking spaces using the sensors installed across the whole parking lot. However, deploying sensors in a large parking lot can be very expensive. Furthermore, the sensors can become inaccurate and would stop functioning easily when time passes. Therefore, it is highly desired to have a reliable and cost effective way to track available parking spaces and guide drivers to the available parking spaces. Besides searching for available parking spaces, vehicle theft in large parking lots also has become a serious concern facing our lives. For example, statistics show that there have been over 170,000 vehicles stolen each year in Canada. In some of the existing system, the user can book the slot in case of free slots are available via SMS and a password is given to each user. On entering the password at the entry gate the user is able to park the car in a secure way. The system ensures security and reservation of slot but doesn’t provide any guidance to reach the designated slot as shown in figure 2.1.

There is no proper validation whenever the user enters the parking area, whether they have already reserved a slot or not. And in some systems, the user has to reserve a slot 4 hours in prior. This system is lack of security for the exit process because the car can easily exit through the sensor detection. It is easy for the unauthorized person to take any car out.

Whereas, some of the system also uses Infrared (IR) sensor nodes sense the status of the car space and transfers the information to a controller. It thereby displays the information on a LED screen with which the user can check for empty vehicle slots, in turn reducing his time. As infrared cannot penetrate walls, therefore it cannot be used in closed parking areas due to low wavelength. (E.g. shopping malls or residential area where parking is done in enclosed area). A multi-storey car-park is a building designed for car parking and where there are a number of floors or levels on which parking takes place as shown in figure 2.2. It is essentially a stacked car park.



**Figure 2.1** Entrance of Parking Area with no Proper guidance to Parking Slot



**Figure 2.2** Multi storey Car Parking Areas

**2.1.2 Drawbacks**

* They are not cost effective because they use Wireless Sensors or Ultrasonic Sensors.
* There is no proper guidance to the destination slot from the entrance of parking area to the user.
* An infrared sensor node like wireless sensor doesn’t work inside closed parking areas.
* Image processing techniques are used to evaluate whether parking space is available or not inside the parking area, but it doesn’t reduce traffic in metropolitan cities as shown in figure 2.3.

**Figure 2.3** Traffic due to non availability of parking spaces

**2.2 Proposed System**

**2.2.1 Description**

Parking problems are becoming ubiquitous and ever growing at an alarming rate in every major city. E-Parking systems provides a way for drivers to reserve a parking space at their desired car park facility of choice to ensure a vacant parking space is available due to pre-booking, when they arrive at the parking area. The Smart Parking System provides the process of pre-booking the parking space by using a simple and interactive web application. This application is expected to provide an efficient and cost-effective solution to the effluent vehicle parking problems.

The reservation of the parking space can be done by using a Website which is also mobile interface compatible, with the recent advances in web applications for parking, manifests that digital data dissemination could be the key to solve emerging parking problems. The graphical interface is displayed to the users of the system to ease the processes like parking space pre-booking, dynamic parking slot allotment, employee and user profile management, system function statistics etc.

**User Interface**

In order to do user operations, he must have a device with internet connectivity to reap the facilities of this application. The user must mandatorily register with the application, that is, the user should signup. The user can add or remove information about cars that he owns, find locations of different parking locations available, editing of user profile, pre-booking of a parking slot etc.

The user can update his profile by entering his personal details like email id and phone number and also car number details. He can select any desired parking area to book a parking slot. Location guide is provided using Google maps to easily locate the parking area. Once the user books a parking space, a buffer time of 30 minutes is allotted. User has to arrive at the parking area within the allotted time period else the booking will be cancelled. However, the user can get a dynamic parking slot also if he arrives at the parking lot without

pre-booking the parking slot.

Once the booking is confirmed, user can view a graphical path indicating the way to the allotted parking slot which makes it easier for the user to locate the parking slot in a large parking area. On return, the user also can view the path back to the parking space where he had parked his car. This facility provided by the system is called as car parking path retrieval.

**Employee Interface**

Once the user arrives at the parking area, the role of the employee comes into picture. An employee interface provides functions like updating their accounts, manage pre-booked scenarios, payment procedures, adding slots to registered parking areas, analysis of parking reports, studying the parking statistics, making fresh dynamic bookings for the users at the entrance of the parking area, etc. Major functionality of the employee is to take image of the car number plate and uploading it to the database. At the backend, the image gets processed and the number alone is extracted from the image and stored in the database.

In case any pre-booking of the parking space is done, then the employee has to just confirm the booking and let the user to park the car in their desired parking slot. If there is no pre-booking done, a photo of the car number plate is taken and uploaded into the system for storing the number into the database for future reference. A timer is started as soon as the booking is confirmed. The employee can generate a bill and collect money when the user is leaving the parking lot.

**Admin Interface**

The Admin controls the whole system functionality and monitors every activity. An Admin will have special privileges like adding or removing parking areas, viewing employee profiles, assigning employees to parking locations for work, adding or removing slots from parking area, getting reports of performance of the system in terms of amount collected, changing parking rates of a parking area etc.

Admin can alter his own profile and also employee profile. He can perform all the functionalities that an employee has access to. The admin has to change the parking rates carefully according to the trending rates where users feel it affordable and comfortable to use the parking facility. He can also view the User Log, Payment Log and Transaction Log.

**2.2.2 Advantages**

* Reduces congestion inside the parking space
* Pre-booking facilities for parking slots
* User-friendly which guides customer to their respective slot
* Minimization of human work

**2.3 Requirements Specifications**

Requirement in system engineering and software engineering compasses those tasks that go into determining the needs or conditions to meet for new or altered products, taking account of possibly conflicting requirements of the various stakeholders, such as beneficiaries or users. Requirement analysis is a critical to the success of a development of a project. Requirement must be actionable, measurable and testable related to identified business needs or opportunities and defined to enable a detailed sufficient for system design.

Conceptually, requirements analysis includes 3 types of activities:

**Eliciting Requirements**: The task of communicating with customers and users to determine what their requirements are. This is sometimes also called requirement gathering.

**Analyzing Requirements**: Determining whether the stated requirements are unclear, incomplete, ambiguous or contradictory and then resolving these issues.

**Recording Requirements**: Requirements maybe documented in various forms, such as natural language documents, use-case, user stories or process specification.

The software requirement specification phase is an important step comes in software development lifecycle and it is a complete description of the behavior of the system to be developed. SRS should describe the general factors that affect the product and its

requirements. The purpose of Software Requirement Specification (SRS) document is to specify the user goals and tasks that needed to be achieved. It must also include detailed description of the context and functional requirements, which are vital to the success completion of the project. Apart from this, the SRS document incorporates in itself the constraints and assumptions made during the course of the project. Requirement must be measurable, testable, related to identified needs or opportunities, and defined to a level of detail sufficient for system design.

**2.3.1 Functional Requirements**

The functional requirements for a system describe the functionality or services that the system is expected to provide. These depend on the type of the software that is being developed and type of the system that is being developed. Functional requirement capture the intended behavior of the system. It is organized into 3 modules:

**Admin/Certificate Authority Module**

The Admin module functions as follows:

* Admin/Certificate authority is main module.
* Admin has the authority to add and manage parking of various parking areas details.
* Admin can add and manage other admins and employee where each employee is allotted only one parking area.
* Certificate authority can also add and manage user’s details who directly come for parking without pre-booking.
* All the transaction logs, payment logs are under the control of admin and can check the feedback received from employees and users.

**Employee**

The employee module functions as follows:

* Employee has access to the entire parking area which is allotted to him by the admin
* The entire pre-booking process is managed only by the employee.
* Employee can register and manage booking for the cars which come directly to parking area.
* He can also add and manage users and also manage the parking charges for every user.
* The facility to give feedback to the admin by the employee is also available.

**User**

The User module functions as follows:

* User has to register himself and also his vehicle before using this application.
* He can add any number of vehicles in his account and also refer or register a friend.
* A user has been provided facility to find the location of the parking area with help of Google maps using this application.
* User can also select the parking slot of his choice well in advance by using pre-booking facility.
* He can edit his account details and check his previous booking details if necessary.
* Facility to give feedback is provided.

**2.3.2 Non Functional Requirements**

Non functional requirements are the functions offered by the system. It includes time constraints and constraints on the development process and standards. The non functional requirements are as follows:

**Speed**: The system should process the given input into output within appropriate time.

**Ease of use**: The software should be user friendly. Then the customers can use easily, so it doesn’t require much training time.

**Reliability**: The rate of failures should be less then only the system is more reliable.

**Portability**: It should be easy to implement in any system.

**2.3.3 Hardware Requirements**

|  |  |
| --- | --- |
| Hardware | Minimum Requirement |
| Processor | Pentium 4 |
| RAM | 1 GB |
| Hard-disk Space | 80 GB |
| Camera | 2 |

**Table 2.1** Hardware Requirements

**2.3.4 Software Requirements**

|  |  |
| --- | --- |
| Software | Minimum Requirement |
| Operating system | Windows 7 and above |
| Web Browser | Google Chrome |
| Technology | Java |
| Tools | * Eclipse ID * Apache * Tomcat * MySQL Server |

**Table 2.2** Software Requirements

**CHAPTER 3**

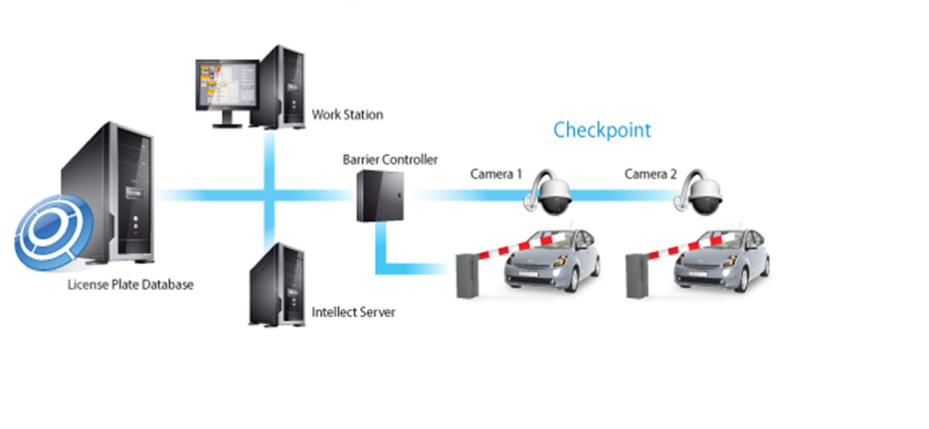
**DESIGN**

**3.1 System Design**

**3.1.1 Design Overview**

Software design is a process of problem solving and planning for a software solution. After the purpose and specification of software is determined, software developer will design or employ designers a plan for a solution. High level design gives an overview of the system flow. However, this describes a lot for the user to understand the logic. Here we see the basic knowledge about the system design and architecture.

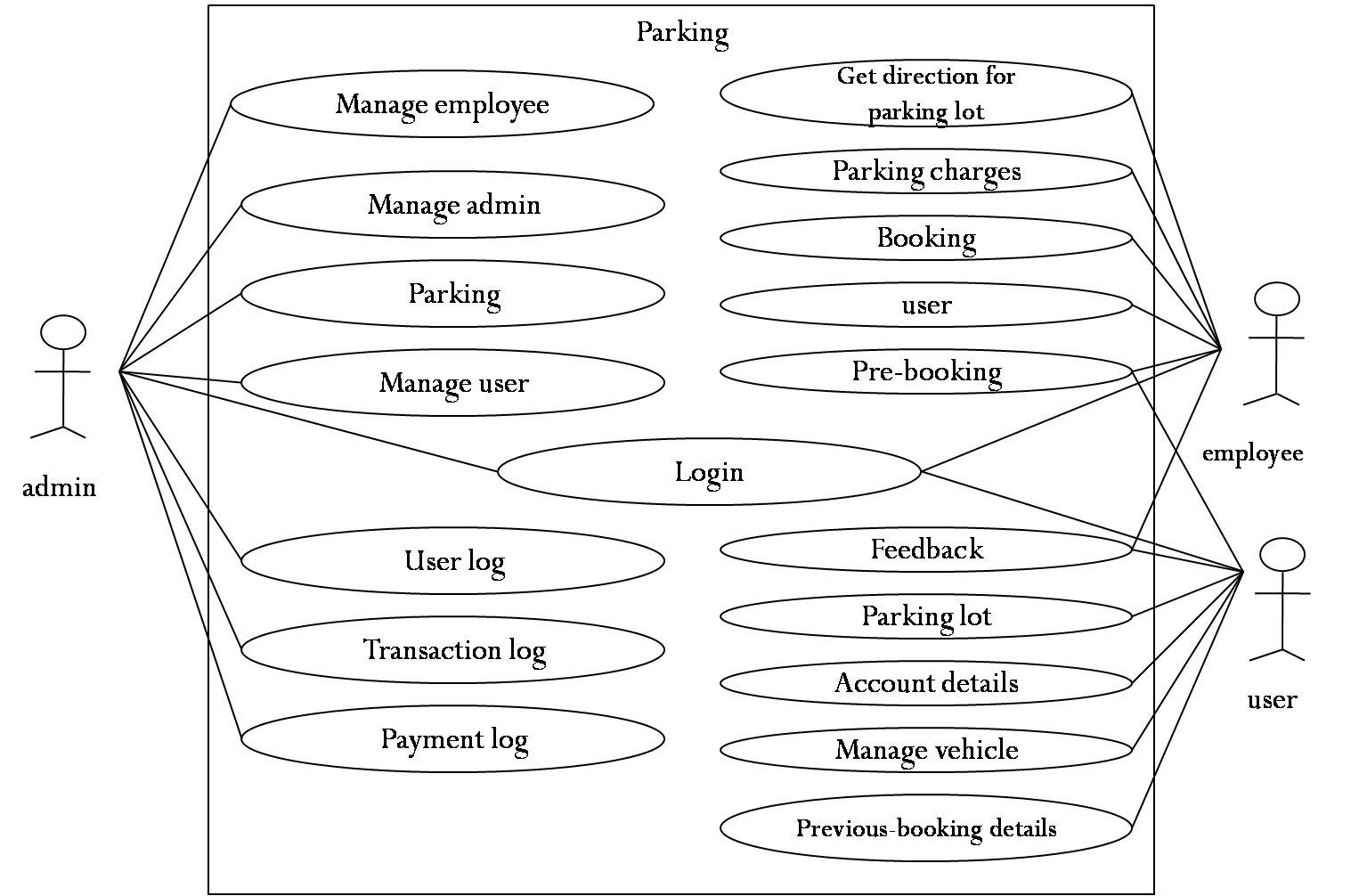
System design is processes through which requirements are translated into are presentation of software. Design provides us with presentation of software development. Design is the only phase where user requirements are accurately translated into finished software product or system. It refers to modeling of a process. It is an approach to create a new system. It can be defined as a transition from user’s view to programmer’s view. The system design Phase acts as a bridge between the required specification and the implementation phase as shown in figure 3.1.

