

Bachelor's Thesis

Degree Programme: Information Technology

2013

HENOK WAKO HUSSIEN

Online Car Parking Reservation System

– A Desktop Application



TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

BACHELOR'S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Degree programme | Information Technology

Spring 2013 | 55 pages

Instructor: Patric Granholm

Henok Wako Hussien

Online Car Parking Reservation System

In almost every major airport in the world, parking is expensive and limited. Inconsistent flow of information from parking guidance information systems in the busiest airports usually creates a long queue of line during car entry and exit. The main purpose for this thesis is to provide an online car parking reservation service using XAMPP. The implementation and a probabilistic analysis of the application are presented as well.

The drivers receive detailed information online, about the vacant parking lot, the payment method, booking period, expire date, and other alternatives before their arrival to the airport. The graphical user interface and the optimization of the functionality of the application can facilitate the booking process.

KEYWORDS: online car booking, date piker, exit calendar, PHP, XAMPP, MySQL

ACKNOWLEDGMENT

First and foremost, I would like to thank the Lord for the wisdom and perseverance that he has been bestowed upon me during this thesis writing. Secondly, I would like to express my sincere gratitude to my advisor, Mr Patric Granholm, for the continuous support of my thesis study, for his patience, motivation, enthusiasm and immense knowledge. My gratitude goes to Mrs. Poppy Skarli for her immense contribution through the design of the application. My sincere thanks also go to my family for their unconditional support throughout all this work.

CONTENTS

1. INTRODUCTION	1
1.1 STATEMENT OF THE PROBLEM	2
1.2 THE PURPOSE OF THE STUDY	3
1.3 IMPORTANCE OF THE STUDY	3
1.4 LIMITATIONS OF THE STUDY	4
1.5 IMPLICATION OF THE STUDY	5
2. LITERATURE REVIEW	6
2.1 INTRODUCTION	6
2.2 ONLINE CAR PARKING MANAGEMENT PRINCIPLE	6
2.3 CASE STUDY: TYPES OF PARKING LOT	7
2.4 PARKING GUIDANCE INFORMATION SYSTEM (PGI)	9
2.5 PGI SYSTEM EVALUATION	10
2.6 ECONOMICS OF PARKING	10
2.7 PARKING PAYMENT	11
2.8 CAR PARK DESKTOP APPLICATION	12
2.9 MAKING A BOOKING	13
3. METHODOLOGY	14
3.1 RESEARCH AND REVIEW	14
3.2 ANALYSIS DESIGN	14
3.2.1 ANALYSIS TOOLS	14
3.3 ARCHITECTURAL IMPLEMENTATION	15
3.4 VALIDATION	16
4. SCHEME ANALYSIS	17
4.1 CONTEXT FLOW DIGRAM	17
4.2 GRAPHICAL USER INTERFACE (GUI)	18
4.3 LEVELS OF USER AUTHENTICATION	19
4.4 SYSTEM DESIGN	20
4.4.1 THE PROPOSED SYSTEM DESIGN	20
4.5 ARCHITECTURAL DESIGN	21
5. PROGRAM ENVIRONMENT	22
5.1 XAMPP	22
5.2 APACHE TECHNOLOGY	23

5.3	PHP	23
5.4	MYSQL	23
5.5	JAVASCRIPT	24
5.6	STYLE SHEETS	25
5.7	HYPERTEXT MARKUP LANGUAGE	25
5.7.1	HTML5	25
5.8	SCRIPT EDITOR	26
6. CONCLUSION		27
REFERENCES		28

APPENDIX

FIGURES

Figure 1.	PGI indicates the available lots	2
Figure 2.	Improper Parking	3
Figure 3.	Online date picker for entry and exit calendar	5
Figure 4.	Map indicates type of parking lot.....	7
Figure 5.	Car parked outside airport.....	9
Figure 6.	Parking cost analysis between aviation and non-aviation.....	11
Figure 7.	Map shows price category relative to parking.....	11
Figure 8.	The new design for price category for each facility.....	12
Figure 9.	Online update calendar.....	12
Figure 10.	The designed BOK NOW tools for online parking	13
Figure 11.	Reservation flow diagram.....	15
Figure 12.	Validation result for index page using W3C validator	16
Figure 13.	Context flow diagram	17
Figure 14.	Flow diagram for issued ticket.....	17
Figure 15.	Information Flow diagram.....	18
Figure 16.	The designed GUI for the desktop application.....	19
Figure 17.	Index page with exit date picker	19
Figure 18.	Level of user authentication	19
Figure 19.	Login authentication for the user	20
Figure 20.	Details of parking information.....	20
Figure 21.	Vacant space retrieved from the database	21
Figure 22.	Architectural design for the new system.....	21
Figure 23.	Illustrated XAMPP control panel.....	22
Figure 24.	MySQL in XAMPP	23
Figure 25.	Creating the main database	24
Figure 26.	Scripting code written with JQUERY plugins for date	24
Figure 27.	Html5 taxonomy	25
Figure 28.	Notepad++ text editor	26

TABLES

Table 1. Code of description for attribute	8
--	---

1. INTRODUCTION

In most car parking systems, especially in areas where parking is scarce, one must pay to park. Apart from air services, most airports offer a service called "airport parking garage" for travelers. When drivers turn up to the airport and try to park their car without pre-booking a space, they may have to pay a higher price.

Parking is an essential component of the transportation system. Vehicles must park at every destination. Parking traffic search can lead a significant contribution to airport traffic congestion during its peak hours. In fact, up to 50-75% of all traffic jams are caused by searching a parking space at the airport (Victoria Transport Policy Institute 2012).

In airports PGI system are in use. The PGI system only indicates the number of vacant spaces and the direction where to find them. Even though a number of PGI systems are installed, airports cannot reduce the overall trafficking during its peak hours and the dominant approaches in these systems are to minimize the queue time at the airport.

There are different modes of parking, for example "*meet and run*" for short period and "*holiday parking*" for long term parking.

Online car booking system is a desktop application which instantly adds value to client website with a graphical user interface and web users access to book a parking space, pay online, update the booked space before the exit date, find the way to available vacant space and the status of pre-booking before them heading to the airport.

1.1 STATEMENT OF THE PROBLEM

Customers often choose the parking service in order to reduce their cost. Parking facilities at airports usually use state-of-the-art technology. If a PGI system is implemented at the airport, this system can only display the number of vacant lots and the direction to the lots. The system cannot provide information about the exact location, the price rate, other alternatives and free space in advance. This information is only visible when the driver approaches the parking lots.



Figure 1. PGI indicates the available lots

Another major drawback is the time it takes to find a vacant lot, especially in peak hours. A study (Park study 2011) showed that 86 % of drivers face difficulty finding a parking space. Finding space during peak hours can take more than 10 minutes for more than 66% of the visitors.

During rush hour drivers mostly park their car an improper way, a car may be parked in such way that it occupies two parking lots rather than one. Improper parking happens when a driver is not careful about another derivers' rights. The driver may notice his/her improper parking after leaving his car.



Figure 2. Improper Parking

1.2 THE PURPOSE OF THE STUDY

In most countries where cars are the most dominant transport system, parking spaces are still an unsolved problem. A study conducted by (Rail Transit in America 2012) has shown that most traffic congestion is caused by a long drive round in the parking garage. As a result traffic congestion, insufficient parking space and polluted crops are increasing in the surroundings. For example, countries like China and the USA contribute up to 40-45% air pollution and over congestion traffic on the road.

This study aims to:

1. Implement a comprehensive online desktop application including different services and fair price.
2. Explore the feasibility and Acceptability of the application.
3. Allocate the disabled bays and other reserved bays automatically through graphical user interface.

1.3 IMPORTANCE OF THE STUDY

Parking facilities are a major cost to the society, and parking conflicts are among the most common problems faced by designers, operators, planners and other officials. Such a problem exists either in terms of supply (too few spaces are available) or in terms of management (available facilities are used inefficiently and should be better managed). Management solutions tend to be better than expanding supply because they support more strategic planning objectives (Agency of Commerce 1998).

The following are the benefits of the study:

1. **Possible jams are avoided:** In the rush hour, long traffic jams always appear at the gates. To cut down such redundancy, the online application must be escalated.
2. **It makes management easy:** once the desktop application is designed, the airport can have a fully integrated central management system through a virtual private network and WiFi.
3. **It saves time:** Clients can directly gain access of a booking space without wasting time driving around for a parking lot.
4. **It offers better service:** The application offers all services in one package so that the customer confidentiality may be maintained.
5. **It provides precise monitoring:** The booking system clearly indicates all spaces that are available, for example, those spaces that occupied are marked with red, and those spaces that are free marked with green.

1.4 LIMITATIONS OF THE STUDY

The following are the limitations of the study:

1. As indicated in Figure 3, both entry and exit date are set at the same time.
2. The designed online booking system is not fully integrated with some other online parking systems.
3. The booking system is not automated with card dispenser, automatic boom barrier, long range RFID reader and lane camera.
4. The booking system has not dynamically directed a certain group of users to specific zones.

5. The designed system cannot fully display the digital map of the parking lot.
6. Secure socket layer was not set up, as a result information available on the credit card are not encrypted during payment.

