INFS3204/7204 Practical 6 – Car Rental Web Application: Part 2

Note that: for all pracs in this course, you need to submit your code in *a single zip file* to the blackboard. Pracs without submitting your codes *before your own prac session expires* will be marked 0. If you submit the codes before your own prac session expires and do not attend the prac session, you have to email your tutor to ask him to mark it offline.

The goal of this practical is to explore the .NET MVC by creating the second prat of a Car Rental web application. All practicals will have to be developed with Microsoft Visual Studio 2010 using C# as the programming language. No other languages will be accepted. This practical contributes to 5% of your overall grade. You must **submit your code before your scheduled lab session expires in week 10**, in order to get it marked. **You are not allowed to change your session.**

Preparation

You have to complete practical 5 before attempting to do this practical. Some of the Web Services implemented in practical 5 can be reused in practical 6. For this practical, you are only allowed to use **MVC** and **LINQ 2 SQL**. This practical should show your understanding of SOA development in the aspects of analysis, design and implementation. The web application has to be user-friendly and highly interactive. The emphasis is on implementing **web services for each functional component accordingly, and the web services compositions**. This course is about Service Oriented Architecture design. So, it is very important to design your application with the implementation of web services and their compositions in mind. You are encouraged to look at various car rental reservation websites to get some ideas.

Car Rental Web Application: Part 2

For this practical, you have to **extend** the Car Rental application that you have created in practical 5. There is a small modification on the database that you had given for practical 5. You can download the new database file from here. (http://itee.uq.edu.au/~shenht/UpdatedCarRentalDatabase.zip)

- **engine_power** attribute has been added to the **Model** table: this attribute indicates the engine power of each model that is calculated based on the **horsepower (hp).**

The Part 2 of your car rental web application includes the following tasks:

Task 1: Compare (1 Mark)

For this task you are required to provide the user with the ability to compare two vehicles. After displaying the search results to the user in the **display results page** (that you have done in practical 5), he/she should then be able to select two vehicles for comparison (you may use checkboxes for each row of the results, or any other ideas). Upon clicking on a compare button/link, the user then should be redirected to the **comparison page**, and the comparison results for the selected vehicles should be displayed to the user. Please note that the comparison should be based on the **daily hire rate**, **engine power**, and **popularity** (considering the total number of bookings that have been made for the selected vehicle inside the Bookings table). For each of these comparison criteria, you also need to somehow display to the user that which vehicle is better in each comparison criteria. As an example on what does the vehicle comparison means, please take a look at the following table:

	daily hire rate	engine power	popularity
Vehicle A	$ 65	125 hp	10
Vehicle B	$ 60	90 hp	12
The winner is:	Vehicle B	Vehicle A	Vehicle B

Task 2: Booking Cancellation (1 Mark)

For this task, you are required to provide the user with the ability to cancel one of his/her **upcoming** bookings. In the **cancellation page**, the user can enter his/her **Passport number,** and then a list of **upcoming** booking for **this user** will be displayed. The result list should include **booking_id** and **vehicle_id**. There should also be a link/button for each row of the results so that user can cancel that booking by clicking on the provided link/button. Upon clicking on this button/link, the associated booking record should be deleted from the Booking table.

The **booking_id** and the **vehicle_id** in the list of results should be displayed as a link so that user can be redirected to a **booking review page** and **vehicle review page** accordingly. The vehicle review page is exactly the same as what you have done in task 3 of practical 5. The **booking review page** is what you have to do in this practical as explained in **task 3. Please note** that the user should then be able to go back to the **cancellation page** where he/she can see the previous booking results list, and can also cancel a booking from the list.

Task 3: Booking Review (0.5 Mark)

Upon a booking selection, in the **cancellation page**, the user will be redirected to a **booking review page** to view all the information of the booking. You are required to display **all** the information about the booking, including the booking-id, booking_status_description, customer's first name, customer's last name, pick_up date, drop-off date, pick-up city and drop-off city.

As mentioned in the previous task, the user should then be able to go back to the **cancellation page** where he/she can see the previous booking results list, and can also cancel a booking from the list.

Task 4: Constraints Checking (2 Marks)

For this task, you are required to extend what you have done so far in order to make your application more reliable against user inputs for **search** and **booking**. Please make sure that you meet all the following error checking requirements for this task (you are required to display an appropriate error message in each case):

- **Customer's age (0.5 Mark):** After user enters his/her information in the **booking page,** if the customer's age is less that 18 then he/she shouldn't be able to make the booking.
- **Vehicle availability (0.5 Mark):** When user performs a search in the **search page**, you should also check the availability of each vehicle in the specified pick-up city. This can be done based on the vehicle's **total** availability (specified in the Vehicle table) and the total number of bookings that have been made for the vehicle (in the Booking table). The application then should not display those vehicles that are not available for hiring to the user.
- **Duplicate booking (0.5 Mark):** When user wants to make a booking for a vehicle, you should also check the Booking table and prevent the user from making a duplicate booking for the same vehicle with the **same** or **conflicting range** of pick-up and drop-off dates.
- **Booking restriction (0.5 Mark):** There should also be a booking restriction on the total number of bookings that each customer can make for each date (**max 2 bookings per customer for the same day** in this pratical). This means that when a customer wants to make a booking, you should check all other bookings for

him/her in the Booking table to prevent the user from making more than 2 bookings for the same day (you should check for the ranges of booking dates).

Task 5: Exception Handling (0.5 Mark)

For this task, you should perform exception handling in **all** parts of your application. This is mostly important in those parts that your application receives inputs from the user and also where there are database communications. Consequently, your application should not crash as a result of an incorrect input format, or as a result of a communication error with the database.

This means that you should:

- Check input validations in **all** parts of your application. For example, you should check **date** validations, **age** validation (to be an integer), **email** validation, **postal code** validation (the post code should match with the state, where user enters his/her address), ...
- Handle the potential exceptions that might occur when your application communicates with the database.

You may find the following link useful for exception handling:

http://msdn.microsoft.com/en-us/library/ms173160.aspx