****

**Figure 3.1** System Design

**3.1.2 Use Case Diagram**

* A use case diagram as its simplest is a representation of a user’s interaction with the system and depicting the specification of a use case.
* A use case diagram contains four components:
* The boundary, which defines the system of interest in relation to the world around it.
* The actors, usually individuals involved with the system defined according to their roles.
* The use cases, which the specific roles are played by the actors within and around the system.
* The relationships between and among the actors and the use cases.



**Figure 3.2** Use Case diagram for the System

The description of figure 3.2 is as follows.

**Admin can perform following functions:**

1. Parking

In this feature, admin is provided with two options, one is he/she can add new parking area and he can also manage the existing parking area.

1. Add Parking- Here, the admin has to provide all the valid details of the parking area like address, latitude, longitude, pin code, commission and the parking map.
2. Manage Parking- Here it displays all the parking areas which are registered and the admin can edit/remove the parking details of any parking area, he can also manage slots of any parking area.
3. User

By using this feature, admin can add a new user and can also manage the existing user.

1. Add User- Admin is provided with rights to add new user, new employee or an additional administrator, by providing all the details of user like name, email id, mobile number, username and password and an optional profile picture.
2. Manage User- Here it displays a table which consists information of all the existing users and the admin can edit/remove the details of user/employee/administrator.
3. User Log

This feature displays a table which consists of user details, personal details, login time and user type. Here the admin can check which all the users had login and their details. In this feature we have given user type, where employee and user type is 1 and administrator is type 3.

1. Transaction Log

This feature also displays a table which consists of transaction details i.e. a confirmation e-mail sent to the users when pre-booking is done. The table consists of User details, Transaction type, Sent time and Description of

the email.

1. Payment Log

This feature displays a table which consists of payment details done by the user. It consists of information such as user details, payment details, payment type, amount and the parking name.

Another special feature provided to the admin is he/she can receive a feedback from an employee or a user. A table format which consists of username, email id, subject and description of the feedback is displayed.

**The features that are included in Employee login are as follows:**

1. Get Direction to Parking Lot

The employee can get the directions which will be displayed to the visitors at the parking lot after their booking is confirmed. The directions are showed using a graphical path, an image with a pointer pointing at the slot that has been booked for the visitor. This helps the driver to find the path to the slot which is booked for his/her car. There is a dropdown menu from which the employee can select the car for which the directions has to be displayed.

1. Pre-Booking
   1. Register Pre Booking- The employee can take an image of the car number plate and upload it. Once he selects, the page will be redirected to a page where he can manually select a slot for the visitor. In the back-end the car number would be stored in the database by processing the image. After selecting a slot, the page is redirected to Manage Pre Booking page.
   2. Register Pre Booking Manual- Employee can select a user from the dropdown menu, who are already registered and are active users of the application. Once the user is selected, the employee has to select the car which the user has registered under his/her profile. After the car is chosen, the employee gets to choose a desired slot for the user and once the submit button is pressed, it redirects to Manage Pre Booking

page for further actions.

* 1. Manage Pre Booking- When the user arrives at the parking lot and has pre booked a slot, the employee can check in the pre booked car list and confirm booking. If the employee confirms booking of a user, it is redirected to Manage Booking page.

1. Booking
   1. Register Booking- This is a facility for the employee to make a booking of parking slot, where he/she can make a new booking by taking an image of the car number plate or make a new booking manually by selecting a user who is already registered or he/she can select a user from the pre booked cars list and confirm their booking. Once the booking has been confirmed, the user will get an e-mail notification immediately.
   2. Checkout Vehicle- When the visitor wants to go out of the parking lot with his car, again the employee can just take an image of the car number plate and upload it. The system matches the car number and generates a bill for that particular user and also it redirects to Manage Booking page. Image processing happens in the back end and the number will be compared with the car numbers that are stored in the back end already. This makes the checkout process easier. An e-mail notification will be sent to the user after the bill is paid.
   3. Manage Booking- The employee can view the list of all the details of the cars whose booking gas confirmed and the ones which are already parked in the parking lot. He/she can generate bill for the users who are leaving the parking lot. It also displays the amount that the user has to pay or has already paid.
2. User
   1. Register User- If there are any new users who have arrived at the

parking lot, the employee can register the new user and create a new profile for the user. Further, the user can use the application to pre-book the parking slot.

* 1. Manage User- The employee can manage user details under this category. He/she can alter user details, especially car details which the user has registered in his profile. This is helpful as the burden on user to change the vehicle details in his/her profile will be eliminated.

1. Parking Charges
   1. Add Parking Charges- Parking charges of parking lots may vary depending upon the area, demand for parking slot, accommodation capacity of the parking area etc. The employee can define a parking rate for specific time for the parking lot to which the admin has assigned him. This will redirect to Manage Parking Charges page.
   2. Manage Parking Charges- Here the employee can view a list of all the parking charges that were previously allotted. Also he/she can edit and make changes to the parking rates that are mentioned already.
2. Feedback

Employee has the privilege to give feedback about the system that he is using. He/she can mention if any difficulties have occurred or if any issues were identified while using the system.

**User can perform the following activities:**

1. Parking Location

If the user does not know the directions in the city to a particular parking area, then he/she can refer to Google maps which are provided in this section of the application. The user has to select a parking lot which is registered and the page is redirected to a page which contains Google map showing the directions. Internet connectivity in user’s device would be required for this facility to work.

1. Pre Booking

There is a category called Register Pre-Booking under this section. User can pre-book a parking space using this feature. User has to select a parking location then

the vehicle that he is taking. It will get redirected to a page where the user can book a

parking space of his/her choice. Once the pre-booking has been made, the user must go to the desired parking lot within 30 minutes, else the booking gets refreshed.

1. Booking Details

The user can check his/her previous booking details under this feature. All the previous bookings are listed along with date, time, parking lot selected and payment information.

1. Account Details
2. Edit My Account- The user can edit his account details which will reflect in his profile. He/she can also edit username and password here in this section.
3. Refer of Register Friend- A user can refer a friend who might want to use this application. This can reduce the burden of the user who is being referred. An e-mail will be sent to that person when a user refers him/her.
4. Vehicle
   1. Add Vehicle- User can add a detail about his vehicle in his profile by specifying details like car model and car number. It gets redirected to Manage Vehicle Page.
   2. Manage Vehicle- When the car has been parked in a desired parking area, the user can keep track of his vehicle using this feature if his vehicle is being used by someone else.

**3.2 Detailed Design**

**3.2.1 Class Diagram**

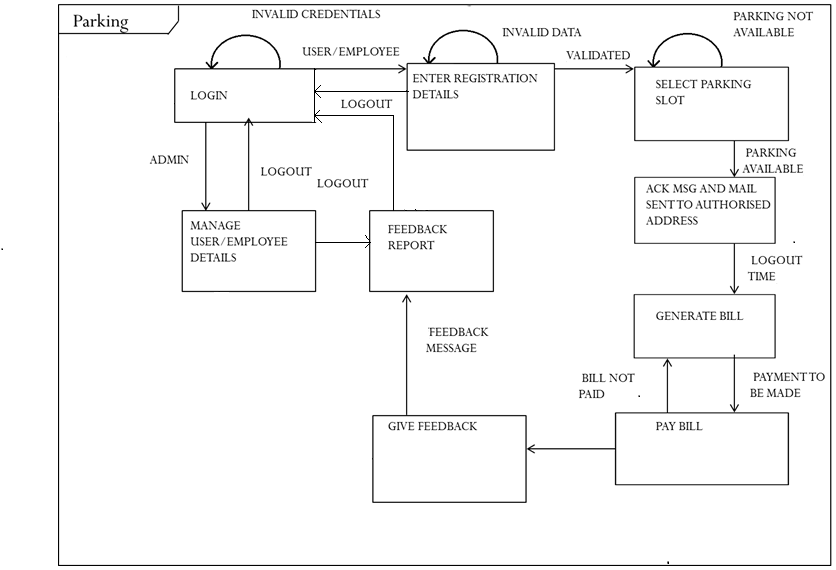
**Figure 3.3** Class Diagram for User

**Figure 3.4** Class Diagram for Pre-Booking

**Figure 3.5** Class Diagram for Booking

**3.2.2 State Chart Diagram**

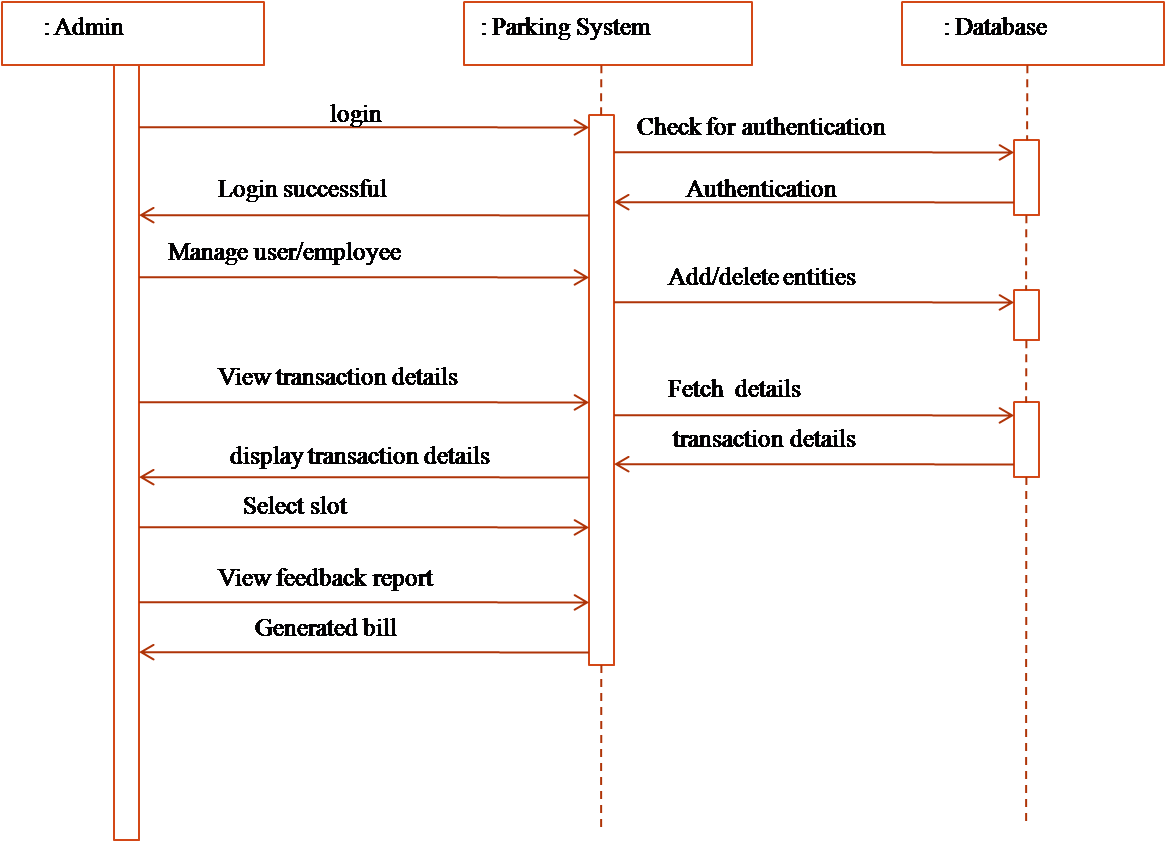
A state diagram is a type of [diagram](http://en.wikipedia.org/wiki/Diagram) used in [computer science](http://en.wikipedia.org/wiki/Computer_science) and related fields to describe the behavior of systems. State diagrams require that the system described is composed of a finite number of [states](http://en.wikipedia.org/wiki/State_(computer_science)); sometimes, this is indeed the case, while at other times this is a reasonable [abstraction](http://en.wikipedia.org/wiki/Abstraction_(computer_science)). Many forms of state diagrams exist, which differ slightly and have different [semantics](http://en.wikipedia.org/wiki/Semantics#Computer_science). The state chart diagram of our system is represented in figure 3.6.



