GENERAL RESERVATION SYSTEM USING WIRELESS APPLICATION PROTOCOL (WAP)

A Project
Presented
to the Faculty of
California State University, Chico

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
in
Computer Science

by
Amar Babroo Kale
Spring 2009
GENERAL RESERVATION SYSTEM USING WIRELESS APPLICATION PROTOCOL (WAP)

A Project

by

Amar Babroo Kale

Spring 2009

APPROVED BY THE DEAN OF THE SCHOOL OF GRADUATE, INTERNATIONAL, AND INTERDISCIPLINARY STUDIES:

_________________________________
Susan E. Place, Ph.D.

APPROVED BY THE GRADUATE ADVISORY COMMITTEE:

Abdel-Moaty M. Fayek
Graduate Coordinator

Clement H. Luk, M.S., Chair

Leonard W. Fisk, Ph.D.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Figures</td>
<td>iv</td>
</tr>
<tr>
<td>Abstract</td>
<td>vi</td>
</tr>
<tr>
<td><strong>CHAPTER</strong></td>
<td></td>
</tr>
<tr>
<td>I. Introduction</td>
<td>1</td>
</tr>
<tr>
<td>Problem Statement</td>
<td>1</td>
</tr>
<tr>
<td>Project Goal and Objectives</td>
<td>1</td>
</tr>
<tr>
<td>Outline of the Report</td>
<td>2</td>
</tr>
<tr>
<td>II. Application Design</td>
<td>4</td>
</tr>
<tr>
<td>Software Tools</td>
<td>12</td>
</tr>
<tr>
<td>III. Methodology</td>
<td>19</td>
</tr>
<tr>
<td>Project Approach</td>
<td>19</td>
</tr>
<tr>
<td>Setting up the Development Environment</td>
<td>19</td>
</tr>
<tr>
<td>Implementation</td>
<td>21</td>
</tr>
<tr>
<td>Testing</td>
<td>52</td>
</tr>
<tr>
<td>IV. Summary</td>
<td>55</td>
</tr>
<tr>
<td>Future Enhancements</td>
<td>55</td>
</tr>
<tr>
<td>Conclusion</td>
<td>56</td>
</tr>
<tr>
<td>References</td>
<td>58</td>
</tr>
<tr>
<td>Appendix</td>
<td></td>
</tr>
<tr>
<td>A. Additional Screen Shots</td>
<td>61</td>
</tr>
</tbody>
</table>
### LIST OF FIGURES

<table>
<thead>
<tr>
<th>FIGURE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Function Hierarchy Diagram</td>
<td>5</td>
</tr>
<tr>
<td>2. Architecture of General Reservation System</td>
<td>8</td>
</tr>
<tr>
<td>3. Class Diagram for General Reservation System</td>
<td>11</td>
</tr>
<tr>
<td>4. Database Model Diagram</td>
<td>13</td>
</tr>
<tr>
<td>5. Website Home Page</td>
<td>24</td>
</tr>
<tr>
<td>6. Customer Registration Form</td>
<td>25</td>
</tr>
<tr>
<td>7. User Login Screens</td>
<td>26</td>
</tr>
<tr>
<td>8. User Home Page</td>
<td>27</td>
</tr>
<tr>
<td>9. Flight Reservation Page</td>
<td>27</td>
</tr>
<tr>
<td>10. Search Flights</td>
<td>28</td>
</tr>
<tr>
<td>11. Flight Search Results</td>
<td>29</td>
</tr>
<tr>
<td>12. User Payment Options</td>
<td>29</td>
</tr>
<tr>
<td>13. Reservation Confirmation</td>
<td>30</td>
</tr>
<tr>
<td>14. Flights Page</td>
<td>31</td>
</tr>
<tr>
<td>15. Flight Cancellation Page</td>
<td>32</td>
</tr>
<tr>
<td>16. Car Reservation Page</td>
<td>32</td>
</tr>
<tr>
<td>17. Car Search Page</td>
<td>33</td>
</tr>
<tr>
<td>18. Car Results Page</td>
<td>34</td>
</tr>
</tbody>
</table>
ABSTRACT

GENERAL RESERVATION SYSTEM USING WIRELESS APPLICATION PROTOCOL (WAP)

by

Amar Babroo Kale

Master of Science in Computer Science

California State University, Chico

Spring 2009

Yesterday, travelers on the go called 800 numbers from their mobile phones to make or change airline reservations and reserve rental cars. Today, approximately 33% of mobile phone owners use their cell phones exclusively for internet access. Mobile phones that subscribe to internet services are continuously enabled for internet connectivity using wireless web services. The “General Reservation System using Wireless Application Protocol (WAP)” implements a website for wireless users using Wireless Application Protocol (WAP). It provides a service for booking rental cars and airlines tickets via cell phones. The web based reservation system offers the same full service functionality as a regular web based service but in an environment adapted to a mobile phone. Using this system, travelers are no longer restricted by access to an internet-enabled computer. Searches can be initiated using date of travel to find the most
up-to-date information and change bookings. Information regarding availability of
flights and cars is accessible in real time using dynamically created web pages. Naviga-
tion on the site is rendered simple to use on handheld devices by the WAP protocol un-
derlying the site design. Using the WAP based service for mobile phones, travelers on
the go have real flexibility and the opportunity to save time and frustration by remain-
ing in control of their travel itinerary at all times.
CHAPTER I

INTRODUCTION

In the 21\textsuperscript{st} century, we have come to expect mobility, even depend on it. Users do not just want access to information anytime, anywhere, but they also expect service at the same level. Originally, cell phones were used primarily for making and receiving telephone calls. Today, more and more applications are available from cell phones featuring wireless technology. With cell phone technology, you are no longer constrained by having to be at a computer with an internet connection [1]. You can read a review of the latest movie, do your shopping, and even transfer money from your savings to your checking account, all from your cell phone.

Problem Statement

This project develops a cost effective solution based on wireless technology for a car and flight reservations system. The system is accessible anytime, anywhere using a cell phone and features cell phone friendly interfaces that are easy for cell phone users to navigate and use.

Project Goal and Objectives

The goal of this project is to design and implement a website using Wireless Application Protocol (WAP).
In order to accomplish the above goal, the following objectives are defined:

1. Provide a user interface that is both aesthetically pleasing and easy to use.
2. Create a Customer Account and Log-in.
3. For a given search-by date and location, provide information on all available flights or all available rental cars.
4. Reserve, view and cancel a flight reservation and/or a car reservation
5. Check your reserved flights or cars.
6. Provide user account management options such as store location and credit card information.
7. Eliminate the “cable tangle,” referred to by H. Peter Alesso in his article “The Intelligent Wireless Web: Our Next Generation Web,” that results from the need to be connected to the internet with a wired desktop computer [2].

Outline of the Report

Chapter I presents the background of the project, includes the problem statement, and lists the overall high-level goals and objectives of the project.

Chapter II of the project is divided into two sections. The first section covers the proposed design for the project. The design section covers the website design and explains the working of each part. The second section covers the software tools that are used to build the website. Each software tool (language or tool) is explained and its advantages are stated in context to the website.