The image shows a web-based date picker for October 2013. At the top, there are navigation arrows and the text "October 2013". Below this is a calendar grid with days of the week (Su, Mo, Tu, We, Th, Fr, Sa) as headers. The dates 1 through 31 are displayed in the grid. The date 4 (Friday) is highlighted in yellow. Below the calendar grid, there are two input fields: "10/4/2013" and "00h00" with a dropdown arrow. Below these, there are labels "Car Park Exit Date" and "Time", followed by another "10/4/2013" input field and a "00h00" dropdown. At the bottom, there is a yellow bar with the text "Manage my booking" and an orange button labeled "BOOK NOW".

Figure 3. Online date picker for entry and exit calendar

1.5 IMPLICATION OF THE STUDY

Desktop Online car parking applications are an emerging technology, this thesis discusses how to book car reservation online with the help of a user-friendly desktop application. In most airports, parking lots are limited so that the desktop application can fully guide a user directing traffic to any area and display all the necessary information of the parking lot at any given time.

2. LITERATURE REVIEW

2.1 INTRODUCTION

A desktop online car park reservation system is built with PHP, MySQL, JAVASCRIPT, JQUIER, HTML5 and CSS. Building a website and adding some futures may increase booking and make it easy for drivers to book a free space in online.

The online car park reservation system allows drivers to book a free space, pay online, add extra services and update the reservation period at any time using the "**Manage My Booking**" tools.

The front-end is customizable through the admin panel and also available in a full package for program developers for future improvement.

Parameters that are indicated in Table 1 are intended to show attributes used for the development process.

2.2 ONLINE CAR PARKING MANAGEMENT PRINCIPLE

The car park management principle defines the purpose and priority of car parking .The principles are enforced and implement at the airport along with the standard traffic signs and availability of parking facilities. The principles are not based on first come, first served basis, but the drivers must have to have a user account before heading to the airport. Once a driver has a member login account, he/she has the right to choose parking options. In this system, it is not possible to make a reservation without logging.

2.3 CASE STUDY: TYPES OF PARKING LOT

❖ Holiday

Among from several parking facilities the "holiday" is the right place to start holiday parking. "holiday-parking" offers an affordable price for short or long period parking.

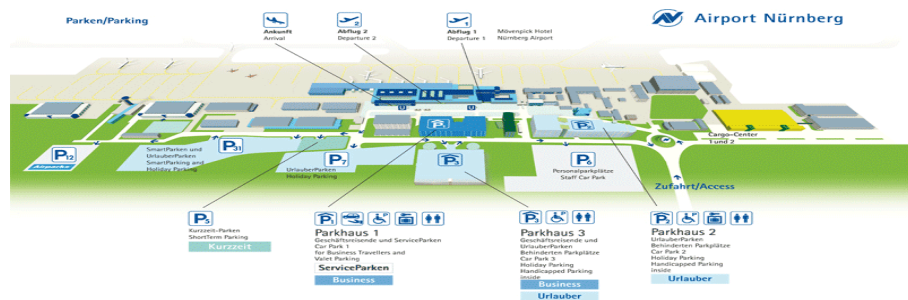


Figure 4. Map indicates type of parking lot

Source: <http://www.airport-nuernberg.de/parking-lots>

❖ Car parks for the disabled

This kind of parking facility can only serve disabled drivers who have a free parking permit.

Drivers rely on a free mode of parking and they do not have to pay a parking fee when they leave the parking space.

❖ Collecting and dropping off

When drivers pick up or drop passengers off at the airport, they are allowed to park onto the "surface parking".

Table 1. Code of description for attribute

Car Park	
Attribute	Description
Car park reference	An external mark to the car park
Car park name	The name entitled for the location of the airport
Car park URL	A public domain
Car park and ride indicator	Boolean value to indicate if the car parked at instant ride point
Car park record last	The data record for an update version

❖ Off airport parking

Parking lots which are located outside the airport boundary are known as "off-airport parking". Even though these parking lots are outside the airport boundaries, it is worth checking the transfer time in order to be punctual at the airport. Fortunately if the drivers are willing to travel a bit further from the terminal, they can save money by parking at an off-airport garage.



Figure 5. Car parked outside airport

2.4 PARKING GUIDANCE INFORMATION SYSTEM (PGI)

A Parking Guidance Information System (PGI) presents dynamic information on parking in controlled areas. The system is combined with traffic monitoring, processing, and changeable message signs in order to provide the service. Searching a vacant space at the airport without a relevant information is a major cause of conjunction and pollution during rush hours. The role of a Parking Guidance Information System is to reduce the parking search traffic in large parking facilities by directing drivers to car parks where occupancy levels are low.

The PGI indicates the number of available parking lot of a particular parking zone, thus giving the total number of empty spaces in the specified parking zone. However, the system does not fully display all the necessary information in a full scale, for example the PGI does not tell the driver where exactly to park the car.

2.5 PGI SYSTEM EVALUATION

In recent years, excessive parking search time has been identified as a significant contributor to airport congestion and as an important influence in destination choice. Recent studies of parking behavior have found that parking search time can constitute up to 25% of the average total travel time, and it usually takes about 1.5 to 2 times of the value of driving time (Polka and Axhausen, 1990).

The following evaluation is based on a particular PGI sign which is fixed at a particular point:

- It reduced search time by allocating drivers to the appropriate facility is about 25%.
- To some extent diverting drivers away from an attempt to park on-street is only indicated by a single arrow to the parking zone. It is quite sophisticated to allocate the exact position.
- Reducing queuing is not quite moderate if the system is compared with the design desktop application.

2.6 ECONOMICS OF PARKING

The dynamic change of parking facility affects the economic affiliations of the sector. Airport parking is the most congested parking place for motor vehicle as well as time-consuming and expensive. The following trend diagram shows how airport parking affects the investment return cost in aviation and non-aviation portfolio.

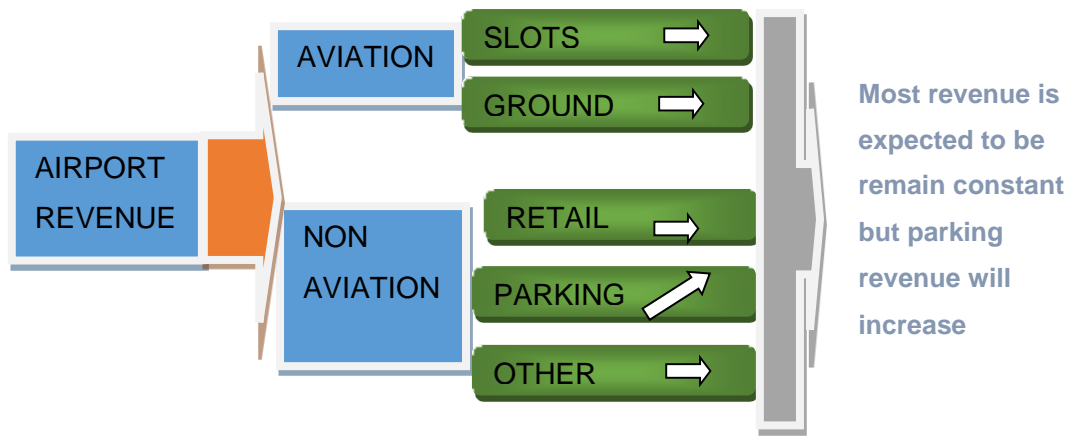


Figure 6. Parking cost analysis between aviation and non-aviation

2.7 PARKING PAYMENT

Parking payments are usually made through inserting a coin into the cash dispatch machine or using a card when upon exit. In this thesis proposal, the drivers can pay a parking fee using an online payment method, e.g. credit card. Once the payment is made online, the driver can have a chance to get a seasonal ticket so that she/he has reduced parking fee. Once the booking and payment are made online, the driver can get the tickets from the help desk or they can print them out.

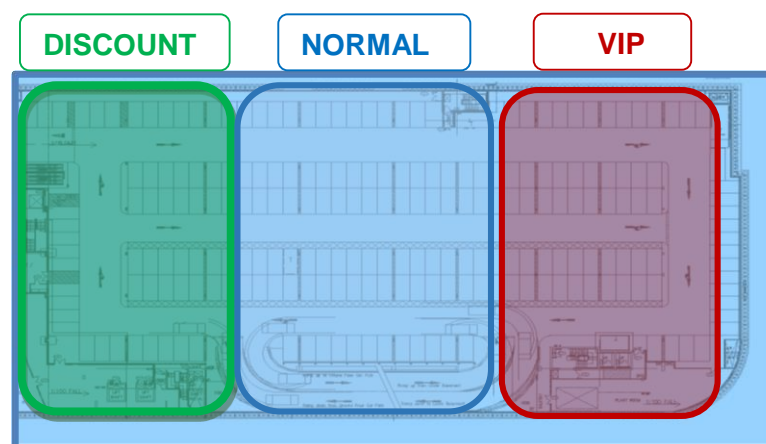


Figure 7. Map shows price category relative to parking

Figure 8 shows the price categories, once the customer has made a decision to park a car at a particular lot, the database automatically calculates the price index and saves it in the database.