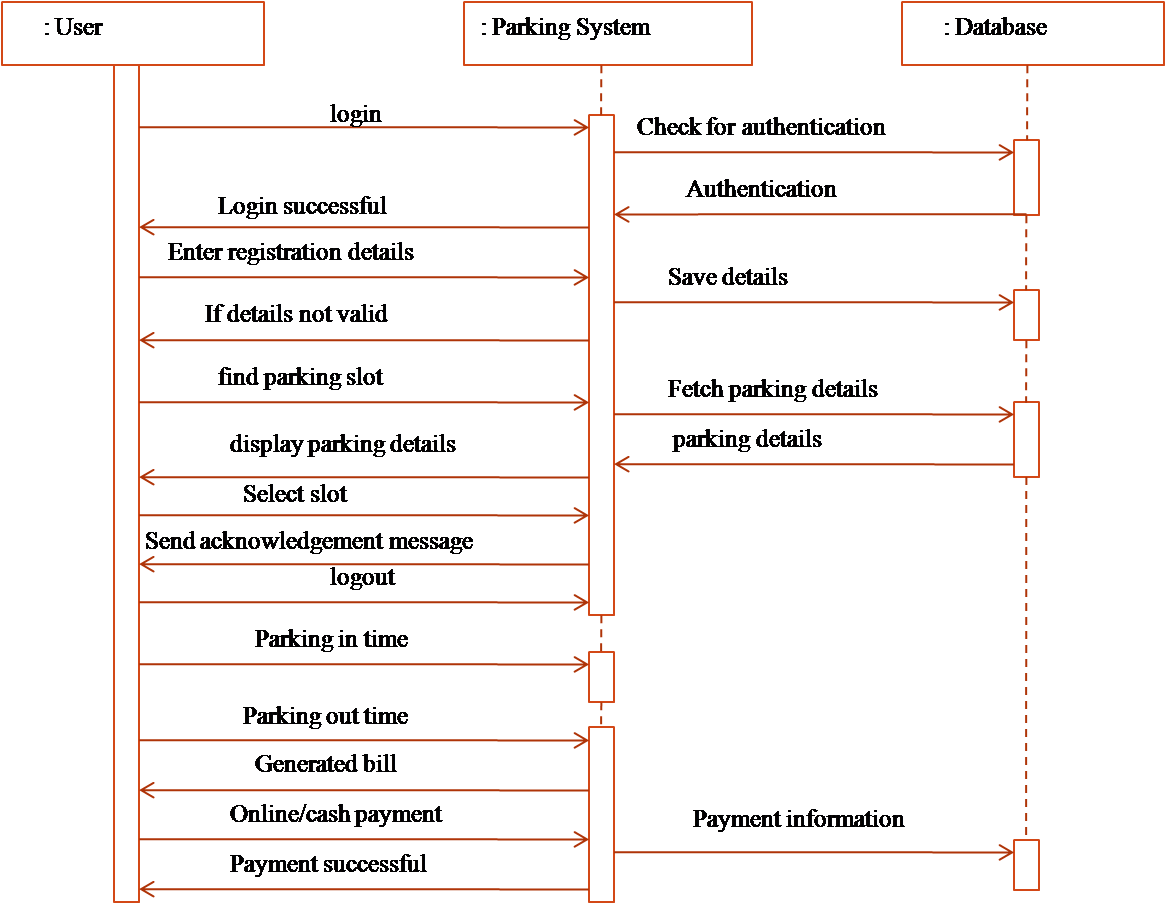
**Figure 3.6** State chart diagram

**3.2.3 Sequence Diagram**

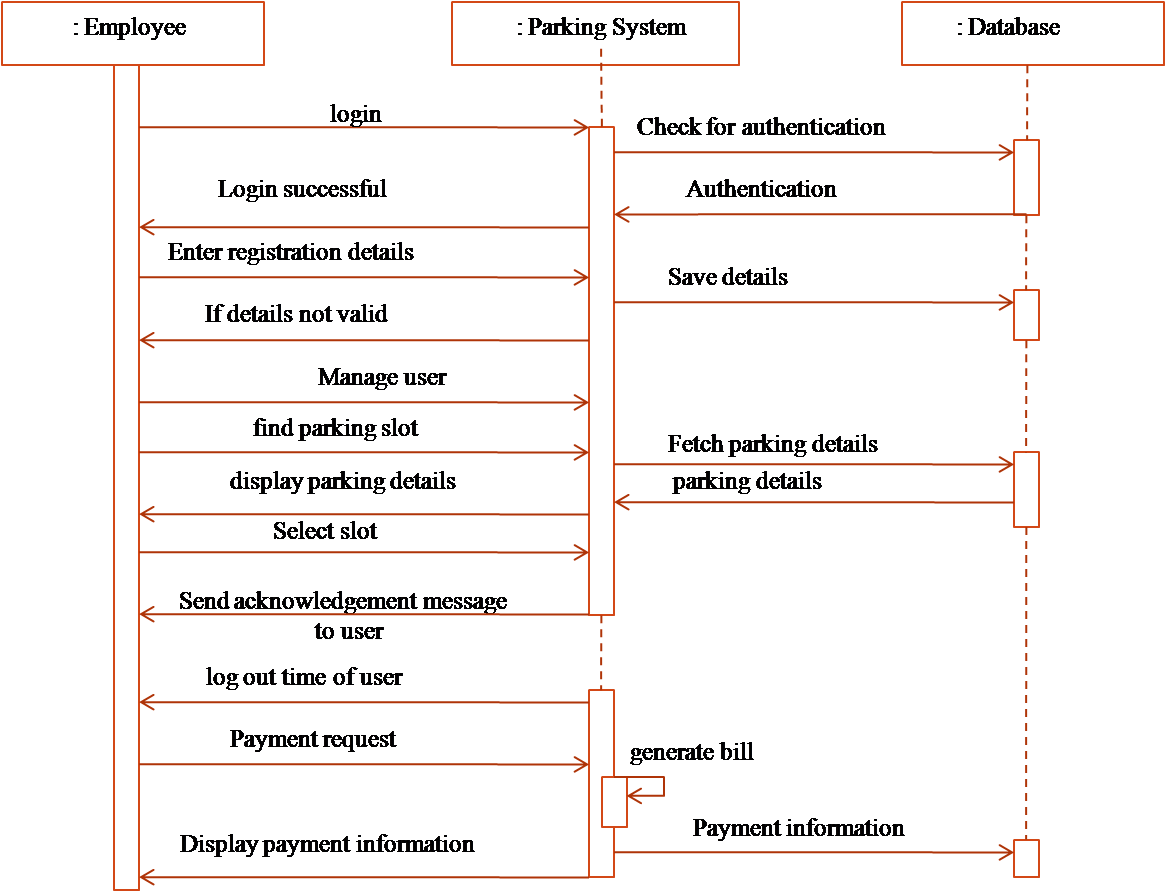
* A sequence diagram is a kind of interaction diagram that shows how processes operate with one another and in what order
* It is a construct of a message sequence chart
* Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.
* A sequence diagram is a interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart
* A sequence diagram shows object interactions arranged in time sequence.



**Figure 3.7** Sequence diagram for Admin



**Figure 3.8** Sequence diagram for User



**Figure 3.9** Sequence diagram for Employee

**3.2.4 Activity Diagrams**

Activity diagrams are graphical representations of [workflows](http://en.wikipedia.org/wiki/Workflow) of stepwise activities and actions with support for choice, iteration and concurrency. In the [Unified Modeling Language](http://en.wikipedia.org/wiki/Unified_Modeling_Language), activity diagrams are intended to model both computational and organisational processes (i.e. workflows). Activity diagrams show the overall flow of control.

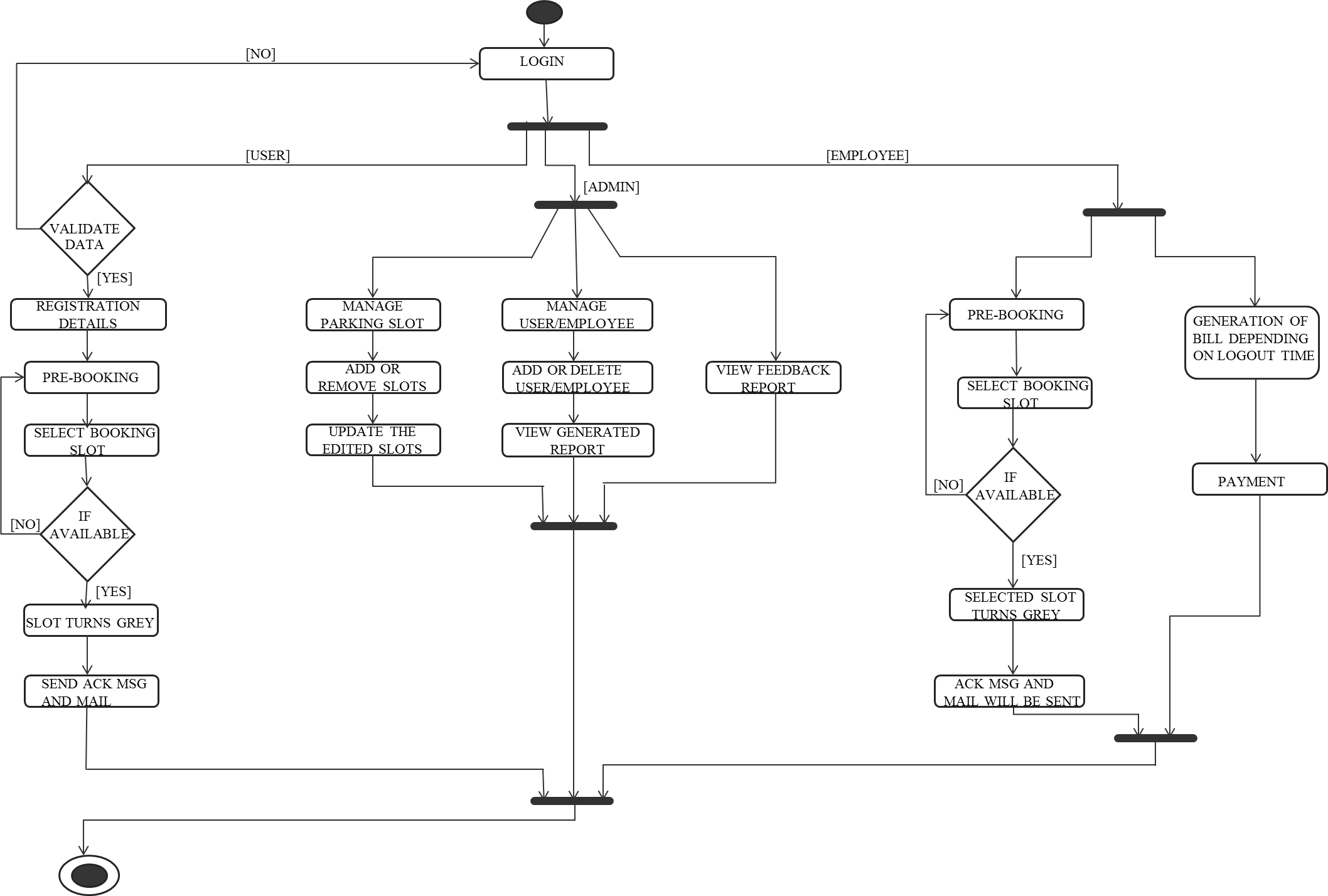
Activity diagrams are constructed from a limited number of shapes, connected with arrows. The most important shape types:

* *rounded rectangles* represent *actions*;
* *diamonds* represent *decisions*;
* *bars* represent the start (*split*) or end (*join*) of concurrent activities;
* a *black circle* represents the start (*initial state*) of the workflow;
* an *encircled black circle* represents the end (*final state*).

Arrows run from the start towards the end and represent the order in which activities happen.

Activity diagrams may be regarded as a form of [flowchart](http://en.wikipedia.org/wiki/Flowchart). Typical flowchart techniques lack constructs for expressing concurrency. However, the join and split symbols in activity diagrams only resolve this for simple cases; the meaning of the model is not clear when they are arbitrarily combined with decisions or loops.

While in UML 1.x, activity diagrams were a specialized form of state diagrams,in UML 2.x, the activity diagrams were reformalized to be based on [Petri net](http://en.wikipedia.org/wiki/Petri_net)-like semantics, increasing the scope of situations that can be modeled using activity diagrams. These changes cause many UML 1.x activity diagrams to be interpreted differently in UML 2.x.