Chapter III covers the methodology used to build the website. It includes four sections. The first section covers the overall project approach. Section two describes the
process of setting up the development environment. Verification of the website and its ability to work properly is insured by installation of the software as well as by running Sanity Checks software. Section three presents the implementation phase. The implementation is done in three phases: Front End, Back End, and Database. This section includes screen shots of the application and an additional screen shot is included in Appendix A. The last section covers the testing of the website. The testing phase begins early in the development cycle, so the cost of fixing bugs is kept to a minimum.

Chapter IV presents future enhancements for this project with a focus on adding more functionality to the present website as well as making the website more sophisticated and secured. The project ends with a conclusion.
CHAPTER II

APPLICATION DESIGN

The website allows users to make, change or check flight reservations, and make, change or check car reservations. From the home page, the user will have access to either of the two major sub-systems – flight reservations or car reservations. The system is therefore called a General Reservation System.

- From the flight reservation system users will be able to:
  - Search for flights
  - Reserve flights
  - Cancel flights
  - Confirm selections

- From the car reservation system users will be able to:
  - Search for cars at a given location
  - Select and reserve a car at that location
  - Cancel an existing car reservation

Figure 1 depicts the Function Hierarchy Diagram for the website.

The website design is based on the MVC (Model, View, and Controller) architecture. This architecture separates the three views, model, view and controller. The Markup Language Tutorial by eNode.com explains how the separation between the three categories is key to the architecture by providing abstraction layers that allow you to
Fig. 1. Function hierarchy diagram.
change one part of the model without impact to the others [3]. This is important to my project because the website will be developed for all the different devices. The MVC architecture allows fast access to dynamic web pages because much of the processing logic and business rules are maintained on the Web Server.

WAP technology enables the application to run wirelessly from a cell phone. WAP is the industry standard for wireless applications and so it is a good choice for this project. It can be used on any wireless device and is particularly suitable for handheld devices such as mobile phones, smartphone and pagers and it is supported by the primary mobile device companies such as IPhone, Ericsson, Nokia and Motorola. WAP uses a browser application known as a micro-browser that works within the constraints of handheld devices such as low bandwidth, small screen size and memory. The underlying transport protocol is based on Hyper-Text Markup Language (HTML) running over Transmission Control Protocol/Internet Protocol (TCP/IP) so it is compatible with other web-based protocols and therefore suited to my application [4]. In his article “WAP Today and Tomorrow,” Goldman explains how different Internet protocols are used for the wireless applications [4].

Additional logic and business rules for the application (compiled WMLScripts to verify the user input) are maintained in the WAP Gateway that supports the wireless protocol. A WAP Gateway is provided by the wireless carrier as a feature of the wireless network, thus web pages can be built dynamically and then abstracted from the processing logic and simply forwarded to the cell phone browser.

As users browse on their cell phone, their requests for information are passed from the view component on their phone to the controller, processed at the web server
and then sent back via the WAP Gateway. The purpose of the WAP Gateway is to serve as a controller component.

In their book “Professional WAP,” Arehart, N. Chidambaram, et al. explain how the WAP Gateway processes each request [1]. WAP based websites behave differently from normal websites. The web browser sends the request to the gateway, which allows the developer to put all the client side scripting on the gateway. A request sent from the web browser enters the gateway and is converted to the appropriate model format. The request is then sent to an Apache web server and processed. In this architecture, the web server is a model component. It receives the request from the gateway (the controller) and processes it, including accessing the database if appropriate. The database is also a model component under the architecture. The web server then generates web pages dynamically in response to the request, incorporating database elements as appropriate. The pages are served in the view required by the browser since different phone models have potentially different browsers running on them. Because the design is based on an MVC architecture, the number of trips on the network is reduced which is advantageous to cell phone users. This concept is explained in detail in the work “Applications Programming in Smalltalk-80(TM) by S. Burbeck, Ph.D. [6]. Efficiency in the architecture is the key due to the resource constraints of cell phones. The design also allows for transparency within the three components. Figure 2 shows the architecture of the General Reservation System.
Fig. 2. Architecture of general reservation system.

Legend:
WAP: Wireless Application Protocol
WSP: Wireless Session Protocol
WML: Wireless Markup Language
PHP: Hypertext Preprocessor
HTTP: Hypertext Transfer Protocol
Information flows through the system in the following manner:

1. User makes a request from the cell phone using the micro-browser.

2. The request is sent to the gateway using Wireless Session Protocol (WSP). WSP is the communication mechanism with the WAP Gateway.

3. WSP opens a session on the network. The user can send and receive single packet messages across the session. This information, found in the book “Dynamic WAP Application Development” by S. M. Foo, C. Hoover, and W.M. Lee, allowed me to understand that the WSP uses session mechanism to interact with the WAP Gateway [7].

4. The WSP closes the session when the message transfer is completed.

When the cell phone browser makes a request it is sent to the WAP Gateway which acts as a proxy server. The article “WAP Gateways” by Service Strategies Inc explains in detail the functionality of the WAP Gateway [8]. The WAP Gateway sends and receives information to the web server, so in this respect it also acts as a client to the web server. The user request includes “headers” that relay information to the gateway about the requesting client. A “User-Agent” header identifies the client to the gateway, while a “Via” header identifies the type of request the client is making. In our application the “Via” header will indicate that information is coming from a WAP Gateway.

When the information arrives at the gateway, it uses the headers to help it decode the WSP request. Some requests only require validation of information being entered on the client side. This can be easily done using Wireless Markup Language Scripts (WMLScripts) which are compiled and stored in the WAP Gateway.

If the request is for information that needs to be received from the web server, then the gateway will format a Hypertext Transfer Protocol (HTTP) request and send it to
the web server. For this application we are using an Apache web server. The Apache web server uses a special scripting language called PHP (which used to stand for Personal Home Page). PHP scripts are used within Apache to process the request and return information to the gateway. In his Article, “Building dynamic WML sites,” A. Baravalle explains how to use PHP functions for website development [9]. The communication from the Apache server to the database, when necessary, is performed using PHP as referred in “Research on a Distributed Architecture of Mobile GIS Based on WAP” article by W. Fangxiong & J. Zhiyong. This reference helped me in understanding how the communication between PHP and the database takes place [10].

At the gateway, the scripts are converted back to WML, compressed, and transmitted in binary form to the client. This compression enables the wireless communication to happen very quickly even for bandwidth constrained devices such as mobile phones. All communication with the client side is done using WMLScripts which are compiled at the WAP Gateway and sent to the mobile phone in binary format. Compiling WML Scripts is explained in detail in the “WAP Gateway” article, previously mentioned [8]. Again, this relieves the requirement on the mobile device client.

The application supports making flight reservations and car reservations for six (6) major USA Cities (San Francisco, LA, Chicago, Seattle, Dallas and New York). I selected major cities for ease of development and for ease of finding flight and car information, as well as to prove the concept. Once the basic application is developed it will be easy to add as many cities as we want through the administrative side of the application. The Administrative side will allow the owner of the website to add cities, create flights and manage the user information. Figure 3 presents the class diagram for
Fig. 3. Class diagram for general reservation system.

the General Reservation System. It shows the classes, attributes, and relationship between the classes.
The main classes for the reservation system are Flight, AirCRAFT, FlightReservation, Car, CarReservation, Customer, and DBManager. Customer, Location and Carbase are the super classes in the reservation system and other classes are inherited from these classes. Each of these classes has functions and attributes which are important for the specific classes. For example, the Customer class has attributes such as CustomerID, Phone, email etc. and has functions which are used to set and retrieve the customer information.