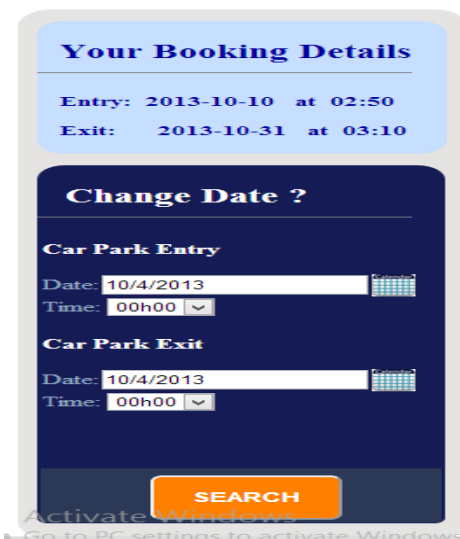
In the application the following parking options are labeled, for example P5, P4A, P3, P2 and P1, once the customer has made a booking, then the application automatically retrieves data from the database and assigns the price according to the different parking options in descending order.

P₅	 12mins walk	The official airport car park for Terminal 2 is also the most popular, with regular, reliable transfers to the airport	€456	book
P_{4A}	 8mins walk	Drop your car off at the terminal-2, just a eight-minute stroll from departures	€216	book
P₃	 7mins walk	The closest Park and Ride to Terminal 2.	€456	book
P₂	 2mins walk	The best value Meet and Greet service at Terminal 1	€552	book
P₁	 1min walk	The most convenient parking at Helsinki-Vantaa terminal 1. Simply drop and go. Hand over your keys and stroll to check-in	€720	book

Figure 8. The new design for price category for each facility

2.8 CAR PARK DESKTOP APPLICATION

The desktop car park booking is a web based reservation system designed to make it easier for driver to book parking spaces online. The application is easily customizable and provides web users with an easy way to reserve parking space, pay online, and manage bookings.



Your Booking Details

Entry: 2013-10-10 at 02:50
Exit: 2013-10-31 at 03:10

Change Date ?

Car Park Entry
Date: 10/4/2013
Time: 00h00

Car Park Exit
Date: 10/4/2013
Time: 00h00

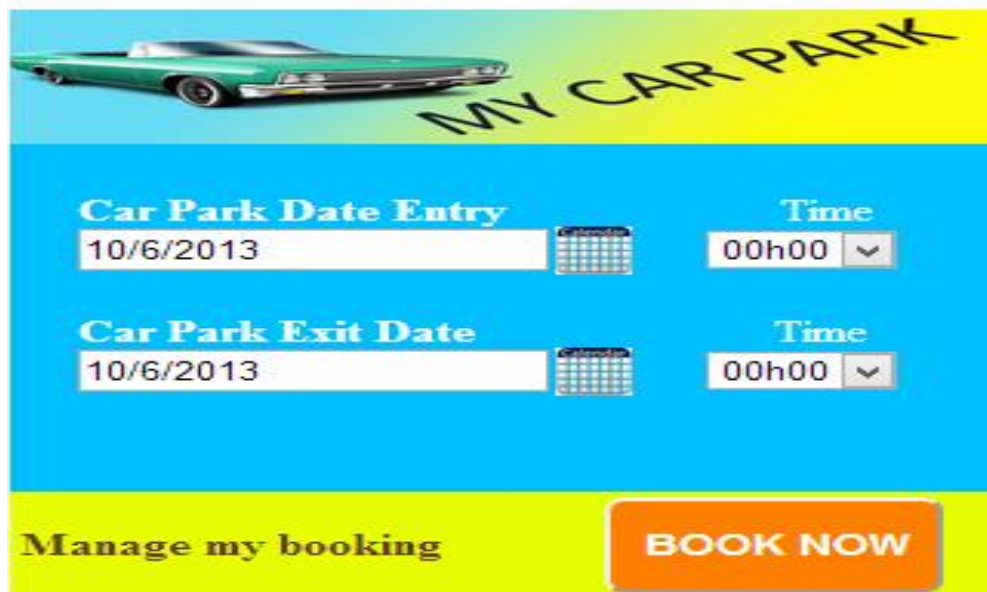
SEARCH

Figure 9. Online update calendar


2.9 MAKING A BOOKING


Making a booking is the process of booking a parking lot. A booking can be made as follows:

1. The user selects the booking calendar time from the quick quotes screen and clicks "BOOK NOW".
2. The user then selects the parking facilities that best suits.
3. The user fills in his/her personal data.
4. The user pays.
5. The user prints out the receipt or is sent an email with the booking information.



MY CAR PARK

Car Park Date Entry
10/6/2013  **Time**
00h00 

Car Park Exit Date
10/6/2013  **Time**
00h00 

Manage my booking **BOOK NOW**

Figure 10. The designed BOK NOW tools for online parking

3. METHODOLOGY

3.1 RESEARCH AND REVIEW

To develop the desktop online car parking reservation system, research of literature on the concept of a desktop application, car park facility and earlier work done on parking facility was reviewed by reading updated books, published journals, searching on the internet using different searching engines like Google. The outcomes of research and review helped in understanding of the different mode of airport parking facilities.

3.2 ANALYSIS DESIGN

Object-oriented analysis and design phase were used for the online desktop application system. The following section summarizes the data collection methods and assessment tools.

3.2.1 ANALYSIS TOOLS

The main aim is to show system requirements and methods to analyze the required data. The required data were analyzed using tools that include decision table and flow chart to assess the current system and come up with the new design outline.

The tools used for the design and analysis include:

- (i) Data Flow Diagram: It was used for graphical representation of the flow of data through an information system. It helps to show how data move in a graphical fashion so that we can easily identify the system components and process.
- (ii) Entity relation diagram: It illustrates the logical structure of the system database used to show the relationship between the entities involved in the system together with their attributes.

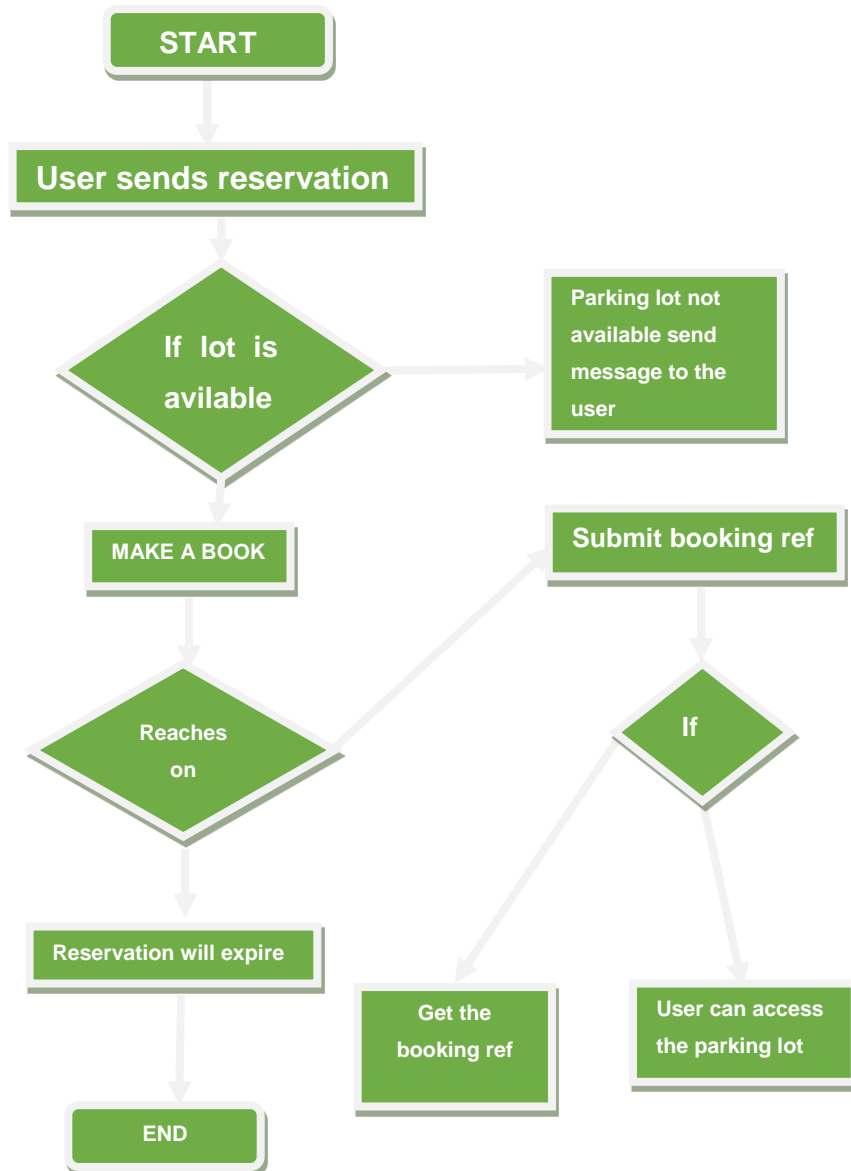


Figure 11. Reservation flow diagram

3.3 ARCHITECTURAL IMPLEMENTATION

The analysis tools were used to show the detailed assessment of the structure and the function. However, the system architecture is the conceptual model of the system that defines views, structure and behavior. In this thesis the implemented architectural system has two components: a database management system and a user interface.

The user is connected to the webpage through an interface generated by PHP and JavaScript. The database storing all the necessary data related to the car park was created using MySQL and managed by DBMS. To connect the DBMS and PHP, APPACHE is used as a bridge linker. All these packages are supplied by XAMPP. Therefore, the desktop application system was developed based on J2EE, PHP and DBMS.

3.4 VALIDATION

Pages on the website must to be validated to ensure the norms or standard which means they have to meet the standards set by W3C and pass a variety of validation for CSS and XHTML.