**Figure 3.10** Activity diagram for the entire system

**CHAPTER 4**

**IMPLEMENTATION**

**4.1 Main Packages**

* **java.io.IOException**: This package provides classes that handle exceptions thrown when input or output error occurs.
* **java.util.List**: This interface provides an ordered collection meaning can access the elements of a list in a specific order.
* **java.util.Calendar**: This package provides methods for converting a specific instant of time and a set of calendar fields such as year, month, day, hour, and so on.
* **java.util.Date**: This class provides that encapsulates the current date and time.
* **java.util.ArrayList**: This class provides resizable array and implements the list interface, all optional list operations.
* **java.util.Iterator**: This interface iterates over a collection.
* **javax.servlet.ServletException**:This class defines a general exception a servlet can throw when it encounters difficulty.
* **javax.servlet.ServletRequest**: This interface defines an object to provide client request information to a servlet. The servlet container creates a ServletRequest object and passes it as an argument to the servlet's service method.
* **javax.servlet.ServletResponse**: This interface defines an object to assist a servlet in sending a response to the client. The servlet container creates a ServletResponse object and passes it as an argument to the servlet's service method.
* **javax.servlet.http.HttpServletRequest**: This interface extends the ServletRequest interface to provide request information for HTTP servlets.
* **javax.servlet.http.HttpSession**: This interface provides a way to identify a user across more than one page request or visit to a web site and to store information about that user.
* **javax.servlet.http.HttpServlet**: This class is an abstract class to be subclassed to create an HTTP servlet suitable for a web site.
* **java.io.File**: This package provides classes for system input and output through data streams.
* **org.hibernate.Session**: The main runtime interface between a java application and hibernate.
* **org.hibernate.SessionFactory**: Creation of session instances. Threads servicing client requests obtain session instances from this factory.
* **org.hibernate.cfg.Configuration**: This package represents an entire set of mappings of an applications java types to an SQL databases.
* **org.hibernate.Query**: An object-oriented representation of a Hibernate query.
* **Controller**: This package contains all the control operations of the application.
* **DAO**: This package consists of classes which access database.
* **VO**: This package consists of Transfer object for setting and getting the required element.

**4.2 Main user defined functions**

|  |  |
| --- | --- |
| **Name of the function** | **PreBookingController** |
| Description | The method is used to add/manage parking stations |
| Syntax | Public PreBookingController (carno,userid,username,email\_id) |
| Parameters | carno,userid,username,email\_id |
| Return value | Slot |
| Calling function | It gets called when object gets created for PreBookingController class |
| Called function | getVehicle() |

**Table 4.1** User-defined PreBookingController

|  |  |
| --- | --- |
| **Name of the function** | **BookingController** |
| Description | The method is used to book a parking slot |
| Syntax | Public BookingController (carno,userid,username,email\_id) |
| Parameters | carno,userid,username,email\_id |
| Return value | Slot |
| Calling function | It gets called when object gets created for BookingController class |
| Called function | preBook(), getVehicle() |

**Table 4.2** User-defined BookingController

|  |  |
| --- | --- |
| **Name of the function** | **User** |
| Description | The method is used to add a new user or manage existing user |
| Syntax | Public user(username,email\_id,ph\_no,carno) |
| Parameters | username,email\_id,ph\_no,carno |
| Return value | Userid |
| Calling function | It gets called when object gets created for User class |
| Called function | update(),insert(),allotParking() |

**Table 4.3** User-defined User function

|  |  |
| --- | --- |
| **Name of the function** | **Feedback** |
| Description | The method is used by employee or user to give the feedback about the system he is using |
| Syntax | public Feedback(user\_id,employee\_id) |
| Parameters | user\_id,employee\_id |
| Return value | - |
| Calling function | It gets called when object gets created for Feedback class |
| Called function | update(),insert(),delete(),edit(),show() |

**Table 4.4** User-defined Feedback function

**4.3 Sample Code for One User Defined Function**

package controller;

importjava.io.IOException;

importjava.text.ParseException;

importjava.text.SimpleDateFormat;

importjava.util.Date;

importjava.util.List;

importjavax.servlet.ServletException;

importjavax.servlet.annotation.WebServlet;

importjavax.servlet.http.HttpServlet;

importjavax.servlet.http.HttpServletRequest;

importjavax.servlet.http.HttpServletResponse;

importjavax.servlet.http.HttpSession;

importvo.FeedBackVO;

importvo.UserVO;

importdao.FeedBackDAO;

/\*\*

\* Servlet implementation class Circular

\*/

@WebServlet("/FeedBack")

@SuppressWarnings("rawtypes")

public class FeedBack extends HttpServlet {

private static final long serialVersionUID = 1L;

/\*\*

\* @see HttpServlet#HttpServlet()

\*/

publicFeedBack() {

super();

// TODO Auto-generated constructor stub

}

/\*\*

\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

// mode=0 Add New

//mode=1 Insert

//mode=2 Edit

//mode=3 update

//mode=4 Delete

//mode=5 Show

//Mode Type defined to String to control over illegal parameter passing through URL

String mode="0";

//Getting the mode variable, if not than default to 0(zero)

if(request.getParameter("mode")!=null)

mode = request.getParameter("mode");

else

mode="0";

if(mode.equals("2")){

edit(request, response);

}

else if(mode.equals("4")) {

delete(request, response);

}else if(mode.equals("manage")){

show(request, response);

}else{

//default Add

response.sendRedirect(request.getContextPath()+"/views/employee/FeedBack/giveFeedBack.jsp");

}

}

/\*\*

\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

// mode=0 Add New

//mode=1 Insert

//mode=2 Edit

//mode=3 update

//mode=4 Delete

//mode=5 Show

//Mode Type defined to String to control over illegal parameter passing through URL

String mode="0";

//Getting the mode variable, if not than default to 0(zero)

if(request.getParameter("mode")!=null)

mode = request.getParameter("mode");

else

doGet(request, response);

if(mode.equals("1")){

insert(request, response);

}else if(mode.equals("3")){

update(request, response);

}else{

//default Add

response.sendRedirect(request.getContextPath()+"/views/employee/FeedBack/giveFeedBack.jsp");

}//default

}

protected void insert(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

/\*This code fetches the user Id of logged in User

Begin\*/

HttpSession session = request.getSession();

longuserID;

if (session.getAttribute("userID") != null) {

userID = (long) session.getAttribute("userID");

} else {

userID=1;

}

/\*ENDS fetching user id\*/

String name = request.getParameter("Name");

String email = request.getParameter("email");

String subject = request.getParameter("subject");

String description= request.getParameter("description");

UserVOuserVO = new UserVO();

userVO.setUserID(userID);

FeedBackVOVO\_obj =new FeedBackVO();

VO\_obj.setUserID(userVO);

VO\_obj.setName(name);

VO\_obj.setEmail(email);

VO\_obj.setSubject(subject);

VO\_obj.setDescription(description);

Date currentDate=null;

try {

currentDate = new SimpleDateFormat("MM-dd-yyyyhh:mm:ss").parse(new SimpleDateFormat("MM-dd-yyyyhh:mm:ss").format(new Date()));

} catch (ParseException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

VO\_obj.setCreatedDate(currentDate);

VO\_obj.setModifiedDate(currentDate);

FeedBackDAODAO\_obj=new FeedBackDAO();

DAO\_obj.insert(VO\_obj);

response.sendRedirect(request.getContextPath()+"/views/employee/FeedBack/success.jsp");

}

protected void update(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

//not to be updated

}

protected void edit(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// not to be edited

}

protected void delete(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

intfeedBackID=Integer.parseInt(request.getParameter("id"));

FeedBackVOVO\_obj =new FeedBackVO();

VO\_obj.setFeedBackID(feedBackID);

FeedBackDAODAO\_obj=new FeedBackDAO();

DAO\_obj.delete(VO\_obj);

show(request, response);

}

protected void show(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

FeedBackDAODAO\_obj=new FeedBackDAO();

List obj= DAO\_obj.showAll();

HttpSession session = request.getSession();

session.setAttribute("obj\_all", obj); response.sendRedirect(request.getContextPath()+"/views/admin/FeedBack/viewFeedBack.jsp");

}

}

**CHAPTER 5**

**TESTING**

The aim of testing process is to identify all defects existing in software product, whether it is Desktop Application or Web Application. If the Program fails by any means or gives unexpected results, then the failure occurred are noted for later debugging and corrections.

**Testing Plans**

The objective of the system testing is to ensure that all individual programs are working as expected, that the programs link together to meet the requirements specified and ensure that the computer system and the associated clerical and other procedures work together. Systems are neither designed as entire systems nor tested as single system. The analyst must perform both unit and system testing.

Different types of testing methods are available. We have tested our system for different aspects like:

**“Does the application meet the goals for which it has been designed?”**

This was very important question that stood before us as the applications was developed and was going to implement on large network.

To fulfill its goal of being able to run on different system, we went through a series of test at different system configuration, with different web browsers, which are widely used. As we need to make our system efficient enough, we need to test it thoroughly.

Finally we tested the system with the real-time data, for which it is actually designed. We are almost successful in satisfying our customers as it was designed according to their requirements. But it is very necessary to maintain this application and so our work is still not over.

**Testing Strategies**

The testing strategy is a formal description of how a software product will be tested. A testing strategy is developed for all levels of testing as and when required. The testing team analyses the requirements, writes the test strategy and reviews the plan with the project team. The test plan may include test cases, conditions and the test environment, a list of related tasks, pass/fail criteria and risk assessment.

As we were working in a team, it was a bit difficult to check the work done, as no individual was assigned particular task or work. Moreover, checking the work done was very important to reduce risk factor. Right from the starting we planned to adopt one technique called Assessment by rotation. Thus, other assessed the work done by one member for some time and again revolved for other level check, which reduces the errors.

**Testing Methods**

Testing methods are mainly divided into two parts:

* + 1. Black Box Testing
    2. White Box Testing

**Black Box Testing**

It is the testing without the knowledge of the internal working of the item being tested. For this testing test groups are often used. Due to the nature of this testing the test planning can begin as soon as the specifications are written. This testing has some advantages like it is more effective on larger units of code than glass box testing, tester needs no knowledge of implementation, including specific programming languages, tester and programmer are independent of each other, tests are done from a user’s point of view, will help to expose any ambiguities or inconsistencies in the specifications, test cases can be designed as soon as the specifications are complete.

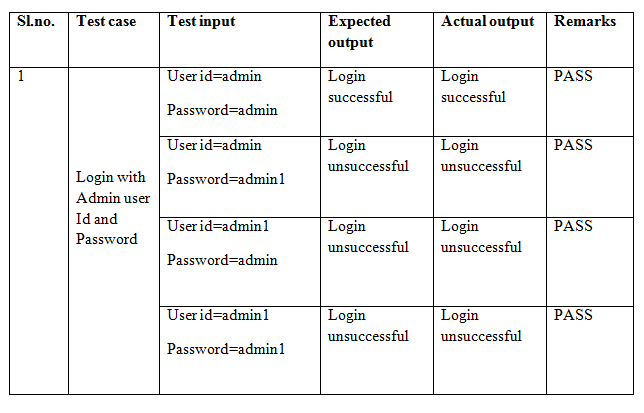
It focuses on the functional requirements of the software. It enables the software engineer to derive sets of input conditions that will fully exercise all functional requirements of the project. Black box testing attempts to find errors in the following categories: incorrect or missing links, interface errors, errors in data structure, behavior or performance errors and the initialization and termination errors.

**White Box Testing**

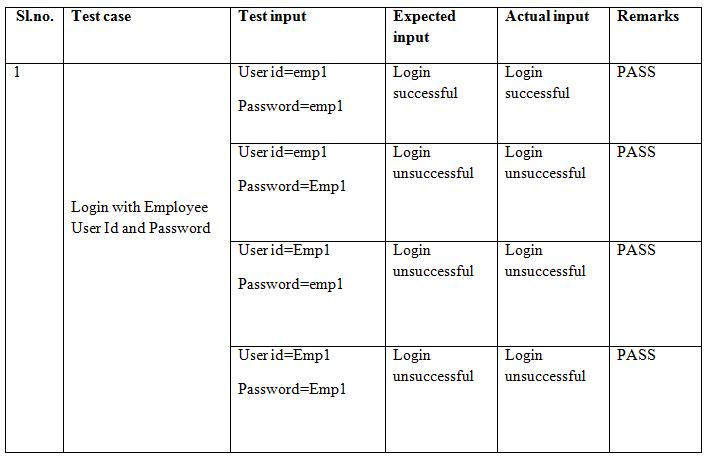
It is the testing strategy that deals with the internal logic and structure of the code. White box testing is also called as the glass testing, structural testing, open testing or the clear box testing. The advantages of the white box testing are that as the knowledge of internal coding structure is needed, it becomes easy to find out which type of input data can help in testing the application effectively. It helps in optimizing the code helping in removing the extra code, which may cause hidden defects in the system. It is test case design method that uses the control structure of the procedural design to derive the test cases. Using white box testing techniques the software engineer can derive test cases that: guarantee that all independent paths within a module have been exercised at least once, exercise all global decisions on their true and false sides, execute all loops at their boundaries and within operation bounds and also exercise the internal data structure to ensure their validity.