For the application backend, I am using a MySQL database to store all the information required for the application. This includes data entered by an Administrator to support the functionality for availability of flights, cars, cities, and so on, as well as the customer information and customer records such as name, address, credit card number and telephone number. Figure 4 shows the database model diagram.

The main tables in the database are FlightReservationID, Flight, Aircraft, Customer, CarReservation, and CarRegistration. Each of these tables has a Primary key Column and other columns. Tables like CarReservation have the CustomerID column as a foreign key which helps in tying the Car reservation to a specific customer. The Database design section explains the database tables in more detail. For e.g. it has the SQL statement to create the Carreservation table. Creservationid is the primary key column.

Software Tools

The primary development of this website was done using the Nokia WAP Toolkit 2.0 and the WAMP Server. WAMP Server stands for “Windows Apache MySQL
The website is built on Apache Web Server (also called Apache Tomcat). MySQL is the database used to store information. It is also the most popular freely available
database is suitable for building web applications. PHP is the scripting language to process the requested information in the Apache web server. PHP is a free web-programming tool, used widely due to its ease of use, open source, and availability. These tools are described further in the following sections.

WAP, WMLScript, WML, PHP and mySQL are used in the development of the website. Below is a detailed description of each. The WAMP implementation that I used in this project runs on Windows operating system. There is also a version available for Linux which is called LAMP for Linux, Apache, MySQL, and PHP.

Nokia WAP Toolkit

The Nokia WAP Toolkit is a tool for writing and compiling WMLScript files, which are then linked with the WML Pages for validation. Toolkit also provides the ability to write new scripts and find any errors in them. All the website front end development is done on the Nokia WAP Toolkit. The Nokia WAP Toolkit is free to use and can be downloaded from the Nokia forum. The advantages of using a Nokia WAP Toolkit are:

- It conforms with WAP standards as defined by the WAP Forum.
- It has Nokia phone Emulators.
- The toolkit has a WML Editor and WMLScript capability which allow us to organize our scripts in a central repository.

The toolkit includes a emulator with the ability to browse WML Pages and render them for client applications. It also includes a WAP Gateway for communication with the server side of the application. “Professional WAP” by C. Arehart, N.
Chidambaram et al. discusses the advantages of using the Nokia toolkit over other tools present in the market [1].

**WAMP Server**

The application uses WAMP Server 2.0 which includes the following versions of software:

- Apache 2.2.8
- PHP 5.2.6 + PECL
- SQLLiteManager
- MySQL 5.0.51b
- PHPMyAdmin

**Apache Web Server**

Apache Web Server is the best-suited free web server used in building the websites. It supports HTTP and is fully configurable. Apache is faster and a more stable than some other free web servers. It is compatible with PHP and MySQL Database and is the solution most frequently chosen by application developers looking for free web servers. “Develop WAP applications with Java servlets and JavaServer Pages,” by Q. H. Mahmoud discusses configuring my web server to run WML pages [11].

**PHP**

PHP is the Hypertext Processor that is used for the server side scripting. PHP allowed me to create dynamic websites depending on the user’s demands. PHP code integrates easily with the WML decks and allowed me to create custom content from various sources. This topic is discussed in “Building Dynamic WML Sites” [9]. It is also
compatible with Apache Web Server and MySQL. PHP code is written and executed on
the server side. PHP can be downloaded for free software on its website [5].

**MySQL**

MySQL is the most popular free Database Management System. MySQL supports SQL, which is used to define and manipulate data in the MySQL database. MySQL is readily available and is compatible with PHP and Apace Web Server [5]. Figure 4 explains the Database Model Diagram. It explains the database tables and their dependencies.

**WAP**

Wireless Application Protocol, or WAP, provides a framework for quick and easy application building using wireless technology. The language of WAP is known as Wireless Markup Language, or WML, which is a subset of XML. It is a “tagged” language, as well as a global standard, making it easy to share information with different applications. Scripts built using WML are called WMLScripts. WAP is designed for lightweight environments on portable devices such as mobile phones and Smart Phones and supports rapid transfer of data on small footprint devices which may lack the power, memory and bandwidth of large computers. A discussion of how to build dynamic websites is included in the book “Dynamic WAP Application Development,” by S. M. Foo, C. Hoover, and W. M. Lee [7].

**WML**

WML stands for the Wireless Markup Language. WML is an XML-based language and is supported by most phones worldwide. WML is used to write the decks and the cards. Cards are web pages for wireless applications. Several cards can be
combined to form a deck, which allows us to display the cards faster. A deck consists of a heading, a card template and several cards. WMLScript is a lightweight scripting language based on JavaScript. The scripts themselves are converted to byte code so WML pages only contain references to URLs. It is this feature of WMLScripts that makes them ideally suited to lightweight portable device applications. The WMLScript is used to do all the client-side scripting. All the processing is done on the client side (WAP Devices) and provides the user with a much faster experience. WMLScript is used for user input validation and allows client side data manipulation. Syntax for this is covered in Rischpatar’s “Wireless Web Development with PHP and WAP” [5]. WMLScript has a standard set of library functions which reside on the browser side making it easier for us to do all the validity checks.

The following is a list of validity checks that were performed:

- User name should be a cell phone number.
- Password should be at least 8 characters.
- Password should be at least 8 characters.
- Email should be in an email format.
- Address should be greater than 20 characters.
- Date should be in a correct date format.
- Start date should be in the future.
- Start date is prior to end date.
- Credit card number should be 15 or 16 characters long.
- Expiration date of credit card should be greater than current date.
WAP Gateway

The WAP Gateway is the broker that interacts both with the client data coming from the mobile phone side and the web server and database on the backend. The WAP Gateway comprises several components including a Wireless Transport Layer Security (WTLS). WTLS is a security layer which is responsible for encryption services. The WTLS is not implemented for this website. I will implement security, an enhanced feature referred to in Chapter IV under Future Enhancements. The transport layer of the WAP Gateway uses WAP datagram protocol (WDP) to send and receive data from connecting applications. The WDP transport layer is supported by the WAP transaction protocol (WTP) layer, adding reliability to the transport layer.

The WAP Gateway works by creating sessions using the WAP session protocol (WSP). These lightweight sessions exist for the duration of a transaction from the client. The initial connection from a client to the gateway is via Hypertext Transfer Protocol (HTTP). The WAP Gateway uses WMLScript to process the request and to decide if the request is to be executed locally on the gateway or to send it to the server over HTTP. Once the gateway receives the requested information back from the server, it converts the returned data into byte code for fast and easy transport back to the client. The mobile device translates the byte code into a format which is compatible to the specific device and displays the contents on the wireless browser. The article on WAP Gateways covers how the WAP Gateway processes each request [10].
CHAPTER III

METHODOLOGY

In this section, I describe the approach for the project and then cover the software application design, including screen shots and database tables.