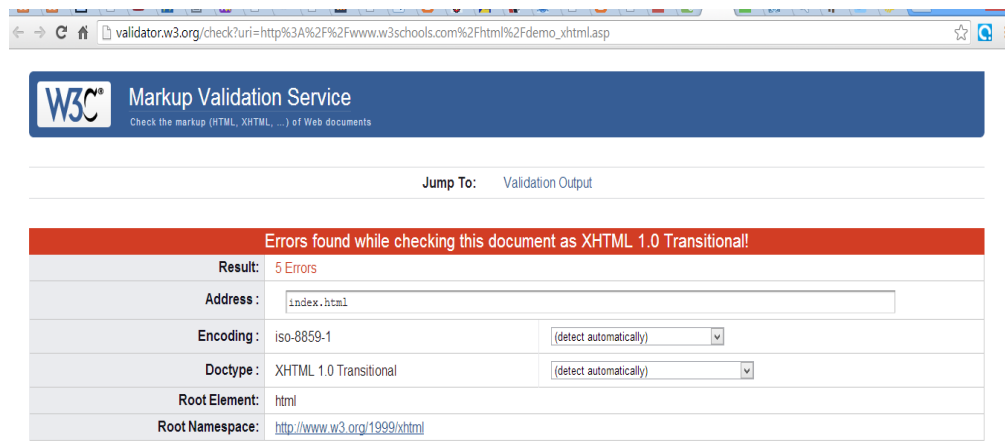


Figure 12. Validation result for index page using W3C validator

4. SCHEME ANALYSIS

4.1 CONTEXT FLOW DIGRAM

The following context diagram defines the boundary between the system and its environment and shows the entities that interact with it.

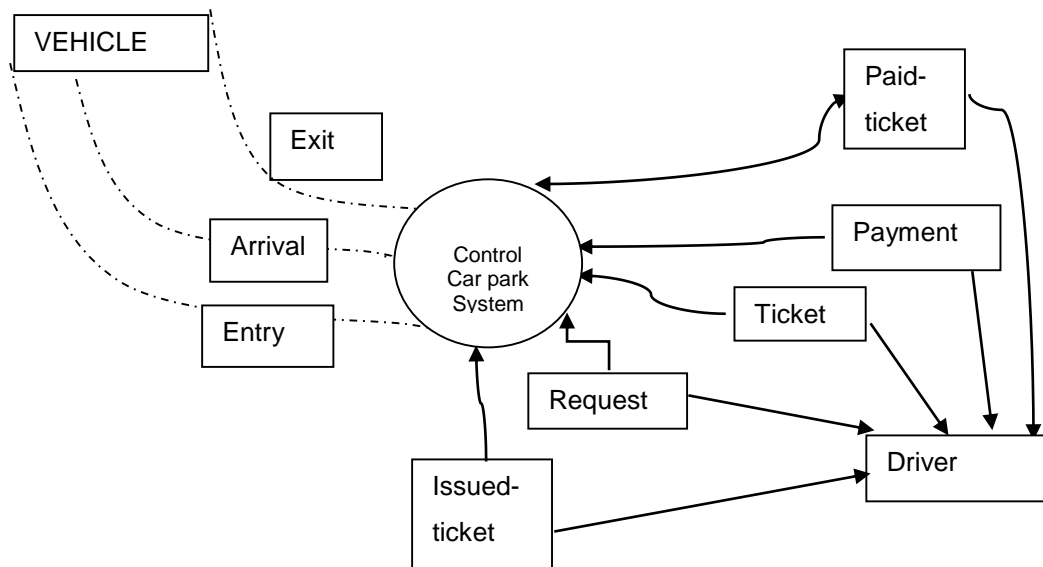


Figure 13. Context flow diagram

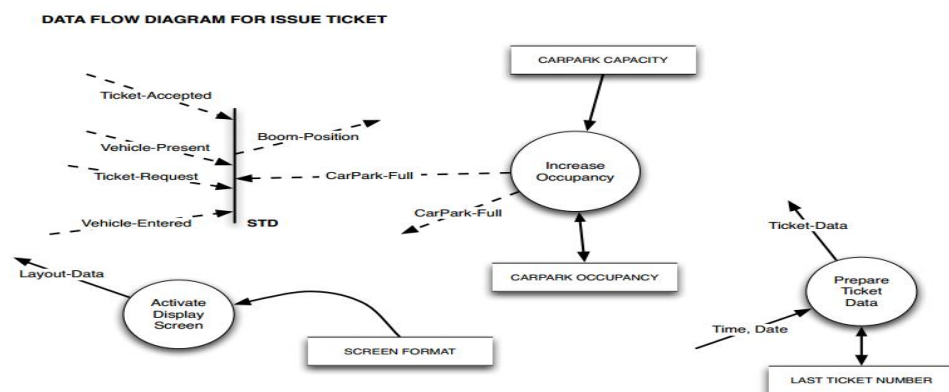


Figure 14. Flow diagram for issued ticket

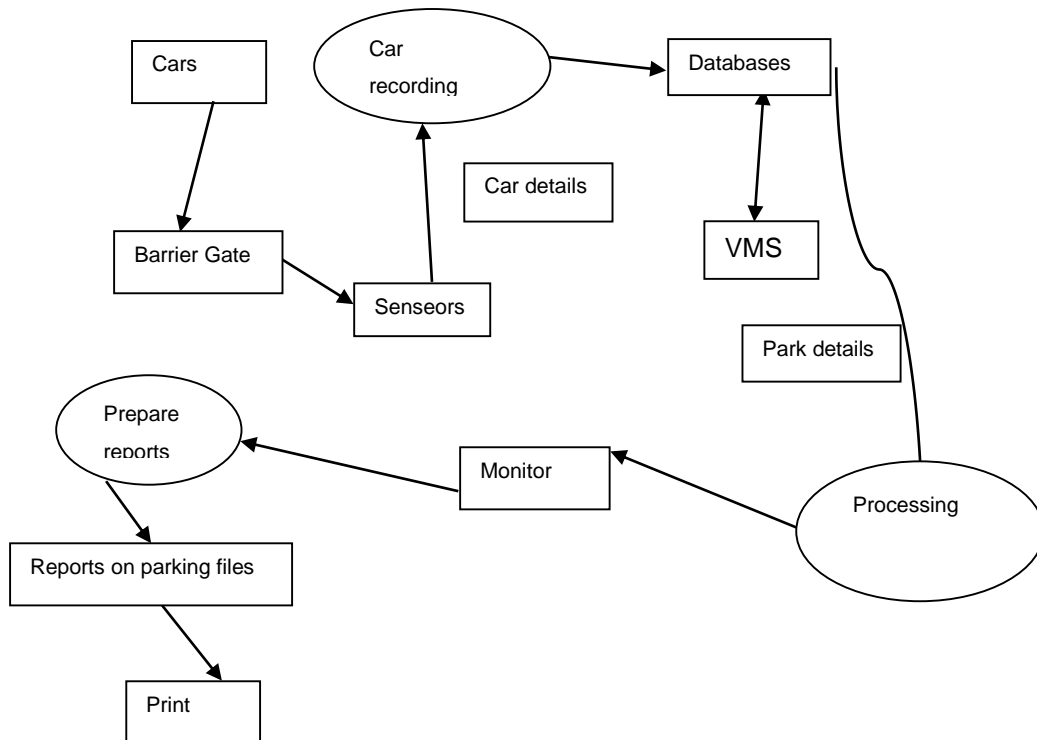


Figure 15. Information Flow diagram

4.2 GRAPHICAL USER INTERFACE (GUI)

Designing the visual composition of a GUI is an important part of the online application. It aims to enhance the efficiency and ease of use of the underlying logical design. The GUI consists of various forms, for example, submit form for registration, data picker for entry and exit calendar, validation for payment. The horizontal navigational menu is added on every page in order to facilitate the booking process.

For interactive purpose only, swapping images are added in the index page. When a user browses a page, the page is downloaded first and then the images. The swapping images on the page are written in JavaScript (Appendix).