**5.1 Unit testing**

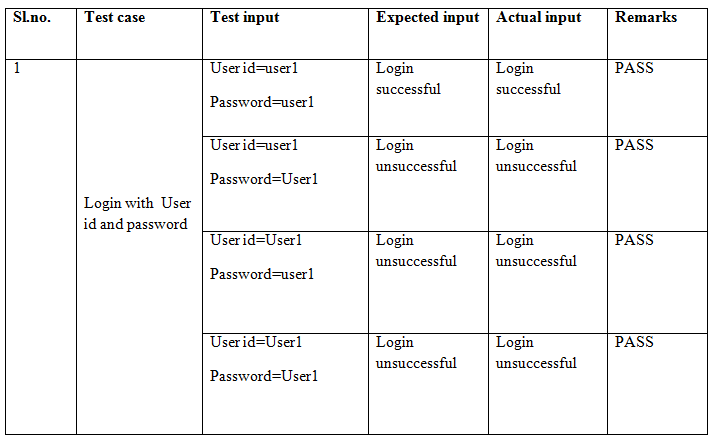
Unit Testing focuses on verification effort on the smallest unit of project that is the module. Unit testing exercises specific paths in a module’s control structure to ensure complete coverage and maximum error detection. This test focuses on each module individually, ensuring that it functions properly as a unit. Hence, the naming is Unit Testing.

****

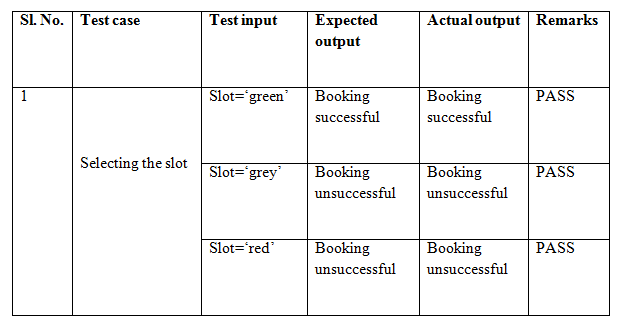
**Table 5.1** Test cases for login module for admin

****

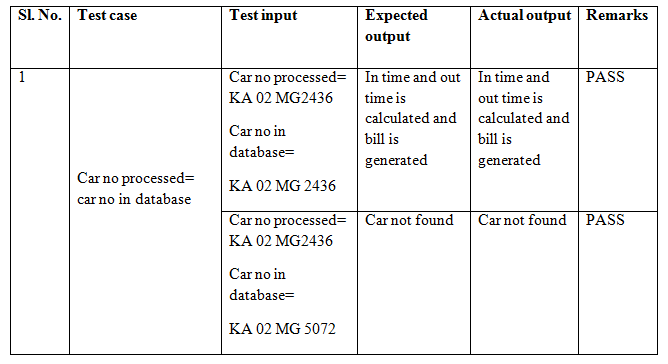
**Table 5.2** Test cases for login module of employee

****

**Table 5.3** Test cases for login module of user

****

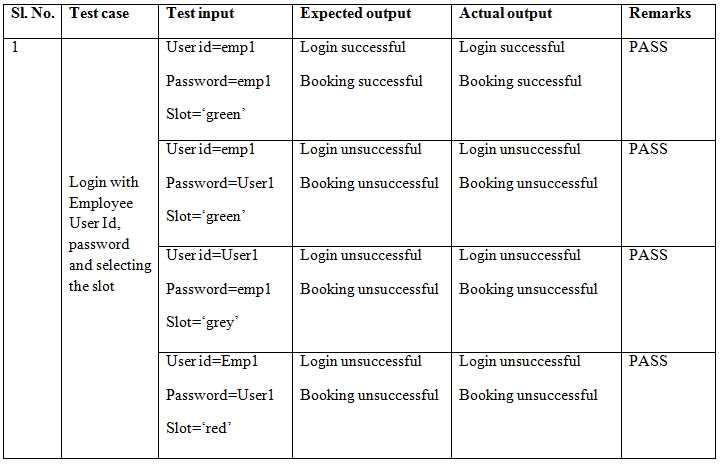
**Table 5.4** Test cases for pre-booking module

****

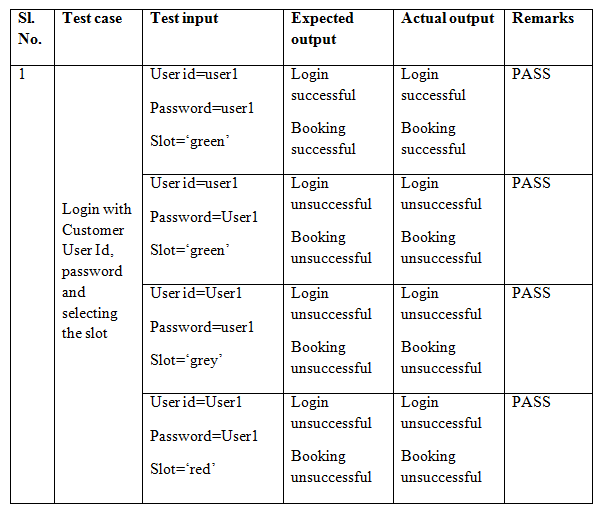
**Table 5.5** Test cases for logout module for user

**5.2 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 orders 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.

****

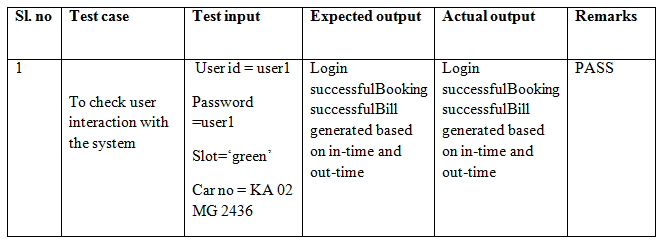
**Table 5.6** Test cases for integration of login module for employee and prebooking module

****

**Table 5.7** Test cases for integration of login module for user and prebooking module

**5.3 System testing**

After the integration of all modules we deploy whole application. One of the member of the development team will do the testing of the whole project and create the excel sheet of the bugs. After completion of the solving of these errors the project would be ready to use.

****

**Table 5.8** Test cases for system testing

**CHAPTER 6**

**CONCLUSION AND FUTURE ENHANCEMENTS**

**6.1 CONCLUSION**

Compared to other developed countries, the problem of parking is disheartening in India as there is no well devised plan in place. There is a wide gap and total mismatch between the production of vehicles and the parking slots.The parking problem is quite acute in places of entertainment such as theatres and shopping malls.A well thought parking plan saves the time of visitors in booking a parking slot in advance and the administration to allocate the vacant slot in a methodical and organized manner.

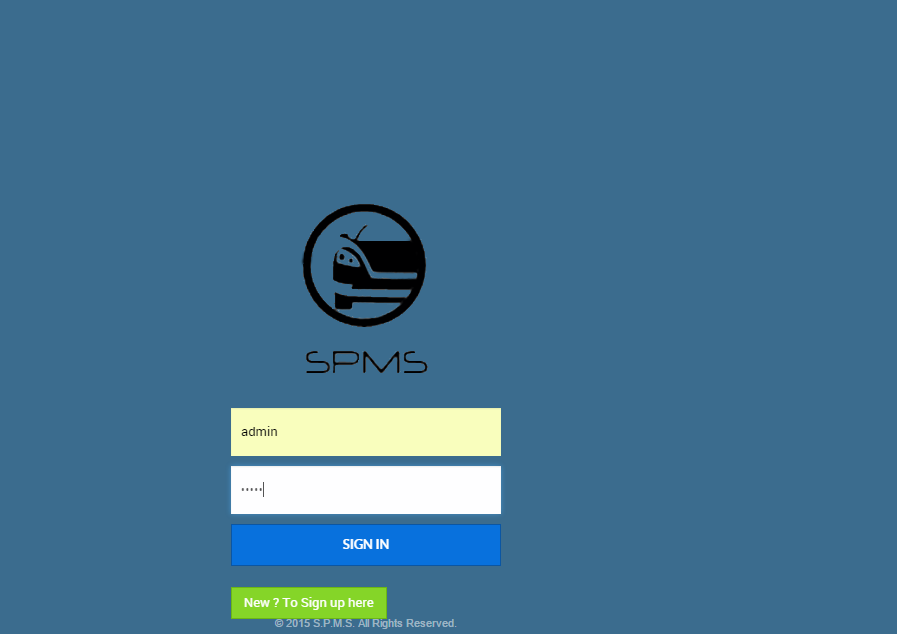
This new approach of Smart Parking System helps people to find the free parking space the large parking areas in huge metropolitan cities. It helps the visitors in finding out the availability of a parking slot, get the availability confirmed, and reach the place within the time slot allotted. It reduces time and efforts required for the entire process of car parking as compared to the present scenario. This system also helps the users to retrieve the path to the slot where they had parked their cars as it is difficult to find the car in a large parking lot.

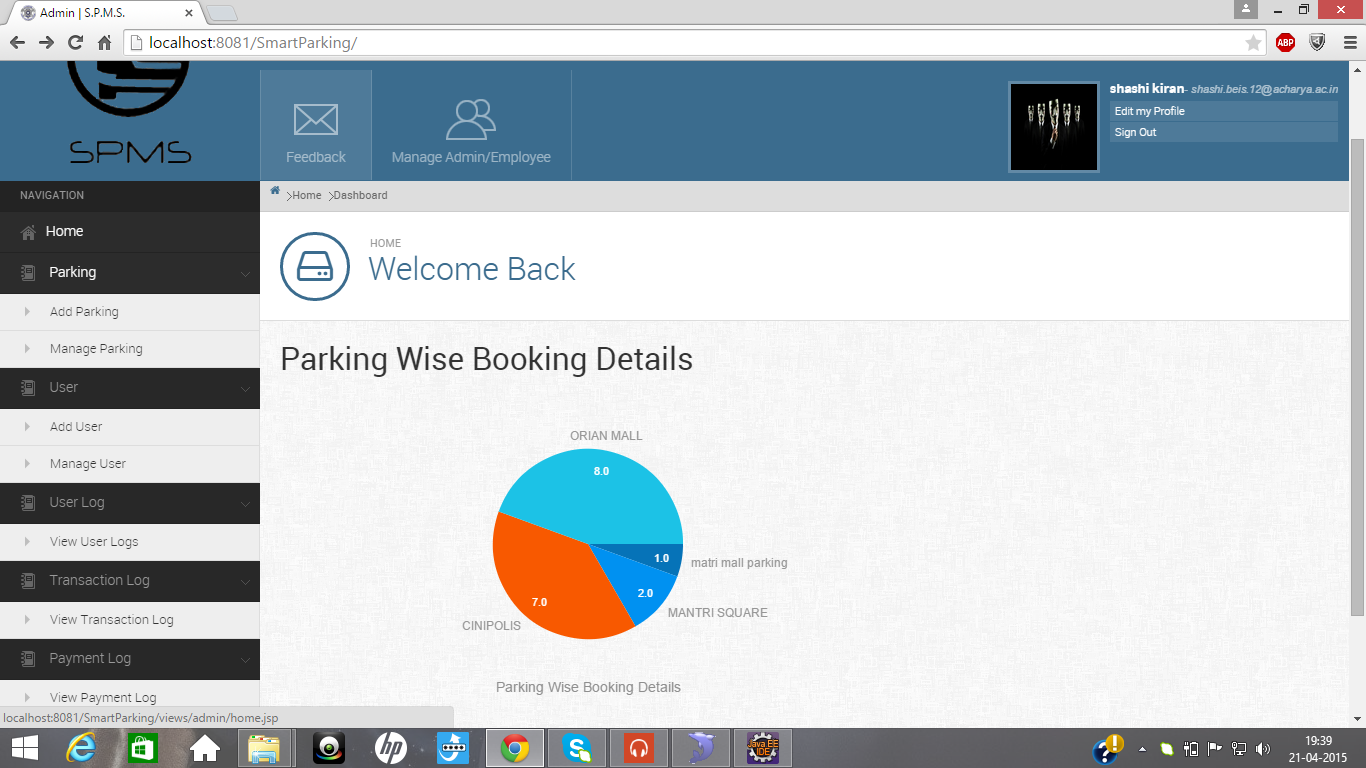
The parking operator can use this system data to prevent vehicle thefts and the staffing requirement for traffic is reduced. The system significantly reduces traffic and the resulting vehicle emissions by decreasing the time required for customers to locate open spaces.The proposed system also reduces the usage of paper to a larger extent and hence it is proved to be ‘eco-friendly’. We have a partially automated as Image processing concepts have been used in order to store the car number. User can pre-book a parking slot which is a special feature of the system. We have developed a web application which acts as the interface for the users.