Project Approach

The Development of the website was done in four phases: Design, Setting up the Development Environment, Code Development, and Testing. I started by defining the scope and by creating the business requirement document. I then proceeded to describing, ranking and scheduling the Use Cases. The Functional Specification document was written and reviewed.

After the requirements and initial use cases were developed I was able to begin setting up the development environment and begin coding. The software development process consisted of developing three components, the front end, the backend and the database. Finally, when coding was complete, the application was tested.

Setting up the Development Environment

The required tools were downloaded and installed on the computer. Pilot tests were performed to make sure the installation was complete. The following tools used for development were installed:
Nokia WAP Toolkit
WAMP Server

The development environment was carried out in three phases which are outlined below.

Phase 1

In the first phase, I installed the Nokia WAP Toolkit and the WAMP Server. I also created the WML pages for the website. All the WMLScripts, to do the client side scripting, were written in this phase.

Phase 2

In the second phase, I created the database and the database tables for the website. The data for the database tables were generated in this phase and populated in the tables.

Phase 3

The third phase involved the development of the PHP code. I created the PHP scripts to interact with WML pages and to access the MySQL Database. All the scripts to generate the dynamic content were written in this phase. Since the project consists of two parts, admin and user therefore the site is having two designs. For the WAP Site, the user will use a mobile device.

I tested the complete development environment using WINWAP WAP Browser and Nokia Series 40 5th Edition SDK feature pack 1, which is available as a free trial on the Internet.
Implementation

The implementation phase has three phases. The first is the front end; the second is the back end, and lastly the database. The three phases are described in the following sections.

Phase 1: Front End Implementation of the Web Site

The purpose of the website is to book flights and car rentals from handheld devices such as mobile phones. In order to accomplish this we need to enable the mobile devices with WML and WMLScript frameworks. The overall site is divided into two parts and each part is comprised of different modules. Each module is implemented using WML and PHP files. The first thing to do was define the role of a user and an administrator. Clearly, the user is the cell phone user who wishes to reserve a flight or a car. The administrator function is necessary to set up the application. Cell phones have a small screen and it is really hard to do all the administration of the website on them. The administrator function allows the administrator to keep the website up-to-date and operational. Following are the capabilities of the user and the administrator:

- Administrator. Add new information for:
  - New cities for Flight Reservations
  - New Flights
  - Flight Details
  - Flight Locations
  - Add Status to Flights
  - Flight Fare Types
- View Flight Reservations
- New cities for Car Reservations
- Add New Car Details
- Car Base Stations
- Car Status
- Car Fare Types
- View Car Reservations
- Analyze and keep track of user behavior
- Run promotions on the website

☐ User

- Login
- Registration
- View Flights
- View Cars
- Search Flights
- Search Cars
- Cancel Flights
- Cancel Cars
- My Flight Reservations
- My Car Reservations
- Manage Profile
- Logout
Each of the screens allows user input. I have used WMLScript to perform client side validation of user inputs at the WAP Browser.

On the registration Page, if a user fills out invalid information the WMLScript checks the details and gives the appropriate error message to the user. For example to check for a valid Email address I have used the following Script:

```wml
extern function register_form_Email_validate()
{
    var register_form_email_address = String.trim(WMLBrowser.getVar("register_email"));
    if (""==register_form_email_address){
        WMLBrowser.setVar("errorMsg", "The Email field cannot be empty.");
        WMLBrowser.refresh();
        return;
    }
    if (""==String.elementAt(register_form_email_address, 0, "){
        WMLBrowser.setVar("errorMsg", "The Email field cannot start with @");
        WMLBrowser.refresh();
        return;
    }
    if (""==String.elementAt(register_form_email_address, 1, "){
        WMLBrowser.setVar("errorMsg", "The Email should have at least 1 chars");
        WMLBrowser.refresh();
        return;
    }
    if (String.elements(register_form_email_address, ") != 2){
        WMLBrowser.setVar("errorMsg", "The Email is not valid");
        WMLBrowser.refresh();
        return;
    }
}
```

The remainder of this section presents screen shots of the working Project with a brief description of how each page is used. Rather than capture every possible screen here, only the primary interface screens are shown. Appendix A captures an additional screen shot for a complete picture of how the website looks.
- **Home Page**: Figure 5 is the Home Page of the website. Here the user has the ability to log to the system or register as a new user. If user has already logged in, he/she can go straight to searching for a flight or a car.

![Fig. 5. Website home page.](image)

- **Registration Page**: Figure 6 shows the customer registration form. The user has to enter the fields with “*”. The Phone number will be used as the primary Sign Up for the user. The Address will be stored for the credit card verification purposes. For security, I have kept the password to be minimum 8 characters.
• **Login Page**: Figure 7 presents where the user inputs his or her phone number as username and password to log into the system. A user can only access the booking of the car/flight on the website after he/she has successfully logged in to the system.

• **User Home Page**: Figure 8 shows the user home page. The user arrives at this page after successfully logging in to the system. From here user can perform searches for the car and flights and can edit their details. From the User Home Page, the user should be able to reserve cars and flights. Edit Profile is also added so the user can edit his/hers stored information.
Fig. 7. User login screens.

- **Flight Reservation System Page**: From the home page, the user comes to the page indicated in Figure 9 and uses various options related to the flight reservation. Here the user has the ability to do flight searches, view reserved flights and cancel any reserved flights. The user should be able to go home from this page.

- **Flight Search Page**: Figure 10 shows the Search Flight capability. The user fills in the details of the source and destination cities, flight date, number of travelers and option to book both to and from journeys. The user is able to book one-way ticket or a round trip. A message is displayed for the user to leave the return date blank for one-way flight.
Fig. 8. User home page.

Fig. 9. Flight reservation page.
• **Search Results Page**: Figure 11 shows how the user views the various flights that are returned after the search details are entered. The Price link is used by the user to book a flight. At the top of the search page the information regarding the search is displayed. The first results page will display the Departing flight and the second results page will display the returning flight.

• **Flight Booking Page**: Figure 12 shows how the user selects from the payment options available. Currently the payment type is only credit. The WMLScript will do all the client side scripting which will involve checking the credit card information and the year field. The total price for the flights is also displayed for the user.
Fig. 11. Flight search results.

Fig. 12. User payment options.
- **Booking Confirmation Page**: Figure 13 shows the user confirmation of the reservation. The page displays in detail the itinerary for the user and also the credit card used. A confirmation message is displayed to let the user know that the flights have been booked.

![Reservation Confirmation](image)

Fig. 13. Reservation confirmation.

- **View/Cancel Flight Reservation Page**: Figure 14 shows the list of the flights reserved by the user. For each user reservation, there is a Cancel link from where the user is able to cancel his/hers reservation.
Fig. 14. Flights page.

- **Flight Cancelled Page**: Figure 15 shows the confirmation page for a cancellation of a particular flight by a user. The reservation number for the flight is displayed with the Cancellation confirmation message.

- **Car Reservation System Page**: Figure 16 shows the Car Reservation Page. This page is opened by the user after login. From this page, the user has the ability to search cars, view or cancel booked cars and go back to the home page.