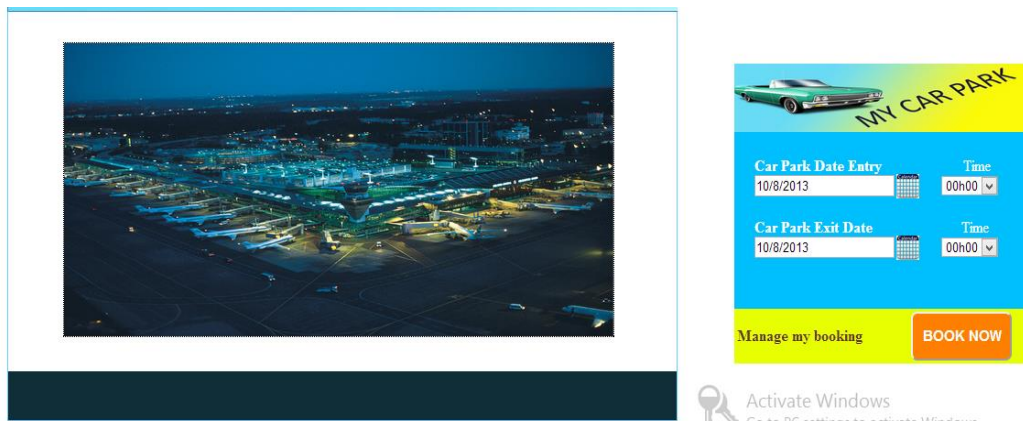


Figure 16. The designed GUI for the desktop application

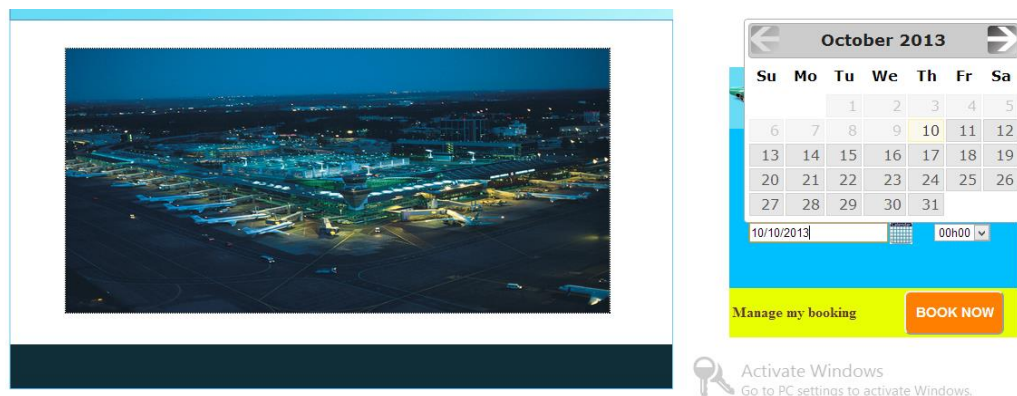


Figure 17. Index page with exit date picker

4.3 LEVELS OF USER AUTHENTICATION

Decision tree is used to show how the users are authenticated at different levels to finalize the booking process.

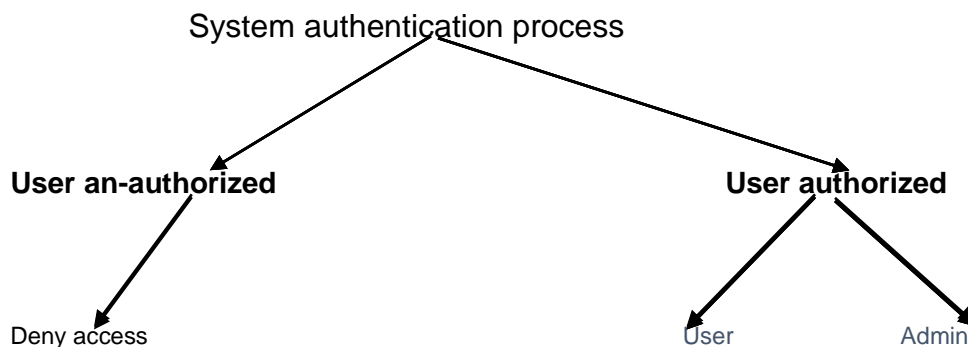


Figure 18. Level of user authentication

From 2013-10-05 00:00 To 2013-10-31 00:00

Username

Password

LOGIN

@copy right Helsinki vantan airport

Figure 19. Login authentication for the user

4.4 SYSTEM DESIGN

4.4.1 THE PROPOSED SYSTEM DESIGN

The system is to give detailed information of the registered customers and is able to generate a report of booked details. In addition, the system enables customers to register, view the status of the parking lot, update, and delete their previous records.

SECURE, CONVENIENT, AIRPORT PARKING
MAKE A BOOKING
MANAGE MY BOOKING

Helsinki Vantaan Airport Parking

From 2013-10-10 02:50 To 2013-10-31 03:10

P₅		12mins walk	The official airport car park for Terminal 2 is also the most popular, with regular, reliable transfers to the airport	€399 book
P_{4A}		8mins walk	Drop your car off at the terminal-2, just a eight-minute stroll from departures	€189 book
P₃		7mins walk	The closest Park and Ride to Terminal 2.	€399 book
P₂		2mins walk	The best value Meet and Greet service at Terminal 1	€483 book
P₁		1min walk	The most convenient parking at Helsinki Vantaan terminal 1. Simply drop and go. Hand over your keys and stroll to check-in	€630 book

PARK	SPACE	AVAILABLE	STATUS
P5	300	191	AVAILABLE

PARK	SPACE	AVAILABLE	STATUS
P4	300	250	AVAILABLE

PARK	SPACE	AVAILABLE	STATUS
P3	300	294	AVAILABLE

PARK	SPACE	AVAILABLE	STATUS
P2	300	289	AVAILABLE

PARK	SPACE	AVAILABLE	STATUS
P1	300	247	AVAILABLE

Your Booking Details

Entry: 2013-10-10 at 02:50
Exit: 2013-10-31 at 03:10

Change Date ?

Car Park Entry

Date:

Time:

Car Park Exit

Date:

Time:

SEARCH

Figure 20. Details of parking information

PARK	SPACE	AVILABLE	STATUS
P5	300	191	AVILABLE

PARK	SPACE	AVILABLE	STATUS
P4	300	250	AVILABLE

PARK	SPACE	AVILABLE	STATUS
P3	300	294	AVILABLE

PARK	SPACE	AVILABLE	STATUS
P2	300	289	AVILABLE

PARK	SPACE	AVILABLE	STATUS
P1	300	247	AVILABLE

Figure 21. Vacant space retrieved from the database

4.5 ARCHITECTURAL DESIGN

Architectural design is high-level design and describes the major components and how they interact with the new system. At this point of system development, the aim is to focus on the central architectural definition of the new system. Both the client and the server have different perspective views.

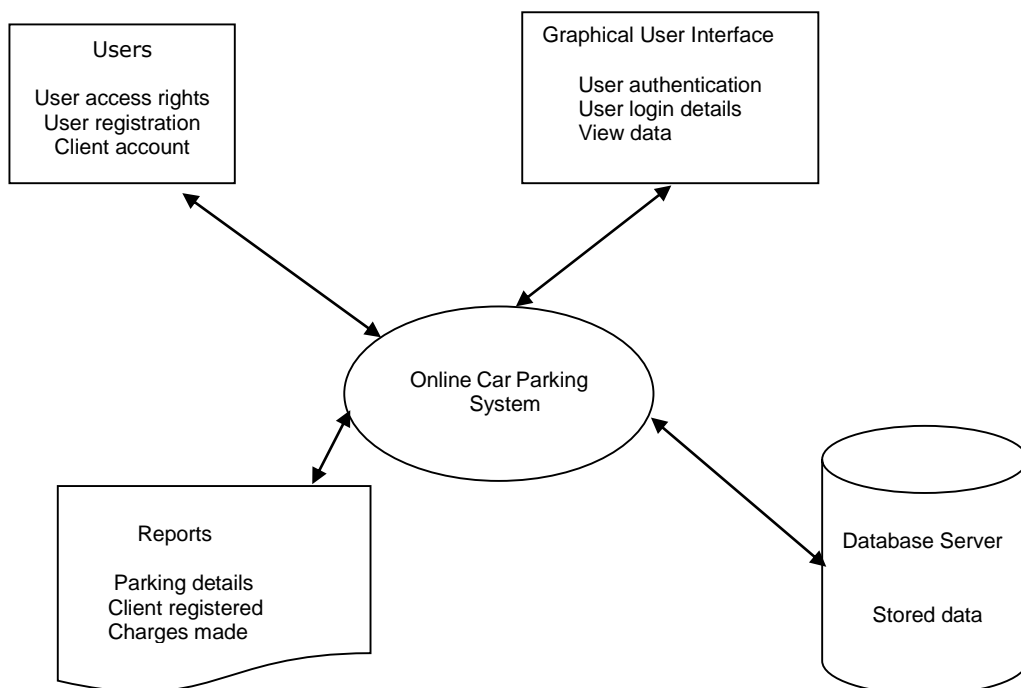


Figure 22. Architectural design for the new system

5. PROGRAM ENVIRONMENT

In order to create the program environment, the following technologies were used:

5.1 XAMPP

XAMPP is free and open source cross-platform web server solution stack package. For the purpose of this thesis, the XAMPP 1.8.2 version was used and the package included the following programming components:

- Apache 2.4.4
- MYSQL 5.5.32
- PHP 5.4.16
- PhpMyAdmin 4.0.4
- FileZilla FTP server 0.9.41
- Tomcat 7.0.41(with_mod_proxy_ajp as coonector)
- Strawberry perl 5.16.3.1 portable
- XAMPP control panel 3.2.1

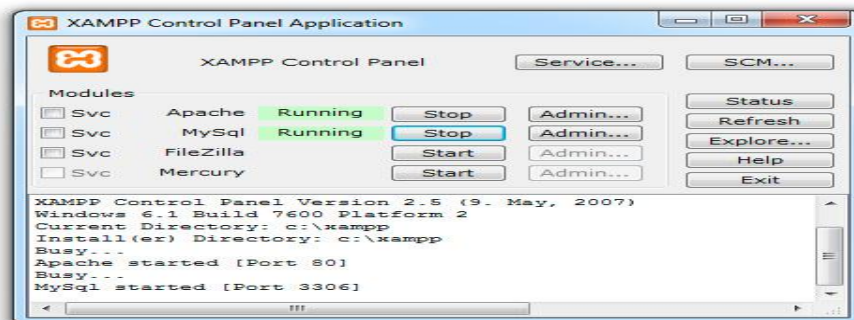


Figure 23. Illustrated XAMPP control panel

XAMPP is free application for windows under the GNU license policy it can be downloaded from www.apachefriends.org/en/xampp.html. In the XAMPP package there are different types of components, such as MySQL database, Apache server and PHP. The Apache web server is more convenient than any other types of web server solution as it allows web site designers and

programmers to test their work without internet access. XAMPP technology also provides the architectural design and manipulation of MySQL and SQLite.