**6.2 FUTURE ENCHANCEMENT**

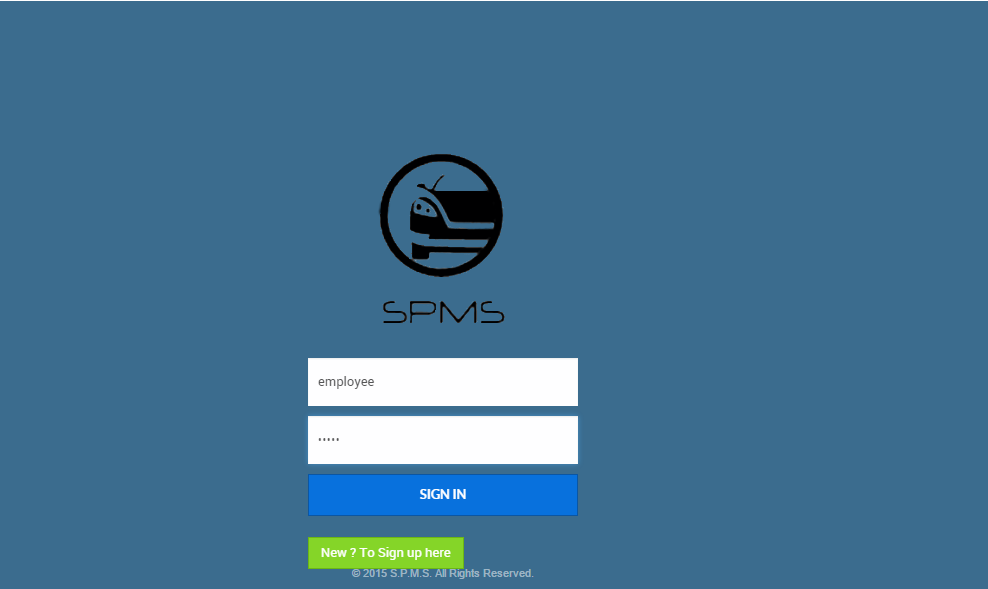
* In our system, the font on the number plate is standardized to one particular font style. But Image Processing techniques can be used for different fonts which a number plate of a car can contain.
* We can provide a kiosk in the parking area so that the user can easily obtain a graphical path by entering his/her car number.
* We may train the system to detect different coloured number plates.
* The car sizes may differ because of which, positioning of the number plate may vary. Hence, the system can be enhanced to use sophisticated morphology techniques in order to capture the number plate image within a frame and then extract data from it.
* The system’s scope can be extended for two-wheeler parking also.
* We can implement the same idea using a barcode or QR code as well. We can also use RC cards provided extensively for each vehicle as a magnetic cards which stores all the data about the car.

**SNAPSHOTS**

****

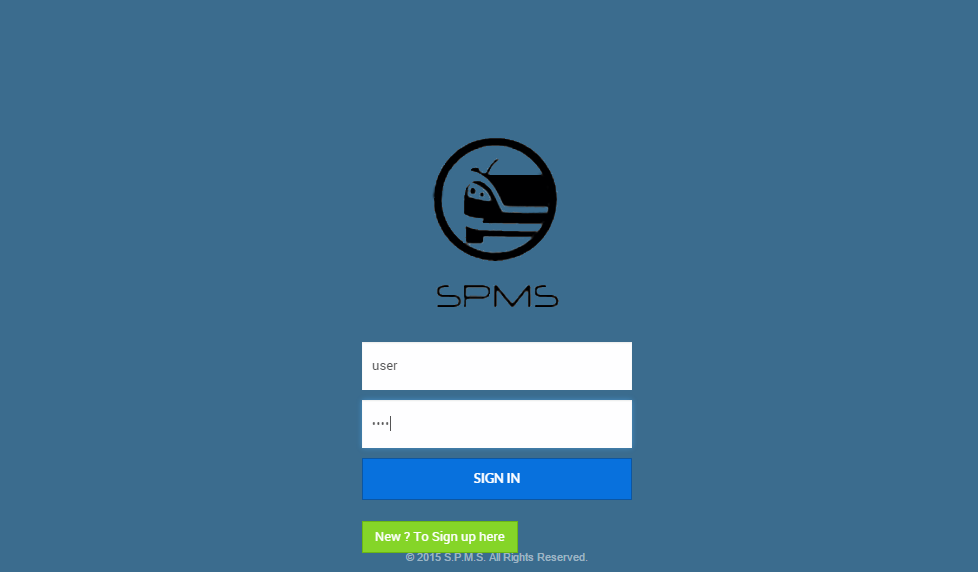


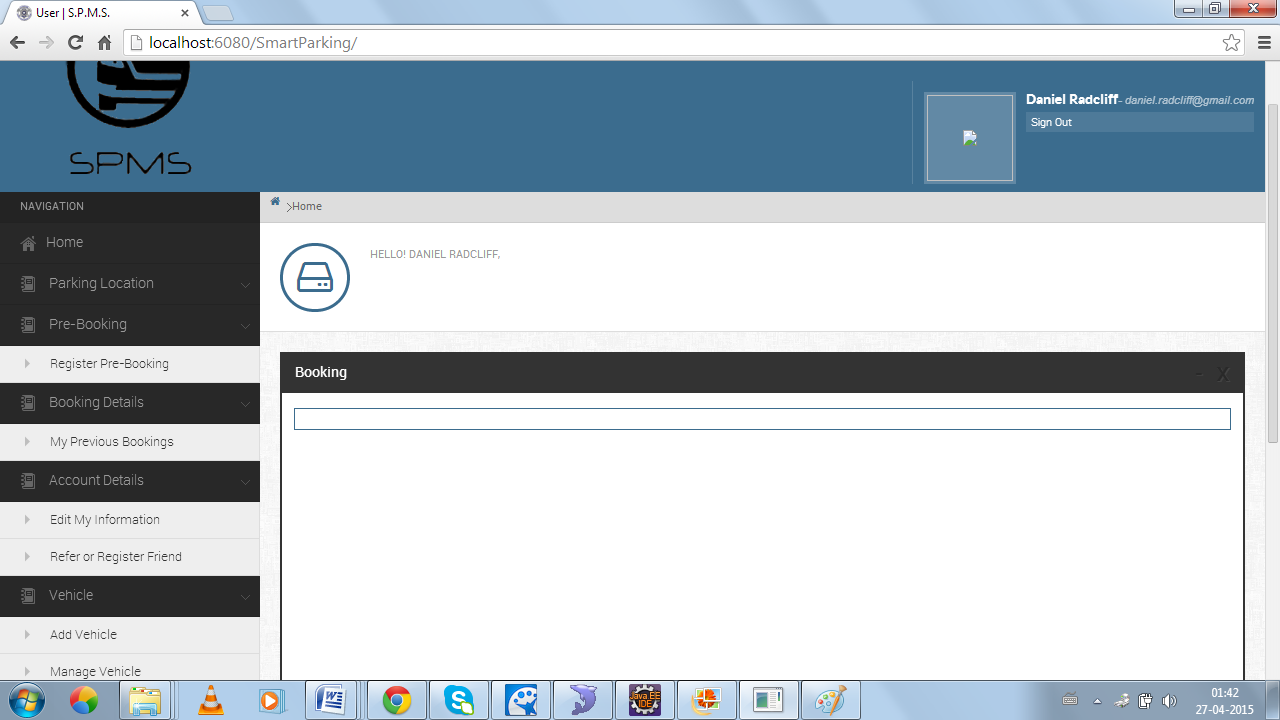
Admin Login page and home page with all the functions that the admin can perform.



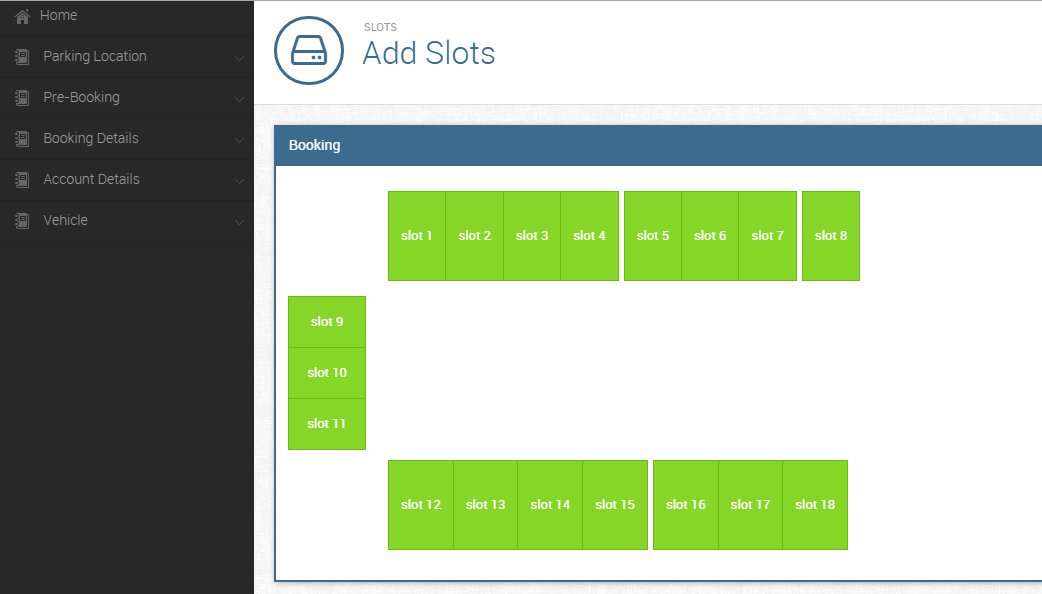


Employee Login page and home page with all the functions that the employee can perform.

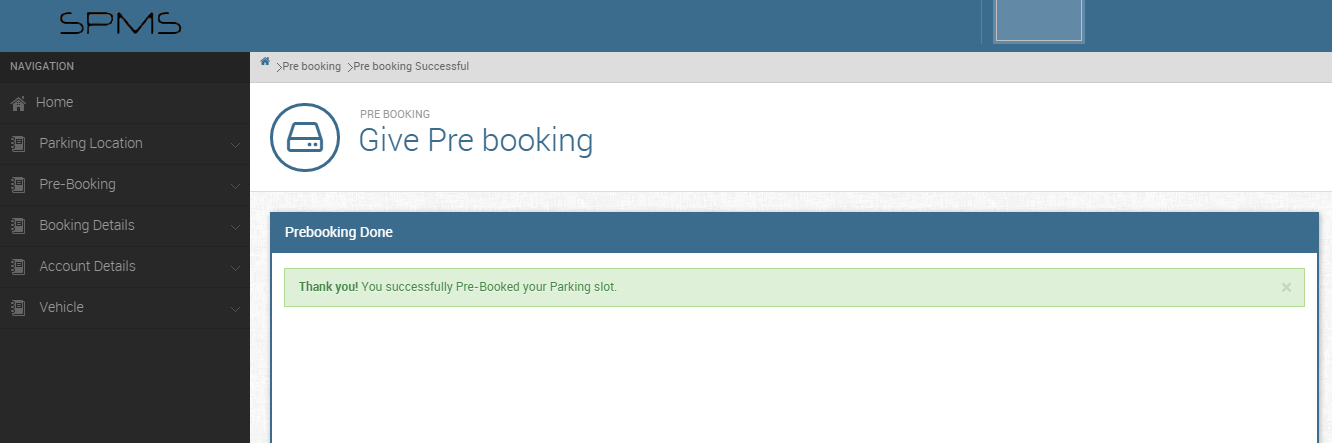




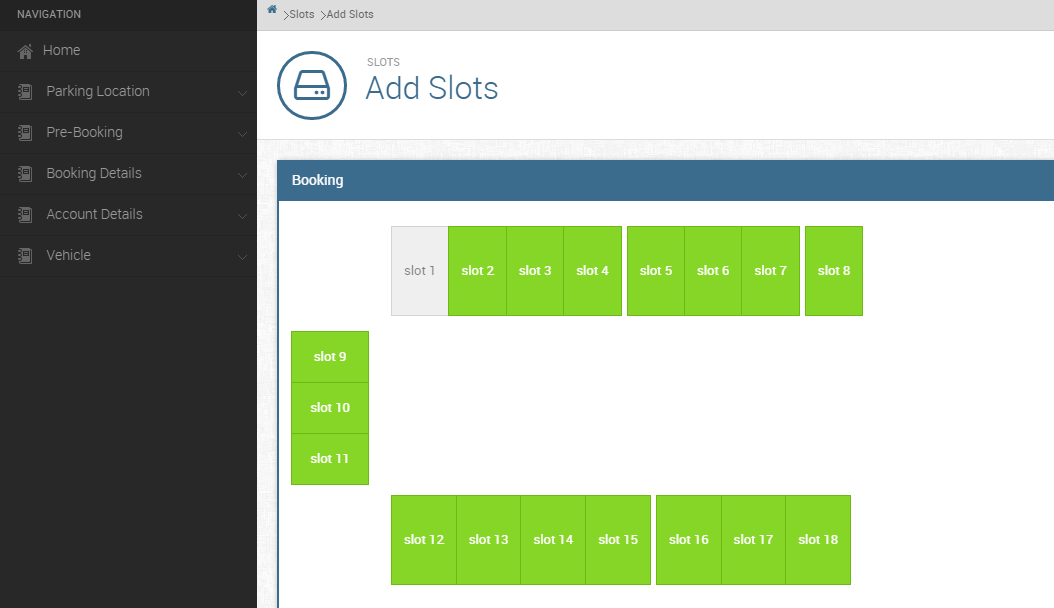
User Login page and home page with all the functions that the user can perform.