- **Car Search Page**: Figure 17 shows the Search Car Page. Here a user fills out the pick up location, pick up date and drop off date that he/she requires so that the system
Fig. 15. Flight cancellation page.

Fig. 16. Car reservation page.
can search for a car to fit his/her needs. The user should be able to pick the pickup location from the available locations. The date to be searched is in the mm/dd/yyyy format.

- **Car Results Page**: Using the screen in Figure 18, a user can view the list of the cars matching the details he/ she entered on the car search page. User can click on the price link to book a car from here.

- **Car Book Page**: The screen in Figure 19 shows how the user selects one of the various payment methods available. Currently it is only Credit. The user will be able
Fig. 18. Car results page.

Fig. 19. Car book page.
to enter his/her credit card information on this page. The number of days is calculated automatically by the system using the pick up date and drop off date.

- **Car Booking Confirmation Page**: Figure 20 is the Car Booking Confirmation Page. The user arrives at this page by entering all the required payment details i.e. credit card information etc. on the payment page. The Car Booked message will be displayed to let the user know his/her reservation was successful.

Fig. 20. Car booking confirmation page.
- **Car View/Cancel Page**: Figure 21 shows the Car View/Cancel Page. Here user can view or cancel cars he/she has booked so far. The Cancel button will be present to let the user cancel his/her reservations. If the user has no history it will display a message saying “No Cars Reserved.”

![Car View/Cancel Page](image)

Fig. 21. Car view/cancel page.

- **Car Booking Cancelled Page**: Figure 22 shows the confirmation of a Car Cancellation. The Reservation number will be displayed, as well as a message confirming the cancellation.
- **Edit Profile Page**: The user arrives at the page in Figure 23 from the user home page after clicking on the Edit Profile option. The user can update his/her profile details, except for the phone number.

**Phase 2: Back End (Administrator Section)**

The Administrator section of the website is created using PHP and MySQL Database. The server engine depends on the integration of the WAMP Server with the
WAP Nokia Toolkit. The WAMP Server processes PHP and MySQL. Requests made by
the WAP Browser (WinWAP or Nokia Toolkit) are sent to the WAP which converts the
requests into HTTP. The HTTP requests are then sent to the WAMP Server. Integration
between the WAMP and WAP BROWSER is done by the WAP GATEWAY which is
part of the NOKIA Toolkit.

The page flow is indicated in the following screen shots and descriptions of
the pages. Figure 24 shows the basic interface. The Administrator’s part is divided into
five major sections: Home Page, Car Details, Flight Details, About Me, and Contact Me.
Fig. 24. Administrator interface.

The Home page, About Me page, and the Contact Me page are all similar in design. The Car Details and Flight Details page have additional functionality which is further divided in the following way:

- **Car Details**: This Section is used for entering the car details available for rent on site. Figure 25 shows how you add car details such as Color, Model, Make, Price Per day etc. into the system so that any user can view them on their Car Search Page. It provides the following capabilities:
  - Add New Car Details
  - Edit Exiting Car Details
  - Delete Existing Car Details

- **Car Base Station Details**: This section, illustrated in Figure 26, is used to add the base stations (e.g., Cities) from where cars will start. Any user can book a car from the available base stations. Car Details are linked to the base stations using a referential integrity. It provides the following capabilities:
MOBILE RESERVATION SYSTEM

- Home | Car Details | Flight Details | About Me | Contact Me |
- Car Details | Base Station | Fare Details | Car Reservation | Car Status |

### Car Base Station Details

<table>
<thead>
<tr>
<th>CAR DETAILS</th>
<th>CAR ID</th>
<th>COLOR</th>
<th>MAKE</th>
<th>MODEL</th>
<th>CAPACITY</th>
<th>STATUS TYPE</th>
<th>BASE NAME</th>
<th>PRICE PER DAY</th>
<th>MODIFY</th>
<th>DELETE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Silver</td>
<td>Honda</td>
<td>Civic</td>
<td>4</td>
<td>available</td>
<td>New York</td>
<td>60</td>
<td>EDIT</td>
<td>DELETE</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Red</td>
<td>Toyota</td>
<td>Prius</td>
<td>5</td>
<td>available</td>
<td>LA</td>
<td>100</td>
<td>EDIT</td>
<td>DELETE</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>White</td>
<td>Chevy</td>
<td>Aveo</td>
<td>5</td>
<td>available</td>
<td>San Francisco</td>
<td>75</td>
<td>EDIT</td>
<td>DELETE</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Black</td>
<td>Nissan</td>
<td>Altima</td>
<td>5</td>
<td>available</td>
<td>Dallas</td>
<td>120</td>
<td>EDIT</td>
<td>DELETE</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Red</td>
<td>Chevy</td>
<td>Cobalt</td>
<td>4</td>
<td>available</td>
<td>Chicago</td>
<td>90</td>
<td>EDIT</td>
<td>DELETE</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>White</td>
<td>Honda</td>
<td>CRV</td>
<td>7</td>
<td>available</td>
<td>Seattle</td>
<td>150</td>
<td>EDIT</td>
<td>DELETE</td>
<td></td>
</tr>
</tbody>
</table>

Click Here to Add New Car Details

*Click Links to Edit/Delete

#### Fig. 25. Add car details.

- Add New Car Base Station
- Edit Exiting Car Base Station
- Delete Existing Car Base Station

### Car Base Station Details

<table>
<thead>
<tr>
<th>AVAILABLE BASE NAME DETAILS</th>
<th>BASE ID</th>
<th>BASE NAME</th>
<th>BASE MODIFY</th>
<th>BASE DELETE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>San Francisco</td>
<td>EDIT</td>
<td>DELETE</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>New York</td>
<td>EDIT</td>
<td>DELETE</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Dallas</td>
<td>EDIT</td>
<td>DELETE</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>LA</td>
<td>EDIT</td>
<td>DELETE</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Chicago</td>
<td>EDIT</td>
<td>DELETE</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Seattle</td>
<td>EDIT</td>
<td>DELETE</td>
<td></td>
</tr>
</tbody>
</table>

Click Here to Add New

*Click Links to Edit/Delete

#### Fig. 26. Car base station details.
• **Car Fare Details:** In this section, illustrated in Figure 27, I have provided the ability to add fare types acceptable for car rental. Currently all pricing and payments are based on using a credit card. Additional fare types can be added later with more fare payment methods. This table is associated with the car details section using referential integrity. This section also has following abilities:

  - Add New Car Fare Type
  - Edit Exiting Car Fare Type
  - Delete Existing Car Fare Type

![Car Fare Details Table](image)

Fig. 27. Car fare details.

• **Car Status Details:** This section, illustrated in Figure 28, is used to check the status of the car. When a car is booked by a customer for rental, then the status changes to “reserved,” otherwise by default it is “available.” There are different status types for cars such as “not working,” “available,” “reserved” etc. This table is associated with the car details section using referential integrity. This section also has the ability to:
Fig. 28. Car status details.

- Add New Car Status
- Edit Exiting Car Status
- Delete Existing Car Status

- **Car Reservations Details**: This section, illustrated in Figure 29, is only used to view the different bookings made by a user. By using this section, the administrator can view the reservation patterns and can generate various reports. It also gives accurate predictions for how the business is doing which allows the administrator to make appropriate decisions related to future demands and availability of the cars for the users.