5.2 APACHE TECHNOLOGY

The Apache HTTP Server develops and maintains an open HTTP server for operating systems, including UNIX and Windows NT. Apache technology provides a secure, efficient and extensible HTTP services in synchronization with the current HTTP standards.

5.3 PHP

PHP is a server side scripting language which makes dynamic websites and web applications for the development software and requires minimal set-up. Functions in PHP are reusable bits of code that can be used to make the program more efficient.

5.4 MYSQL

In XAMPP, the MySQL database is an integral part of the package. To create a MySQL database using XAMPP, the following procedure needs to be performed:

1. Open the browser and enter <http://localhost/phpmyadmin/>. This will bring the MySQL setup page.

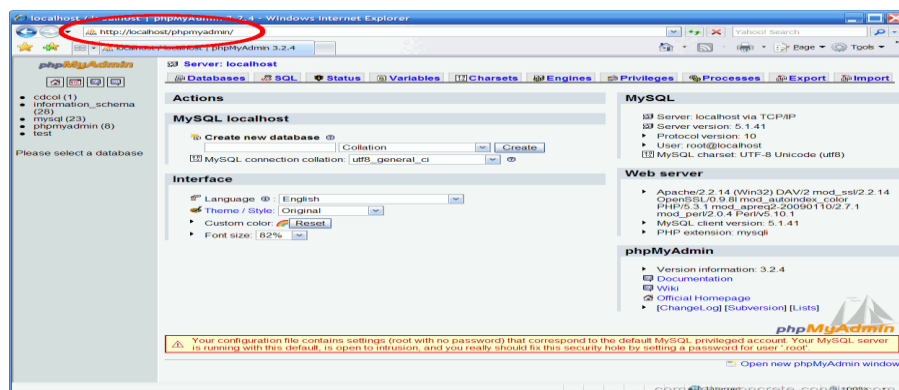


Figure 24. MySQL in XAMPP

2. Enter a name for the database, then click on the create button.

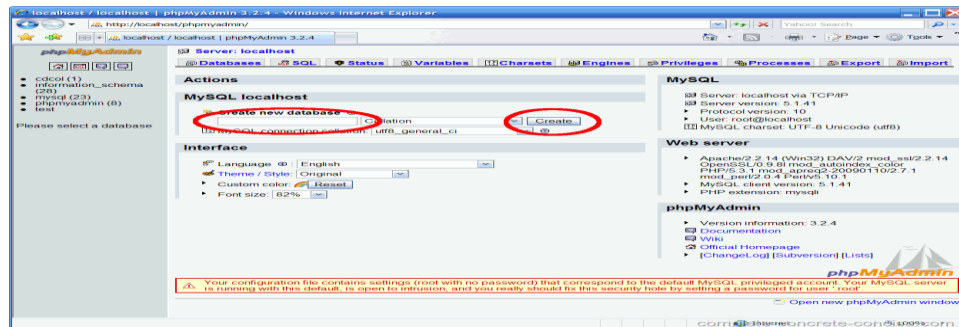


Figure 25. Creating the main database

Once the database is successfully created, the database can now be used by any application that requires MYSQL database.

5.5 JAVASCRIPT

JavaScript is the scripting language of the web, for the software development project it was applied JQUERY for date picker and JavaScript for the swapping image. Every created form in the page was validated with JavaScript validator in order to facilitate the functionality and validate the input forms so that they easily communicate with the web server.

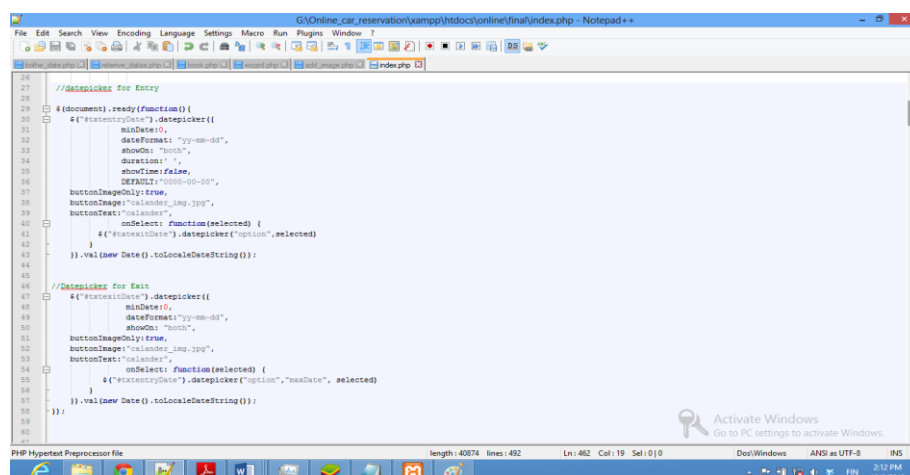


Figure 26. Scripting code written with JQUERY plugins for date

5.6 STYLE SHEETS

Using styles in the contents of the browsed pages on the web can influence the presentation of its visual layout. Different forms of style sheets were applied, for example, external, in-line and internal. The code that was used is shown in the Appendix.

5.7 HYPERTEXT MARKUP LANGUAGE

The hypertext markup is the main method of creating the online parking booking system and displaying the contents on the browser. The type of code is "**TEXT**" and is written in the form of html tag and has a uniform public identifier "index.html".

5.7.1 HTML5

HTML5 is a markup language and the core technology of the internet. HTML5 is more coherent and consistence for structuring and presenting contents for WWW. It is verified so most browser tools support the markup language.

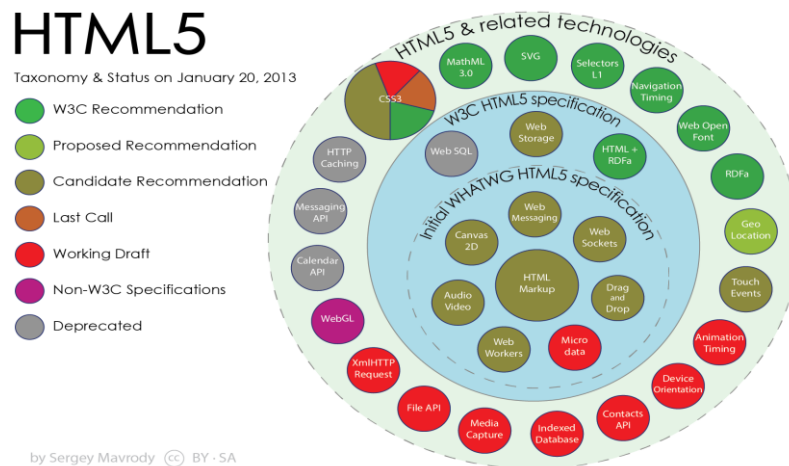


Figure 27. Html5 taxonomy

Source: <http://en.wikipedia.org/wiki/File:HTML5-APIs-and-related-technologies-by-Sergey-Mavrody.png>

6. Conclusion

Drivers always face difficulties in finding available parking space while entering busy airports. The desktop online car park application is perhaps a viable solution to reduce the amount of time needed to search for a vacant car park lot. The drivers can easily be allocated a vacant parking lot based on the information displayed on the booking tools in the web page. Developing this system came from the fact that minimum cost is involved because the internet is used rather than the PGI system.

REFERENCES

Axhausen, J. 1991. Choice of Parking: Stated preference approach [J]. Transportation 59-81.

Feeney, B. A review of the impact of parking policy measures on travel demand. Transport planning and technology, 1989, 13:229-244.

John, R. 2007. Date picker calendar. USA. JQuery date picker Google code project. Available at: (<http://blog.jquery.com/category/jquery-ui/>).

Moshe E. B. (1985) Discrete choice analysis: Theory and application to Travel Demand. Massachusetts: Massachusetts Institute of Technology.

Todd, L. 2012. A Comprehensive evaluation of benefits of parking. USA: Victoria Transport Policy Institute.