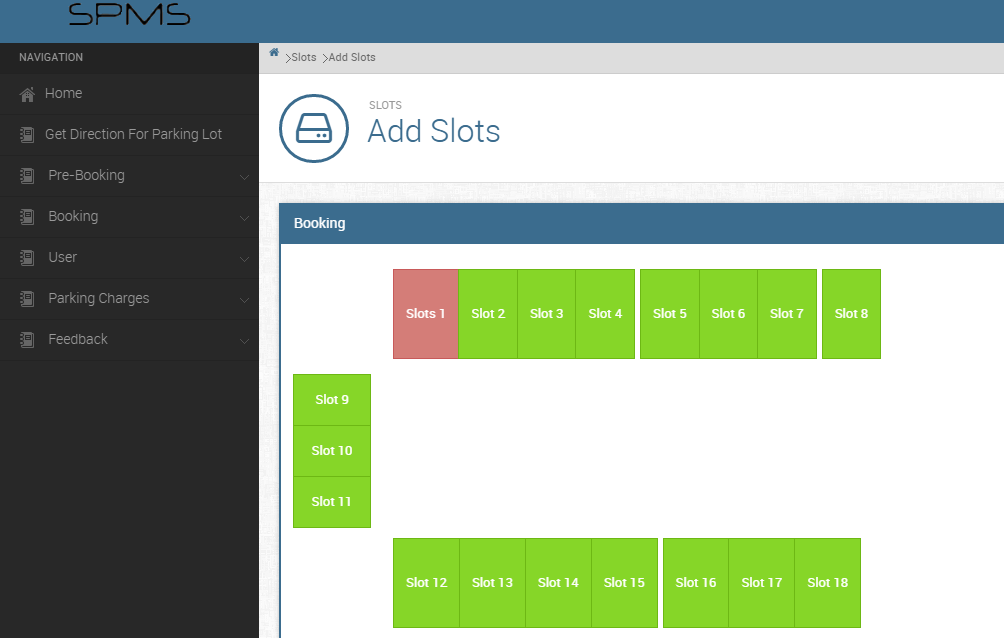
Representation of available slots indicated in green before pre-booking.



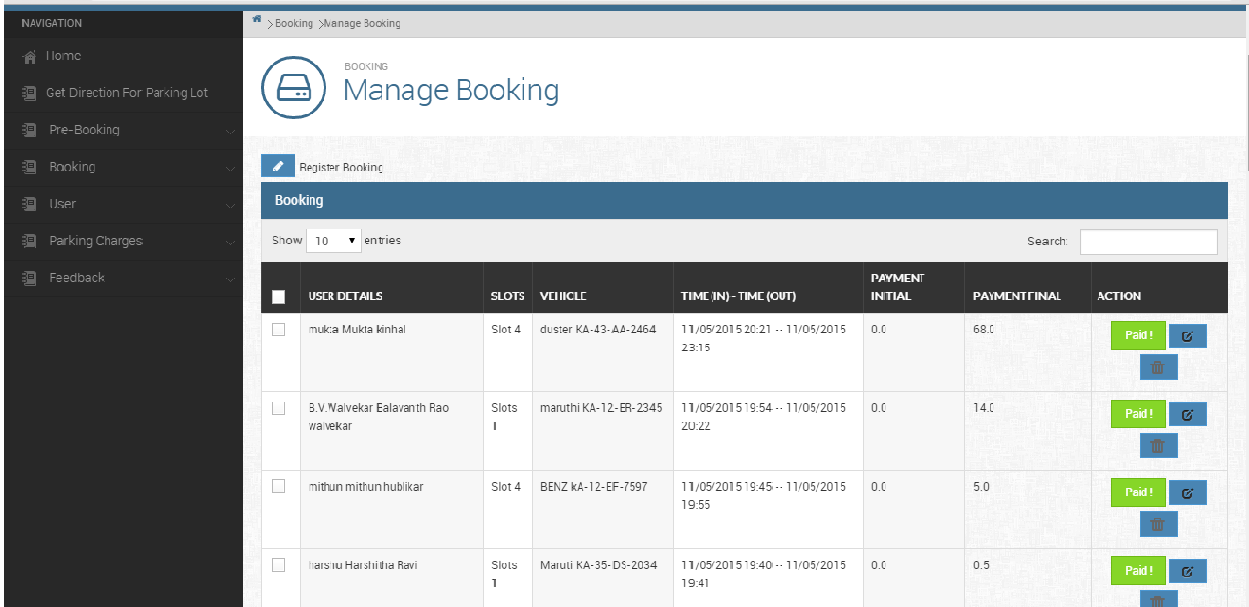
Successful pre-booking message shown to user.



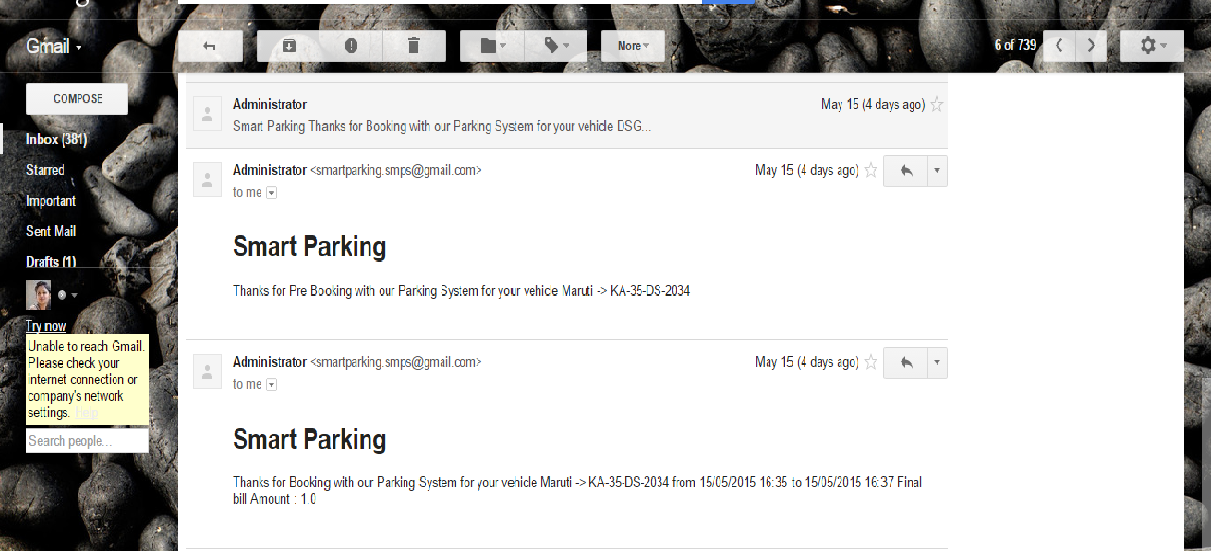
Representation of pre-booked slot indicated in grey after pre-booking.

****

Representation of occupied slot indicated in red after check-in of the vehicle.

****

Payment details of the users displayed on employee’s manage booking page.



Acknowledgement Mail received by the user after pre-booking and successful payment.

**ANNEXURE A**

**GLOSSARY**

**Image processing**

It is any form of signal processing for which the input is an image, such as a photograph or video.

**Mathematical morphology**

Car number plates’ detection is performed by means of mathematical morphology. It makes use of an intelligent filtering of the input image based on a set of filters removing unnecessary image elements, but preserving the position and shape of characters of the car number plate.

**Hibernate**

It is an object-relational mapping framework for java language, providing a framework for mapping an object-oriented domain model to a traditional relational database. Hibernate solves object-relational impedance mismatch problems by replacing direct persistent related database accesses with high level object handling functions.

**LabView**

It is a system design platform and development environment for a visual programming language from National Instruments.

**ZigBee**

It is a specification for a suite of high level communication protocols used to create personal area networks built for small, low power digital radios.

**Google Wallet**

It is a mobile payment system developed by Google that allows its users to store debit cards, credit cards, loyalty cards and gift cards among other things, as well as redeeming sales promotions on their mobile phone.

**ANNEXURE B**

**ACRONYMS**

**LPR** License Plate Recognition

**LabView** Laboratory Virtual Instrument Engineering Workbench

**VNPR** Vehicular Number Plate Recognition

**GSM** Global System for Mobile communication

**micro-RTU** micro-Remote Terminal Unit

**PIC** Peripheral Interface Controller

**SMS** Short Message Services

**ORM** Object-Relational Mapping

**SPSR** Smart Parking System based on Reservation

**ANNEXURE C**

**Java**

**Overview of Java**

Java is loosely based on C++ syntax, and is meant to be Object-Oriented Structure of java is midway between an interpreted and a compiled language. The java compiler into Byte Codes, which are secure and portable across different platforms, compiles Java programs. These byte codes are essentially instructions encapsulated in single type, to what is known as java virtual machine (JVM), which resides in standard browser.

JVM is available for almost all OS. JVM converts these byte codes into machine specific instructions at runtime. Java is actually a platform consisting of three components:

1. Java programming language.
2. Java library of classes and interfaces.
3. Java Virtual Machine

**Features of Java**

* Java is a simple language. It does not make use of pointers, function overloading etc...
* Java is object-oriented language and supports encapsulation, inheritance, Polymorphism and dynamic binding, but does not support multiple inheritances.
* Everything in java is an object except some primitive data types.
* Java is portable.
* It is an architecture neutral that is java programs once compiled can be executed on any machine that is enabled.
* Java is distributed in its approach and used for Internet programming.
* Java is robust, secured, high performing and dynamic in nature.
* Java supports multithreading. Therefore different parts of the program can be executed at the same time.

**Java Database Connectivity (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.

**History and Implementation**

* Sun Microsystems released JDBC as part of [JDK](http://en.wikipedia.org/wiki/Java_Development_Kit) 1.1 on February 19, 1997. It has since formed part of the [Java Standard Edition](http://en.wikipedia.org/wiki/Java_Platform,_Standard_Edition).
* The JDBC classes are contained in the [Java package](http://en.wikipedia.org/wiki/Java_package) [java.sql](http://docs.oracle.com/javase/7/docs/api/java/sql/package-summary.html) and [javax.sql](http://docs.oracle.com/javase/7/docs/api/javax/sql/package-summary.html).
* Starting with version 3.1, JDBC has been developed under the [Java Community Process](http://en.wikipedia.org/wiki/Java_Community_Process). JSR 54 specifies JDBC 3.0 (included in J2SE 1.4), JSR 114 specifies the JDBC Rowset additions, and JSR 221 is the specification of JDBC 4.0 (included in Java SE 6).
* The latest version, JDBC 4.1, is specified by a [maintenance release](http://en.wikipedia.org/wiki/Maintenance_release) of JSR 221 and is included in Java SE 7.

**Functionality**

JDBC allows multiple implementations to exist and be used by the same application.

The API provides a mechanism for dynamically loading the correct Java packages and registering them with the JDBC Driver Manager. The Driver Manager is used as a connection factory for creating JDBC connections.

JDBC connections support creating and executing statements. These may be update statements such as SQL's CREATE, INSERT, UPDATE and DELETE, or they may be query statements such as SELECT. Additionally, stored procedures may be invoked through a JDBC connection. JDBC represents statements using one of the following classes:

[Statement](http://docs.oracle.com/javase/7/docs/api/java/sql/Statement.html) – the statement is sent to the database server each and every time.

[PreparedStatement](http://docs.oracle.com/javase/7/docs/api/java/sql/PreparedStatement.html) – the statement is cached and then the [execution path](http://en.wikipedia.org/wiki/Query_plan) is pre-determined on the database server allowing it to be executed multiple times in an efficient manner.

[CallableStatement](http://docs.oracle.com/javase/7/docs/api/java/sql/CallableStatement.html) – used for executing [stored procedures](http://en.wikipedia.org/wiki/Stored_procedures) on the database.

Update statements such as INSERT, UPDATE and DELETE return an update count that indicates how many [rows](http://en.wikipedia.org/wiki/Row_(database)) were affected in the database. These statements do not return any other information.

Query statements return a JDBC row result set. The row result set is used to walk over

the [result set](http://en.wikipedia.org/wiki/Result_set). Individual [columns](http://en.wikipedia.org/wiki/Column_(database)) in a row are retrieved either by name or by column number. There may be any number of rows in the result set. The row result set has metadata that describes the names of the columns and their types.