- **Flight Details**: This section, illustrated in Figure 30, is used for entering the flight details available for booking on site. It provides the following capabilities:
  - Add New Flight Details
  - Edit Exiting Flight Details
  - Delete Existing Flight Details
  - View Flight Information
MOBILE RESERVATION SYSTEM

<table>
<thead>
<tr>
<th>Home</th>
<th>Car Details</th>
<th>Flight Details</th>
<th>About Me</th>
<th>Contact Me</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Flight Details</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Car Reservation</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Car Status</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**VIEW YOUR CAR RESERVATIONS**

<table>
<thead>
<tr>
<th>REGISTRATION ID</th>
<th>DEPARTURE DATE</th>
<th>ARRIVAL DATE</th>
<th>RESERVATION DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2009-02-20</td>
<td>2009-02-25</td>
<td>2009-02-17</td>
</tr>
<tr>
<td>6</td>
<td>2009-03-02</td>
<td>2009-03-05</td>
<td>2009-03-01</td>
</tr>
<tr>
<td>8</td>
<td>2009-03-15</td>
<td>2009-03-20</td>
<td>2009-03-10</td>
</tr>
<tr>
<td>9</td>
<td>2009-03-15</td>
<td>2009-03-20</td>
<td>2009-03-10</td>
</tr>
<tr>
<td>10</td>
<td>2009-02-20</td>
<td>2009-03-20</td>
<td>2009-03-10</td>
</tr>
<tr>
<td>11</td>
<td>2009-03-20</td>
<td>2009-03-20</td>
<td>2009-03-10</td>
</tr>
<tr>
<td>12</td>
<td>2009-03-20</td>
<td>2009-03-20</td>
<td>2009-03-10</td>
</tr>
<tr>
<td>13</td>
<td>2009-03-20</td>
<td>2009-03-30</td>
<td>2009-03-10</td>
</tr>
<tr>
<td>14</td>
<td>2009-03-20</td>
<td>2009-03-30</td>
<td>2009-03-10</td>
</tr>
</tbody>
</table>

Fig. 29. Car reservations details.

MOBILE RESERVATION SYSTEM

<table>
<thead>
<tr>
<th>Home</th>
<th>Car Details</th>
<th>Flight Details</th>
<th>About Me</th>
<th>Contact Me</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Flight Details</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Car Reservation</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Car Status</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**FLIGHT DETAILS**

<table>
<thead>
<tr>
<th>FLIGHT ID</th>
<th>DEPARTURE DATE</th>
<th>ARRIVAL DATE</th>
<th>AIRCRAFT REGISTRATION</th>
<th>STATUS TYPE</th>
<th>FROM</th>
<th>TO</th>
<th>SEATS</th>
<th>TAKE OFF TIME</th>
<th>LANDING TIME</th>
<th>RATE OF TICKET</th>
<th>MODIFY</th>
<th>DELETE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2009-03-15</td>
<td>2009-03-15</td>
<td>1001</td>
<td>READY</td>
<td>LA</td>
<td>Chicago</td>
<td>200</td>
<td>6 AM</td>
<td>10 PM</td>
<td>100</td>
<td>EDIT</td>
<td>DELETE</td>
</tr>
<tr>
<td>2</td>
<td>2009-03-20</td>
<td>2009-03-20</td>
<td>10011</td>
<td>READY</td>
<td>Chicago</td>
<td>LA</td>
<td>250</td>
<td>9 AM</td>
<td>2 PM</td>
<td>150</td>
<td>EDIT</td>
<td>DELETE</td>
</tr>
<tr>
<td>7</td>
<td>2009-05-01</td>
<td>2009-05-01</td>
<td>126</td>
<td>READY</td>
<td>San Francisco</td>
<td>New York</td>
<td>75</td>
<td>10 AM</td>
<td>6 PM</td>
<td>150</td>
<td>EDIT</td>
<td>DELETE</td>
</tr>
<tr>
<td>8</td>
<td>2009-05-11</td>
<td>2009-05-11</td>
<td>767</td>
<td>READY</td>
<td>New York</td>
<td>San Francisco</td>
<td>75</td>
<td>6 PM</td>
<td>11 PM</td>
<td>200</td>
<td>EDIT</td>
<td>DELETE</td>
</tr>
<tr>
<td>9</td>
<td>2009-06-01</td>
<td>2009-06-01</td>
<td>988</td>
<td>READY</td>
<td>San Francisco</td>
<td>LA</td>
<td>50</td>
<td>8 AM</td>
<td>9 AM</td>
<td>60</td>
<td>EDIT</td>
<td>DELETE</td>
</tr>
</tbody>
</table>

Click [here](#) to Add New Flight Details

*Click Links to Edit/Delete*

Fig. 30. Flight details.
This page adds flight details (aircraft registration number, date of flight takeoff, date of flight arrival, time of flight takeoff, time of flight arrival, ticket price and number of seats, etc.) in the system so that any user can view them on their flight search page.

- **Flight Location Details**: This section, illustrated in Figure 31, is used to add the location of the flights (e.g., Cities) from where flights will take off. Any user can book a flight from any available flight locations. Flight Details are linked to the Flight Locations using referential integrity based on flight key of the FLIGHTLOCATION table. It provides the following capabilities:
  - Add New Flight Location
  - Edit Existing Flight Location
  - Delete Existing Flight Location

![Fig. 31. Flight location details.](image)

- **Flight Fare Details**: In this section, illustrated in Figure 32, I have provided the ability to add fare types acceptable for flight booking. Currently the only payment method supported is a credit card. Later more fare types could be added. This table is
Fig. 32. Flight fare details.

associated with the car details section using referential integrity. This section also has the following capabilities:

- Add New Flight Fare Type
- Edit Exiting Flight Fare Type
- Delete Existing Flight Fare Type

- **Flight Status Details**: This section, illustrated in Figure 33, is used to check the status of a flight. Flight Status types are “delayed,” “on time,” “cancelled” etc. Status is indicated when a flight is delayed or cancelled due to any of the various reasons. This table is associated with the flight details section using referential integrity. Other capabilities included in the Flight Status section are as follows:

  - Add New Flight Status
  - Edit Exiting Flight Status
  - Delete Existing Flight Status

- **Flight Reservations Details**: This section, illustrated in Figure 34, is used to view the different bookings being done by a user. By using this section, the administrator can view reservation patterns and generate various reports. This section also provides
accurate predictions for how the business is doing and the administrator can make appropriate decisions with respect to future demands and the availability of flights for users.

Fig. 34. Flight reservations details.
Phase 3: Database Design

This section presents the actual table structures that are used in coding the project.