HTML5 web development. Available at: <http://www.w3schools.com/2013>

Appendix

SAMPLE SOURCE CODE LISTING

<!-------Date picker for exit and entry date----->

```
$(document).ready(function () {

    $("#txtentryDate").datepicker({

        minDate: 0,

        dateFormat: "yy-mm-dd",

        showOn: "both",

        duration: ' ',

        showTime: false,

        DEFAULT: "0000-00-00",

        buttonImageOnly: true,

        buttonImage: "calander_img.jpg",

        buttonText: "calander",

        onSelect: function (selected) {

            $("#txtexitDate").datepicker("option", selected)

        }

    }).val(new Date().toLocaleDateString());

//Datepicker for Exit
```

```

$("#txtexitDate").datepicker({
    minDate: 0,
    dateFormat: "yy-mm-dd",
    showOn: "both",
    buttonImageOnly: true,
    buttonImage: "calander_img.jpg",
    buttonText: "calander",
    onSelect: function (selected) {
        $("#txtentryDate").datepicker("option", "maxDate", selected)
    }
}).val(new Date().toLocaleDateString());
});
<!-------END----->
<!-------Javascript for Sawping image----->
<script type='text/javascript'>

var imageID=0;

function changeimage(every_seconds){

    //change the image

    if(!imageID){

        document.getElementById("myimage").src="swap_image.jpg ";

        imageID++;

    }

    else{if(imageID==1){

```

```

        document.getElementById("myimage").src="swap_image.png ";

        imageID++;

    }else{if(imageID==2){

        document.getElementById("myimage").src="swap_image_2.jpg";

        imageID=0;

    }}}

    //call same function again for x of seconds

    setTimeout("changeimage("+every_seconds+")",((every_seconds)*1000));

}

</script><!--END OF SCRIPT-->

<!-------END----->

<!-------Code for Registration----->

a, A:link, a:visited, a:active

        {color: #0000aa; text-decoration: none; font-family: Tahoma,
Verdana; font-size: 14px}

        A:hover

        {color: #ff0000; text-decoration: none; font-family: Tahoma,
Verdana; font-size: 14px}

        p, tr, td, ul, li

        {color: #000000; font-family: Tahoma, Verdana; font-size:
14px}

        th

        {background: #DBEAF5; color: #000000;}

```

```

.header1, h1
    {color: #ffffff; background: #4682B4; font-weight: bold; font-
family: Tahoma, Verdana; font-size: 13px; margin: 0px; padding-left: 2px; height: 21px}

.header2, h2
    {color: #000000; background: #DBEAF5; font-weight: bold;
font-family: Tahoma, Verdana; font-size: 14px;}

.intd
    {color: #000000; font-family: Tahoma, Verdana; font-size:
14px; padding-left: 15px;}

.wcell
    {background: #FFFFFF; vertical-align: top}

.ctrl
    {font-family: Tahoma, Verdana, sans-serif; font-size: 14px;
width: 100%;}

.btnform
    {border: 0px; font-family: tahoma, verdana; font-size: 14px;
background-color: #DBEAF5; width: 100%; height:18px; text-align: center; cursor: hand}

.btn
    {background-color: #DBEAF5; padding: 0px;}

textarea,select,input
    {font: 14px Verdana, arial, helvetica, sans-serif; background-
color: #DBEAF5;}

/* classes for validator */

.tfvHighlight

```

```

        {font-weight: bold; color: red;}

        .tfvNormal

        {font-weight: normal;      color: black;}

</style><!--END STYLE----->

<!-------EXTERNAL RESOURCES----->

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

<script language="JavaScript" src="validatort.js">

</script>

</head><!--END--->

<!-------MAIN BODY----->

<body>

<?php

if(isset($_POST['add']))

{

/*onLoad="document.registration.firstname.focus();"*/

//-----CONNECT INTO THE DATABASE-----
//-----//

```

\$dbhost = 'localhost';

\$dbuser = 'wako';

\$dbpass = '1234';

`$conn = mysql_connect($dbhost, $dbuser, $dbpass);`

`if(! $conn)`

`{`

`die('Could not connect: ' . mysql_error());`

`}`

`//-----CHECK CONFIGURATION OPTION AND SET EITHER 1 OR
0 -----//`

`if(! get_magic_quotes_gpc())`

`{`

`$form_title = addslashes ($_POST['title']);`

`$form_firstname = addslashes ($_POST['first_name']);`

`$form_lastname = addslashes ($_POST['last_name']);`

`$form_user = addslashes ($_POST['username']);`

`$form_pass = addslashes ($_POST['password']);`

`$form_pass_confirm = addslashes ($_POST['confirm_pass']);`

`$form_addi = addslashes ($_POST['street_address']);`

`$form_addii = addslashes ($_POST['street_address_ln2']);`

`$form_cit = addslashes ($_POST['city']);`


```

    $form_post = addslashes ($_POST['post_code']);

    $form_count = addslashes ($_POST['country']);

    $form_tele = addslashes ($_POST['telephone_number']);

    $form_email = addslashes ($_POST['email']);

    $form_general = addslashes ($_POST['general_information']);

    $_POST['username'] = addslashes($_POST['username']);

}

else

{

    $form_id = $_POST['id'];

    $form_title = $_POST['title'];

    $form_firstname = $_POST['first_name'];

    $form_lastname = $_POST['last_name'];

    $form_user = $_POST['username'];

    $form_pass = $_POST['password'];

    $form_pass_confirm = $_POST['confirm_pass'];

    $form_addi = $_POST['street_address'];

    $form_addii = $_POST['street_address_ln2'];

    $form_cit = $_POST['city'];

```

```

    $form_post = $_POST['post_code'];

    $form_count = $_POST['country'];

    $form_tele = $_POST['telephone_number'];

    $form_email = $_POST['email'];

    $emp_general = $_POST['general_information'];

}

//-----+++++++ CHECK THE USER_&_PASS FROM
THE DATABASE +++++-----//

//This code runs if the form has been submitted

if (!get_magic_quotes_gpc()) {

    $_POST['username'] = addslashes($_POST['username']);

    }

    $usercheck = $_POST['username'];

    mysql_select_db('members');

    $check = mysql_query("SELECT username FROM members WHERE username
= '$usercheck'")

                                or die(mysql_error());

```

```

$check2 = mysql_num_rows($check);

//if the name exists it gives an error

if ($check2 != 0) {

    die('Sorry, the username '.$_POST['username'].' is already in
    use.'.'

```

```
//-----+++++++ MY_ACCOUNT INTO THE DATABASE
+++++++-----//
```

```
$sql = "INSERT INTO members "
```

```
"(title,fname,lname,username,password,confirm_password,address,address_two,city,p
ostal_code,country,telephone,email,info) "
```

```
"VALUES('$form_title','$form_firstname','$form_lastname','$form_user','$form_pass','$f
orm_pass_confirm','$form_addi','$form_addii','$form_cit','$form_post','$form_count','$fo
rm_tele','$form_email','$form_general');"

```

```
mysql_select_db('members');
```

```
$retval = mysql_query( $sql, $conn );
```

```
if(! $retval )
```

```
{
```

```
die('Could not enter data: ' . mysql_error());
```

```
}
```

```
/*echo "Entered data successfully\n";*/
```

```
header('Location:order_summary.php');
```

```
mysql_close($conn);
```

```
}
```

```
else
```

```
{
```

?>

```
<div class="wrap">
```

```
  <div class="header">
```

```
    </div><!--header-->
```

```
    <div class="welecome">
```

```
      <div style="margin-top:30px;color:#ffffff">BOOK YOUR CAR PARK PLACE</div>
```

```
      <div style="margin-left:800px;margin-top:-20px"><a href="#" style="text-decoration:none;color:#ffffff">My account</a></div>
```

```
    </div><!--welecome-->
```

```
  <div style="margin-left:200px;margin-top:5px">
```

```
<!-------regular expression validation----->
```

```
// regular expressions or function to validate the format
```

```
var re_dt = /^(d{1,2})-(d{1,2})-(d{4})$/,
```

```
re_tm = /^(d{1,2}): (d{1,2}): (d{1,2})$/,
```

```
a_formats = {
```

```
  'alpha' : /^w+$/,
```

```
  'alphanum': /^[a-zA-Z ]{2,30}$/,
```

```
  'unsigned': /^d+$/,
```

```
  'integer' : /^[+-]?d*$/,
```

```
  'real' : /^[+-]?d*\.\?d*$/,
```

```
  'email' : /^[w-\.]+\@[w\.-]+\.[a-z]{2,4}$/,
```

```
  'phone' : /^[d\.\s-]+$/,
```

```
  'date' : function (s_date) {
```

```

        // check format

        if (!re_dt.test(s_date))

            return false;

        // check allowed ranges

        if (RegExp.$1 > 31 || RegExp.$2 > 12)

            return false;

        // check number of day in month

        var dt_test = new Date(RegExp.$3, Number(RegExp.$2-1),
RegExp.$1);

        if (dt_test.getMonth() != Number(RegExp.$2-1))

            return false;

        return true;

    },

    'time' : function (s_time) {

        // check format

        if (!re_tm.test(s_time))

            return false;

        // check allowed ranges

        if (RegExp.$1 > 23 || RegExp.$2 > 59 || RegExp.$3 > 59)

            return false;

        return true;

    }

},

a_messages = [

```

```

'No form name passed to validator construction routine',

'No array of "%form%" form fields passed to validator construction routine',

'Form "%form%" can not be found in this document',

'Incomplete "%n%" form field descriptor entry. "l" attribute is missing',

'Can not find form field "%n%" in the form "%form%"',

'Can not find label tag (id="%t%")',

'Can not verify match. Field "%m%" was not found',

'"%l%" is a required field',

'Value for "%l%" must be %mn% characters or more',

'Value for "%l%" must be no longer than %mx% characters',

'"%v%" is not valid value for "%l%"',

'"%l%" must match "%ml%"'

// reset previous error if any

    this.a_fields[n_key].n_error = null;

    // check required fields

    if (this.a_fields[n_key]['r'] && !this.a_fields[n_key]['v']) {

        this.a_fields[n_key].n_error = 1;

        n_errors_count++;

    }

    // check length

    else if (this.a_fields[n_key]['mn'] && this.a_fields[n_key]['v'] !=

" && String(this.a_fields[n_key]['v']).length < this.a_fields[n_key]['mn']) {

        this.a_fields[n_key].n_error = 2;