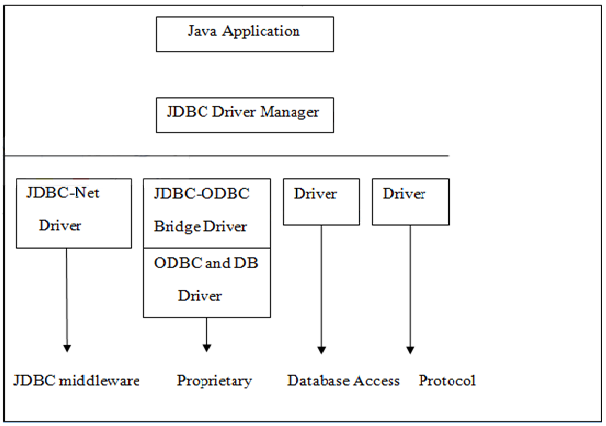
There is an extension to the basic JDBC API in the [javax.sql](http://docs.oracle.com/javase/7/docs/api/javax/sql/package-summary.html).

JDBC connections are often managed via a [connection pool](http://en.wikipedia.org/wiki/Connection_pool) rather than obtained directly from the driver. Examples of connection pools include [BoneCP](http://jolbox.com/), [C3P0](http://sourceforge.net/projects/c3p0) and [DBCP](http://commons.apache.org/dbcp).

**JDBC drivers**

There are four types of JDBC drivers as shown. They are:

1. JDBC-ODBC bridge plus ODBC driver
2. JDBC-Net all-Java driver
3. Native-API partly-Java driver
4. Native-protocol all-Java driver

****

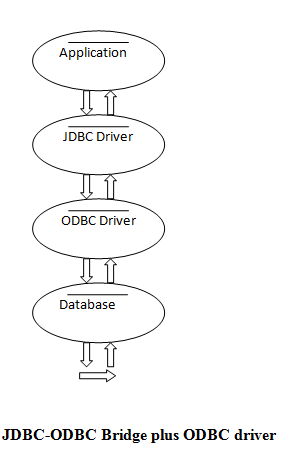
**JDBC Drivers**

Each of the JDBC drivers is explained in detail below.

**JDBC-ODBC Bridge plus ODBC driver**

The Java Soft bridge product provides JDBC access via ODBC drives. The ODBC binary code and in many cases database client code must be loaded on each client machine that uses this driver.

As a result, this kind of driver is most appropriate on a corporate network where client installations are not a major problem, or for application server code written in Java in three-tier architecture.



**JDBC-ODBC Bridge plus ODBC driver**

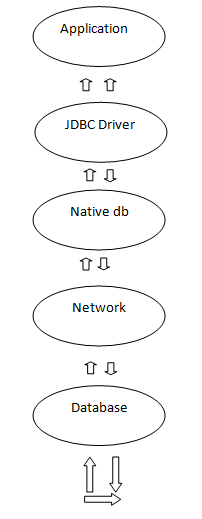
**JDBC-Net all-Java driver**

This driver translates JDBC calls into a DB MS-independent net protocol, which is then translated, to a DBMS protocol by a Server. This net Server middle ware is able to connect its all-Java clients to many different databases. The specific protocol used depends on the vendor. In general this is most flexible JDBC alternative. It is likely that all vendors of this solution will provide products to also support Internet access through firewalls, etc, that the web imposes. Several vendors are adding JDBC drivers to their existing database middleware products.

**Native-API partly-Java Driver**

This kind of driver converts JDBC calls into calls on the client API for Oracle, Sybase, Informix, DB2, or other DBMS. Note that, like the Bridge driver, this style of driver requires that some binary code be loaded on each client machine.

Fig 4.3 shows Native-API partly JAVA Driver, where the application program requires a driver to connect to the database. Usually we use sun.jdbc.odbc.jdbcodbc driver this driver should request driver manager using driver manager.getconnection.



**Native API partly Java driver**

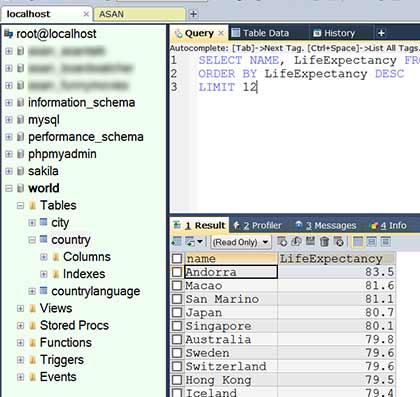
**Native-protocol all-Java driver**

This kind of driver converts JDBC calls into the network protocol used by DBMS's directly. This allows a direct call from the client machine to the DBMS server and is practical solution for Internet access. Since many of these protocols are proprietary, database vendors themselves will be the primary source. Several database vendors have these in progress.

**SqlYog**

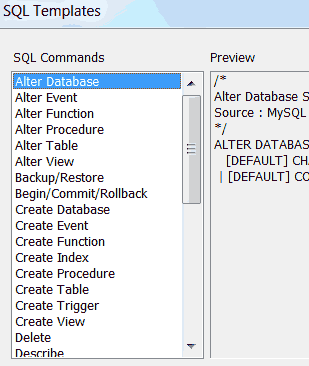
MySQL is a client-server application. The MySQL server will usually be running on a web server where your php programs reside. You can connect a MySQL client to any MySQL server. Figure 3.4 shows connecting to my local MySQL server. SQLYOG is a brilliant interface to MySQL. It lets us work with databases from multiple servers simultaneously in different tabs. It also lets us copy databases from one server to another. Also data can be edited in a spreadsheet-style grid.

The first thing to notice in SQLyog is that it has a tabbed interface. Servers appear in tabs. One the picture below, the top left, you can see a tab for “localhost”, which s a development server, and a second tab for “ASAN”, which stands for a website name that is being used.



**SQLYog interface**

There is a feature of auto-complete of queries in SQLyog which helps the user to write queries faster and the SQLyog auto-corrects the queries. It also has a unique feature which lets the user to edit the information in the tables without writing queries. The user can just edit the data and save it just like how it is in a spreadsheet. There is also provision for SQL templates which makes it easier for the user to write queries. The below table shows how to select the SQL commands which we require.



**SQL Templates**

We can quickly filter or sort the table contents in SQLyog. When you run a saved job, you have to click through a number of screens before your job will run.

**Advantages of SQLyog**

* Automatically refreshing table data after running a query.
* Get information to optimize your queries in the query profiler.
* Simplified output of the query formatter.
* Simplified interface for the users.
* Easy methods of troubleshooting.
* Usage of many SQL shortcuts. Users can define and set their own shortcuts.
* Users can connect with multiple servers at a time and work simultaneously.
* Special features like schema designer and query builder are found in SQLyog.

**Hibernate**

Hibernate ORM (Hibernate in short) is an [object-relational mapping](http://en.wikipedia.org/wiki/Object-relational_mapping) [framework](http://en.wikipedia.org/wiki/Software_framework) for the [Java](http://en.wikipedia.org/wiki/Java_(programming_language)) language, providing a [framework](http://en.wikipedia.org/wiki/Software_framework) for mapping an [object-oriented](http://en.wikipedia.org/wiki/Object-oriented_programming) [domain model](http://en.wikipedia.org/wiki/Domain_model) to a traditional [relational database](http://en.wikipedia.org/wiki/Relational_database). Hibernate solves [object-relational impedance mismatch](http://en.wikipedia.org/wiki/Object-relational_impedance_mismatch) problems by replacing direct [persistence](http://en.wikipedia.org/wiki/Persistence_(computer_science))-related database accesses with high-level object handling functions. Hibernate is [free software](http://en.wikipedia.org/wiki/Free_software) that is distributed under the [GNU Lesser General Public License](http://en.wikipedia.org/wiki/GNU_Lesser_General_Public_License).

Hibernates primary feature is mapping from Java classes to [database tables](http://en.wikipedia.org/wiki/Table_(database)) (and from Java data types to [SQL](http://en.wikipedia.org/wiki/SQL) data types). Hibernate also provides data query and retrieval facilities. It generates SQL calls and relieves the developer from manual result set handling and object conversion. Applications using Hibernate are portable to supported SQL databases with little performance overhead

**Mapping**

Mapping Java classes to database tables is accomplished through the configuration of an [XML](http://en.wikipedia.org/wiki/XML) file or by using [Java Annotations](http://en.wikipedia.org/wiki/Java_annotation). When using an XML file, Hibernate can generate skeleton [source code](http://en.wikipedia.org/wiki/Source_code) for the persistence classes. This is unnecessary when annotations are used. Hibernate can use the XML file or the annotations to maintain the [database schema](http://en.wikipedia.org/wiki/Database_schema).

Facilities to arrange [one-to-many](http://en.wikipedia.org/wiki/Point-to-multipoint_communication) and [many-to-many](http://en.wikipedia.org/wiki/Many-to-many_(data_model)) relationships between classes are provided. In addition to managing associations between objects, Hibernate can also manage [reflexive](http://en.wikipedia.org/wiki/Reflexive_relation) associations where an object has a one-to-many relationship with other instances of its own [type](http://en.wikipedia.org/wiki/Data_type).

Hibernate supports the mapping of custom value types. This makes the following scenarios possible:

* Overriding the default SQL type that Hibernate chooses when mapping a column to a property.
* Mapping Java [Enum](http://en.wikipedia.org/wiki/Enumerated_type" \o "Enumerated type) to columns as if they were regular properties.
* Mapping a single property to multiple columns.

**Integration**

Hibernate can be used both in standalone [Java](http://en.wikipedia.org/wiki/Java_(programming_language)) applications and in [Java EE](http://en.wikipedia.org/wiki/Java_EE) applications using [servlets](http://en.wikipedia.org/wiki/Java_Servlet), [EJB](http://en.wikipedia.org/wiki/EJB) session beans, and [JBI](http://en.wikipedia.org/wiki/JBI) service components. It can also be included as a feature in other programming languages. For example, [Adobe](http://en.wikipedia.org/wiki/Adobe_Systems) integrated Hibernate into version 9 of [ColdFusion](http://en.wikipedia.org/wiki/ColdFusion) (which runs on J2EE app servers) with an abstraction layer of new functions and syntax added into [CFML](http://en.wikipedia.org/wiki/CFML).

**Application Programming Interface**

The Hibernate API is provided in the [Java package](http://en.wikipedia.org/wiki/Java_package) [org.hibernate](http://docs.jboss.org/hibernate/stable/core/javadocs/index.html?overview-summary.html).

### [org.hibernate.SessionFactory](http://docs.jboss.org/hibernate/stable/core/javadocs/org/hibernate/SessionFactory.html) interface

### [org.hibernate.Session](http://docs.jboss.org/hibernate/stable/core/javadocs/org/hibernate/Session.html) interface

Represents a Hibernate session, i.e., the main point of the manipulation performed on the database entities. The latter activities include (among the other things) managing the persistence state ([transient](http://en.wikipedia.org/wiki/Transient_(computer_programming)), [persisted](http://en.wikipedia.org/wiki/Java_Persistence_API), detached) of the objects, fetching the persisted ones from the database and the management of the transaction demarcation.

A session is intended to last as long as the logical transaction on the database. Due to the latter feature, Session implementations are not expected to be thread safe nor to be used by multiple clients.

**REFERENCES**

[1] M. Y. I Idris, E. M Tamil, Z. Razak, N. M Noor, L. W Kim, “Smart Parking System using Image Processing Techniques in Wireless Sensor Network Environment”, Information Technology Journal 8 (2): 114-127, 2009, ISSN 1812-5638.

[2] Satish V.Reve, SonalChoudhri, “Management of Car Parking System Using Wireless Sensor Network”, International Journal of Emerging Technology and Advanced Engineering, ISSN 2250-2459, Volume 2, Issue 7, July 2012.

[3] M.O. Reza, M.F. Ismail\*, A.A. Rokoni, M.A.R. Sarkar, “Smart Parking System with Image Processing Facility”, *I.J. Intelligent Systems and Applications,* 2012, 3, 41-47, Published Online April 2012 in MECS

[4] Hilal Al-Kharusi, Ibrahim Al-Bahadly, “Intelligent Parking Management System Based on Image Processing”, World Journal of Engineering and Technology, 2014, 2, 55-67, Published Online May 2014 in SciRes.

[5] Hongwei Wang and Wenbo He, “A Reservation-based Smart Parking System”, First International Workshop on Cyber-Physical Networking Systems.

[6] MohitPatil, Rahul Sakore, “Smart Parking Based on Reservation”, International Journal of Scientific Engineering and Research (IJSER), ISSN (Online): 2347-3878, Volume 2 Issue 6, June 2014.

[7] LalithIyer, Manali Tare, Renu Yadav, HetalAmrutia, “Android Application for Vehicle Parking System: Park Me”, International Journal of Innovations and Advancements in Computer Applications, IJIACS, ISSN 2347 – 8616 Volume 3, Issue 3 May 2014

[8] Cosmo H.Munuo, Dr. Michael Kisangiri, ***“***Vehicle Number Plates Detection and Recognition using improved Algorithms: A Review with Tanzanian Case study”, International Journal Of Engineering And Computer Science ISSN:2319-7242, Volume 3 Issue 5, May 2014, Page No. 5828-5832.

[9] S.Kranthi, K.Pranathi, A.Srisaila, “Automatic Number Plate Recognition”, International Journal of Advancements in Technology, ISSN 0976-4860.

[10] NimaAsadi, Intelligent Embedded Systems Mälardalen University Västerås, Sweden, A Study of Automatic License Plate Recognition Algorithms and Techniques.