- **CAR Table**: The CAR table maintains car detail data that is made available to users for making reservations. CARREGISTRATIONID is the Primary Key of the table and is used to associate various car reservations made by the customers. The table is linked with CARBASE, CARFARE, CARESTATUS tables using a Foreign Key relationship.

```sql
CREATE TABLE IF NOT EXISTS `car` (
  `carregistrationid` bigint(10) unsigned NOT NULL auto_increment,
  `color` varchar(50) NOT NULL default '',
  `make` varchar(50) NOT NULL default '',
  `model` varchar(50) NOT NULL default '',
  `capacity` int(2) unsigned NOT NULL default '0',
  `statusid` bigint(3) unsigned NOT NULL default '0',
  `baseid` bigint(3) unsigned NOT NULL default '0',
  `priceperday` bigint(10) unsigned NOT NULL,
  PRIMARY KEY  (`carregistrationid`)
) ENGINE=MyISAM  DEFAULT CHARSET=latin1 AUTO_INCREMENT=6 ;
```

- **CARBASE Table**: The CARBASE table is used to keep the details of the various cities where a car is available for rental. This table is associated with the CAR Table and BASEID column of CAR Table and it works as the Foreign Key for CAR Table. BASEID is the Primary Key Column of the CARBASE Table.

```sql
CREATE TABLE IF NOT EXISTS `carbase` (
  `baseid` bigint(3) unsigned NOT NULL auto_increment,
  `basename` varchar(30) NOT NULL default '',
  PRIMARY KEY  (`baseid`)
) ENGINE=MyISAM  DEFAULT CHARSET=latin1 AUTO_INCREMENT=9 ;
```

- **CARFARE Table**: The CARFARE Table is used to maintain the details of the various fare types i.e. payment type options we can offer to the users for a car rental.
This table is associated with the CAR Table and FAREID column of it works as the Foreign Key for CAR Table. FAREID is the Primary Key Column of the CARFARE Table.

```
CREATE TABLE IF NOT EXISTS `carfare` (
   `fareid` bigint(2) NOT NULL auto_increment,
   `faretype` varchar(50) NOT NULL default ",",
   PRIMARY KEY (`fareid`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 AUTO_INCREMENT=2 ;
```

- **CARSTATUS Table**: The CARSTATUS Table is used to keep the details of the various STATUSES for a car, i.e. a car can be reserved/available/not available etc. This table is associated with the CAR Table and STATUSID column. It works as the Foreign Key for CAR Table. STATUSID is the Primary Key Column of the CARSTATUS Table.

```
CREATE TABLE IF NOT EXISTS `carstatus` (
   `statusid` bigint(3) unsigned NOT NULL auto_increment,
   `statustype` varchar(30) NOT NULL default ",",
   PRIMARY KEY (`statusid`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 AUTO_INCREMENT=14 ;
```

- **CARRESERVATION Table**: The CARRESERVATION Table maintains the reservation being entered by the users using their WAP Browser. An Administrator can view these details and make various strategies for future use. Various reports can be generated using this Table for improving processes within the company.

```
CREATE TABLE IF NOT EXISTS `carreservation` (
   `creservationid` bigint(10) unsigned NOT NULL auto_increment,
   `customerid` bigint(10) unsigned NOT NULL default '0',
   `carregistrationid` bigint(10) unsigned NOT NULL default '0',
   `reservationdate` date NOT NULL default '0000-00-00',
   `departuredate` date NOT NULL default '0000-00-00',
   `arrivaldate` date NOT NULL default '0000-00-00',
   `fareid` bigint(3) unsigned NOT NULL default '0',
   `iscancelled` varchar(1) NOT NULL default 'n',
```
- **CUSTOMER Table**: The CUSTOMER table is where customer data are stored for a customer who is registering using the WAP Browser. This table is also used to create the user session whenever a user logs in for verification of the user data. I have used the user phone number as the username for login and CUSTOMERID as the primary key.

```sql
CREATE TABLE IF NOT EXISTS `customer` (
    `customerid` bigint(10) unsigned NOT NULL auto_increment,
    `name` varchar(100) NOT NULL default '',
    `address` varchar(200) NOT NULL default '',
    `phonenumber` varchar(12) NOT NULL default '',
    `email` varchar(100) NOT NULL default '',
    `password` varchar(10) NOT NULL,
    PRIMARY KEY (`customerid`)
) ENGINE=MyISAM  DEFAULT CHARSET=latin1 AUTO_INCREMENT=14 ;
```

- **FLIGHT Table**: FLIGHT Table is where details of the flights are stored for booking. It contains all the details related to the available flights online for booking such as DEPARTURE DATE, ARRIVAL DATE, TAKEOFFTIME, LANDING TIME, Price of tickets, Number of the Seats in the flight etc. This table is linked to various tables using referential integrity constraints i.e. through FLIGHTLOCATION, FLIGHTFARE and FLIGHTSTATUS etc.

```sql
CREATE TABLE IF NOT EXISTS `flight` (
    `flightnumber` bigint(10) unsigned NOT NULL auto_increment,
    `departuredate` date NOT NULL default '0000-00-00',
    `arrivaldate` date NOT NULL default '0000-00-00',
    `aircraftreg` bigint(3) NOT NULL default '0',
    `statusid` bigint(2) unsigned NOT NULL default '0',
    `deplocationid` bigint(3) NOT NULL default '0',
    `arrlocationid` bigint(3) NOT NULL default '0',
    `seats` int(5) unsigned NOT NULL,
    `takeofftime` varchar(10) NOT NULL,
```
CREATE TABLE IF NOT EXISTS `flightfare` (  
  `fareid` bigint(2) NOT NULL auto_increment,  
  `faretype` varchar(50) NOT NULL default '',  
  PRIMARY KEY  (`fareid`)  
) ENGINE=MyISAM DEFAULT CHARSET=latin1 AUTO_INCREMENT=2 ;

- **FLIGHTFARE Table**: This table is used to keep the details of the various fare types i.e. payment type options offered to users for a flight booking. This table is associated with the FLIGHT Table and FAREID column of it works as the Foreign Key for FLIGHT Table. FAREID is the Primary Key Column of the FLIGHTFARE Table.

CREATE TABLE IF NOT EXISTS `flightlocation` (  
  `locationid` bigint(2) unsigned NOT NULL auto_increment,  
  `locationname` varchar(50) NOT NULL default '',  
  PRIMARY KEY  (`locationid`)  
) ENGINE=MyISAM DEFAULT CHARSET=latin1 AUTO_INCREMENT=7 ;

- **FLIGHTLOCATION Table**: The FLIGHTLOCATION table is used to keep the details of the various cities where a flight is available. This table is associated with the FLIGHT Table and LOCATIONID column of the FLIGHT Table. It works as the Foreign Key for FLIGHT Table. LOCATIONID is the Primary Key Column of the FLIGHTLOCATION Table.

CREATE TABLE IF NOT EXISTS `flightstatus` (  
  `statusid` bigint NOT NULL auto_increment,  
  `statusname` varchar(50) NOT NULL default '',  
  PRIMARY KEY  (`statusid`)  
) ENGINE=MyISAM DEFAULT CHARSET=latin1 AUTO_INCREMENT=7 ;

- **FLIGHTSTATUS Table**: This table is used to maintain the details of the various STATUSES for a Flight i.e. a flight can be delayed/cancelled/available etc. This table is associated with the FLIGHT Table and STATUSID column of the FLIGHT Table. It works as the Foreign Key for FLIGHT Table. STATUSID is the Primary Key Column of the FLIGHTSTATUS Table.
CREATE TABLE IF NOT EXISTS `flightstatus` (  `statusid` bigint(2) unsigned NOT NULL auto_increment,  `statustype` varchar(50) NOT NULL default '',  PRIMARY KEY (`statusid`) ) ENGINE=MyISAM DEFAULT CHARSET=latin1 AUTO_INCREMENT=3 ;

- **FLIGHTRESERVATION Table**: The FLIGHTRESERVATION table maintains the reservation being made by users using their WAP Browser. The Administrator can view these details and develop various strategies for future use. Various reports can be generated using this Table for improving company processes. This table is also used to check how many seats are available on any particular flight by summing the booked seats in that flight number.