```

```

        n_errors_count++;

    }

    else if (this.a_fields[n_key]['mx'] &&
String(this.a_fields[n_key]['v']).length > this.a_fields[n_key]['mx']) {

        this.a_fields[n_key].n_error = 3;

        n_errors_count++;

    }

    // check format

    else if (this.a_fields[n_key]['v'] && this.a_fields[n_key]['f'] &&
(

        (typeof(o_format_check) == 'function'

        && !o_format_check(this.a_fields[n_key]['v']))

        || (typeof(o_format_check) != 'function'

        &&
!o_format_check.test(this.a_fields[n_key]['v'])))

        ) {

            this.a_fields[n_key].n_error = 4;

            n_errors_count++;

        }

    // check match

    else if (this.a_fields[n_key]['m']) {

        for (var n_key2 in this.a_fields)

            if (n_key2 ==

this.a_fields[n_key]['m']) {

```



```

n_key2;

n_another =

break;

}

if (n_another == null)

return this.f_alert(this.f_error(6,

this.a_fields[n_key]));

if (this.a_fields[n_another]['v'] !=

this.a_fields[n_key]['v']) {

this.a_fields[n_key]['ml'] =

this.a_fields[n_another]['l'];

this.a_fields[n_key].n_error = 5;

n_errors_count++;

}

}

}

// collect error messages and highlight captions for erroneous fields

var s_alert_message = "",

e_first_error;

if (n_errors_count) {

for (var n_key in this.a_fields) {

var n_error_type = this.a_fields[n_key].n_error,

```

```

s_message = "";

if (n_error_type)

s_message =

this.f_error(n_error_type + 6, this.a_fields[n_key]);

if (s_message) {

if (!e_first_error)

e_first_error =

o_form.elements[n_key];

s_alert_message += s_message

+ "\n";

// highlighted state parameters
assigned here

if (b_dom &&

this.a_fields[n_key].o_tag)

this.a_fields[n_key].o_tag.className = 'tfvHighlight';

}

}

alert(s_alert_message);

// set focus to first erroneous field

if (e_first_error[0])

e_first_error = e_first_error[0];

if (e_first_error.focus && e_first_error.type != 'hidden' &&
!e_first_error.disabled)

```

```

        eval("e_first_error.focus()");

        // cancel form submission if errors detected

        return false;

    }

    for (n_key in this.a_2disable)

        if (o_form.elements[this.a_2disable[n_key]])

            o_form.elements[this.a_2disable[n_key]].disabled = true;

    return true;

}

function validator_error(n_index) {

    var s_ = a_messages[n_index], n_i = 1, s_key;

    for (; n_i < arguments.length; n_i++)

        for (s_key in arguments[n_i])

            s_ = s_.replace('%' + s_key + '%',

arguments[n_i][s_key]);

    s_ = s_.replace('%form%', this.s_form);

    return s_

}

function get_element (s_id) {

```

```

        return (document.all ? document.all[s_id] : (document.getElementById ?
document.getElementById(s_id) : null));
    }

```

```

<!-------Function for date----->

```

```

(function( window, undefined ) {

```

```

    // Can't do this because several apps including ASP.NET trace

```

```

    // the stack via arguments.caller.callee and Firefox dies if

```

```

    // you try to trace through "use strict" call chains. (#13335)

```

```

    // Support: Firefox 18+

```

```

    //"use strict";

```

```

    var

```

```

        // The deferred used on DOM ready

```

```

        readyList,

```

```

        // A central reference to the root jQuery(document)

```

```

        rootjQuery,

```

```

        // Support: IE<9

```

```

        // For `typeof node.method` instead of `node.method !== undefined`

```

```

        core_strundefined = typeof undefined,

```

```

        // Use the correct document accordingly with window argument (sandbox)

```

```
document = window.document,

location = window.location,


// Map over jQuery in case of overwrite
_jQuery = window.jQuery,


// Map over the $ in case of overwrite
_$ = window.$,


// [[Class]] -> type pairs
class2type = {},


// List of deleted data cache ids, so we can reuse them
core_deletedIds = [],


core_version = "1.9.1",


// Save a reference to some core methods
core_concat = core_deletedIds.concat,
core_push = core_deletedIds.push,
core_slice = core_deletedIds.slice,
core_indexOf = core_deletedIds.indexOf,
core_toString = class2type.toString,
```

```

core_hasOwn = class2type.hasOwnProperty,

core_trim = core_version.trim,

// Define a local copy of jQuery

jQuery = function( selector, context ) {

    // The jQuery object is actually just the init constructor
    'enhanced'

    return new jQuery.fn.init( selector, context, rootjQuery );

},

// Used for matching numbers

core_pnum = /[+-]?(?:\d*\.|)\d+(?:[eE][+-]?\d+|)/.source,

// Used for splitting on whitespace

core_rnotwhite = /\S+/g,

// Make sure we trim BOM and NBSP (here's looking at you, Safari 5.0 and
IE)

rtrim = /^[\s\uFEFF\xA0]+|[\s\uFEFF\xA0]+$/g,

// A simple way to check for HTML strings

// Prioritize #id over <tag> to avoid XSS via location.hash (#9521)

// Strict HTML recognition (#11290: must start with <)

rquickExpr = /^(?:(<[\w\W]+>)[^>]*|#[\w-]*)$/g,

```

```

// Match a standalone tag

rsingleTag = /^<(\w+)\s*\V?>(?:<\1>|)$/,

// JSON RegExp

rvalidchars = /^[\],:{}\s]*$/,

rvalidbraces = /^(?:^|:|,)(?:\s*\[)+/g,

rvalidescape = /\\"(?:\\b|f|r|n|t)|u[da-fA-F]{4})/g,

rvalidtokens = /"^(\\|\"|\/|\\r|\\n)*"|true|false|null|-?(?:\d+\.\d+|\d+)(?:[eE][+-]?\d+|)/g,

// Matches dashed string for camelizing

rmsPrefix = /^-ms-/ ,

rdashAlpha = /-([da-z])/gi,

// Used by jQuery.camelCase as callback to replace()

fcamelCase = function( all, letter ) {

    return letter.toUpperCase();

},

// The ready event handler

completed = function( event ) {

    // readyState === "complete" is good enough for us to call the
    dom ready in oldIE

```

```

        if ( document.addEventListener || event.type === "load" ||
document.readyState === "complete" ) {

            detach();

            jQuery.ready();

        }

    },

    // Clean-up method for dom ready events

    detach = function() {

        if ( document.addEventListener ) {

            document.removeEventListener(
"DOMContentLoaded", completed, false );

            window.removeEventListener(        "load",
completed, false );

        } else {

            document.detachEvent( "onreadystatechange",
completed );

            window.detachEvent( "onload", completed );

        }

    };

/*-----sample style used for date picker-----*/

.ui-helper-hidden {

    display: none;

}

```



```
.ui-helper-hidden-accessible {  
    border: 0;  
    clip: rect(0 0 0 0);  
    height: 1px;  
    margin: -1px;  
    overflow: hidden;  
    padding: 0;  
    position: absolute;  
    width: 1px;  
}
```

```
.ui-helper-reset {  
    margin: 0;  
    padding: 0;  
    border: 0;  
    outline: 0;  
    line-height: 1.3;  
    text-decoration: none;  
    font-size: 100%;  
    list-style: none;  
}
```

```
.ui-helper-clearfix:before,
```

```
.ui-helper-clearfix:after {
```

```
        content: "";

        display: table;

        border-collapse: collapse;
    }

    .ui-helper-clearfix:after {

        clear: both;

    }

    .ui-helper-clearfix {

        min-height: 0; /* support: IE7 */

    }

    .ui-helper-zfix {

        width: 100%;

        height: 100%;

        top: 0;

        left: 0;

        position: absolute;

        opacity: 0;

        filter:Alpha(Opacity=0);

    }

    .ui-front {

        z-index: 100;
```

```
}

//-----sample style for submission form-----//

#henok{

margin-top:-0.2cm;

margin-right:-0.19cm;

margin-left:-0.19cm;


}

.wrap{

position:absolute;

width:949px;

padding:0px;

margin-left:172px;

margin-top:-8px;

border:solid 1px #6CF;

}


.header{

width:950px;

height:120px;

margin-top:0px;
```

```
background-image:url('header1.jpg');
```

```
background-repeat:no-repeat;
```

```
background-size:950px 170px;
```

```
border:solid 0px #009999;
```

```
}
```

```
#header_border{
```

```
border-radius: 50px 15px 0 0;
```

```
border:0px solid #408AD2;
```

```
}
```

```
.welecome{
```

```
width:949px;
```

```
height:50px;
```

```
margin:0px;
```

```
border:solid 1px #408AD2;
```

```
background:#009999;
```

```
}
```

```
#content{
```

```
width:950px;
```

```
height:1062px;
```

```
border:solid 0px green;
```

```
}
```

```
fieldset{
```

```
padding:1em;
```

```
font:80%/1 sans-serif;
```

```
width:500px;
```

```
border:solid 1px #0000A0;
```

```
}
```

```
label {
```

```
float:left;
```

```
width:25%;
```

```
margin-right:1.5em;
```

```
padding-top:0.2em;
```

```
text-align:right;
```

```
font-weight:bold;
```

```
}
```