CREATE TABLE IF NOT EXISTS `flightreservation` (  `freservationid` bigint(10) unsigned NOT NULL auto_increment,  `customerid` bigint(10) NOT NULL default '0',  `flightnumber` bigint(5) NOT NULL default '0',  `reservationdate` date NOT NULL default '0000-00-00',  `seats` bigint(2) NOT NULL default '0',  `fareid` bigint(3) NOT NULL default '0',  `iscancelled` varchar(100) NOT NULL default 'n',  PRIMARY KEY (`freservationid`) ) ENGINE=MyISAM DEFAULT CHARSET=latin1 AUTO_INCREMENT=28 ;

- **PAYMENT Table**: The PAYMENT table maintains customer credit card information. Also it keeps track of the number of seats booked, which rental car has been taken and details of the Total payment made by the customer.

CREATE TABLE IF NOT EXISTS `payment` (  `serialno` bigint(10) unsigned NOT NULL auto_increment,  `customerid` bigint(10) unsigned NOT NULL,  `f_c_no` bigint(10) unsigned NOT NULL,  `firstname` varchar(50) NOT NULL,  `lastname` varchar(50) NOT NULL,  `cardtype` varchar(20) NOT NULL,  `cardnumber` bigint(15) NOT NULL,  `expirydate` varchar(10) NOT NULL,  `vehicletype` varchar(10) NOT NULL,  PRIMARY KEY (`serialno`) ) ENGINE=MyISAM DEFAULT CHARSET=latin1 AUTO_INCREMENT=28 ;
Testing

Testing was useful in identifying the critical bugs during the various phases of the development cycle and isolating them. Testing began in the design phase by creating the Quality Assurance Test plan. The test plan covered:

- Brief information about the website.
- Schedule of testing.
- Types of testing used to test the website.
- Features to be tested during each test phase.
- Test cases to be executed during each test phase.
- Entry criteria for each phase.
- Exit criteria for each phase.
- Sign off criteria.

Testing was continued throughout the development process until the maintenance phase. The testing used for this application included unit testing, integration testing and user acceptance testing. Each testing phase was used to test various features of the website. The sign off criteria for the testing were as follows:

- Correct navigation of all links.
- WML Pages displayed in correct format.
- Verify client side scripting.
• Able to search, reserve, view and cancel flights.

• Able to search, reserve, view and cancel cars.

Unit Testing

Unit Testing was done on each module of the source code. Unit testing covered testing of the functions and allowed me to expose design flaws early in the development cycle. Unit testing was performed to

• check the syntax of the code.

• make sure features are implemented correctly.

Unit testing was done on each function to confirm that they are working properly. Each function was tested for

• boundary conditions (To and From dates for Flight reservation).

• expected conditions (Credit card date later than current date).

• abnormal conditions (Leaving names blank for credit card information).

• valid format (Email address while creating your profile).

Unit testing was done by enabling the debugger mode and message boxes. WMLScripts were tested by calling the dialogs.alert function. The dialogs.alert function allowed me to test the variable values passed to the WMLScript by displaying it on the screen.

Integration Testing

Integration testing is an extension of the Unit testing and is part of Black Box Testing. The flights reservation module was integrated with the car reservation module and tested together. All the links and options for both modules were tested to make sure
the appropriate component was being called and executed. The integration testing was used to test the behavior of functions when combined together. The entry criteria for the integration testing is that all the individual functions are thoroughly tested and working properly. The Integration testing was carried out by enabling the debugging mode.

**User Acceptance Testing**

User Acceptance Testing (UAT) was performed to check if the application is stable and does not crash under any circumstances. It was also done on a fully developed website and covered verification of the links, functionality, and the navigation of the website in an environment closer to real world.

The user was able to test the website's functionality as a whole. User Acceptance Testing (UAT) was the final milestone on the testing side. It allowed me to get the user’s perspective on the website's functionality and to remove any false assumptions I had made.
CHAPTE IV

SUMMARY

Future Enhancements

In the future, this project could include the following enhancements to improve the overall experience and added value to the final product.

Hotel Reservation System

Adding a hotel reservation system will allow the user to reserve his or her whole trip by accessing the central website. This additional option will make trip planning convenient and time efficient.

Customer History

Maintaining history for each customer will allow users to save their favorite locations. This will also permit the website company to use a tailored marketing approach for its customers based on their booking and travel history.

Credit Card Security Using Anonymization

User information is very critical to the website. Information passed over the internet is always at risk of being hacked. The website will provide security for credit card information sent over the web using encryption and network hardening.
Sending Reservation Reminders via Text Message

The best websites are the ones that take care of their customers’ every need. The website will send the customers a text message as a reminder of their reservation 4 hours prior to the flight or car pickup.

Conclusion

Choosing the “General Reservation System using Wireless Application Protocol (WAP)” as the subject of my project was strongly driven by the fact that I myself am an example of the wireless generation and its development. I now use my phone mostly to access the internet to read the news, get concert tickets and also to make and receive phone calls. A phone is no longer just a phone. It is a tool offering more and more interesting features to end users such as myself. Building wireless applications seems to have just scratched the surface of what to expect in the future. Creating the wireless reservation website by with WAP technology is an interesting idea that expands on what is already being offered today by companies. The application fits well with the needs of a niche market of business and leisure travelers.

The objectives for the project were outlined above. The project successfully met the criteria for searching by date and location all available flights or available cars in the cities that were targeted. The search data available in the finished system is dependent upon what an administrator can enter, but the concept is extensible and additional cities can be added over time. It is possible to reserve, confirm, cancel and view a reservation for a car or a flight, in accordance with the objectives. Another objective was the ability to provide account management options such as storing credit card information. This
The objective was also met. The user interface is easy to use and meets the objective of being user friendly and pleasing to the eye. Finally, the overarching objective of designing a wireless application that is available around the clock by a cell phone, thus freeing the traveler from the necessity of being at a desktop computer, is met with this project design and implementation.

WAP technology is the network layer responsible for delivering WML to mobile handsets that are WAP ready and network enabled. WAP supports developing wireless websites that are compatible with different cell phone browsers. WAP also helps to build secure websites as well as websites where speed is crucial. The simplified WML code can deliver more information compared to HTML in the same number of bytes. The reduced number of round trips over the network makes WML suitable even on slower networks. The website is designed specifically for wireless cell phones, which allows us to focus more on content and functionality than flashy graphics.
REFERENCES
REFERENCES


APPENDIX A
ADDITIONAL SCREEN SHOTS

Log Out Page

The user sees the Log Out Successful message when he or she clicks on the logout option on the user home page. This is illustrated in Figure A-1.

Fig. A-1. Log out page.