INFS3204/7204 Practical 5 –Car Rental Web Application: Part 1

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 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 9**, in order to get it marked. **You are not allowed to change your session.**

Preparation

For this practical, you are only allowed to use **MVC** and **LINQ 2 SQL** for your database queries. 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.

A Car Rental database has been provided to you. **You are required to use this database for this practical**. Please refer to figure 1 for the database schema. You can download the database from here (http://itee.uq.edu.au/~shenht/CarRentalDatabase.zip) and import it into the SQL server in visual studio.

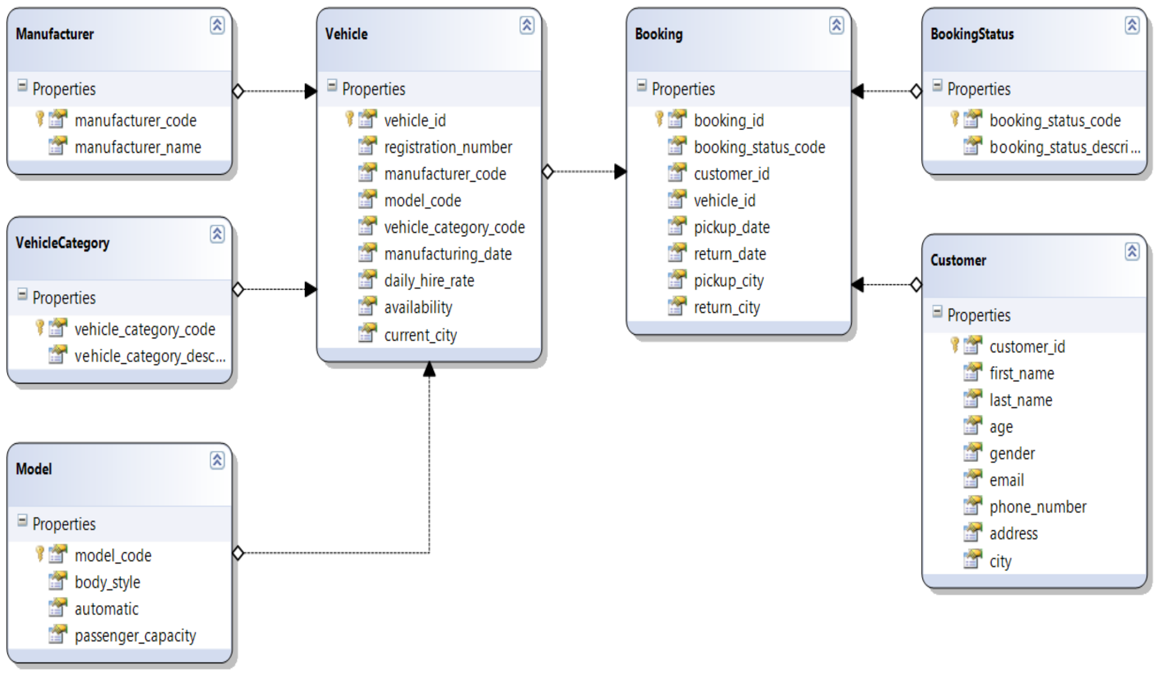

Figure 1: Database schema for Car Rental

Manufacturer table:

- manufacturer_code (primary key): the World Manufacturer Identifier (WMI) code that has been assigned to car manufacturers around the world (e.g. 1J4 (Jeep), 1C3 (Chrysler), WBA (BMW),...)
- manufacturer_name: the name of each manufacturer (e.g. Volvo, Chrysler, BMW,...)

VehicleCategory table:

- vehicle_category_code (primary key): the code that has been assigned to each vehicle category name (1, 2)
- vehicle_category_description: the description of each vehicle category (commercial, non-commercial)

Model table:

- model_code (primary key): the code that has been assigned to each car model by its manufacturer
- body_style: the body syle of the car (e.g. coupe, convertible, Standard, Van, SUV, Luxury, ...)
- automatic: whether the model is manual or automatic
- passenger_capacity: the maximum number of passengers for the model

Vehicle table:

- vehicle_id (primary key): a unique identifier for each vehicle
- registration_number: the vehicle's registration number

- manufacturer_code: foreign key to the Manufacturer table
- model_code: foreign key to the Model table
- vehicle_category_code: foreign key to the VehicleCategory table
- manufacturing_date: the production date for the **vehicle**
- daily_hire_rate: the rate of renting the vehicle per day
- availability: the **total** number of the same vehicle that the car rental agency has for rent in a city
- current_city: the current city that the vehicle exists there for rental (can be picked up from this city)

Booking table:
- booking_id (primary key): a unique identifier that has to be assigned for each booking (you have to create this id every time that a booking happens. e.g. 9983455654)
- booking_status_code: foreign key to the BookingStatus table
- customer_id: foreign key to the Customer table
- vehicle_id: foreign key to the Vehicle table
- pickup_date: the date specified by customer to pick up the vehicle
- return_date: the date specified by customer to return the vehicle
- pickup_city: the city specified by customer to pick up the vehicle
- return_city: the city specified by customer to return the vehicle

BookingStatus table:
- booking_status_code (primary key): the code that has been assigned to each booking status
- booking_status_description: the description of each booking status (e.g. payment pending, payment confirmed, picked up, returned,...)

Customer table:
- customer_id (primary key): an unique identifier for each customer (can be passport number or ID card number)
- first_name: the customer's first name
- last_name: the customer's last name
- age: the customer's age
- gender: the customer's gender
- email: the customer's email
- phone_number: the customer's phone number
- address: the customer's address
- city: the customer's city

The Car Rental Web Application

You are required to create a working prototype of a Car Rental web application that allows users to rent cars online. Each customer can have multiple bookings, but each booking can have only one customer (just information of the person that makes the booking has to be saved).

The application is divided into five major tasks (**Please read the whole specification before attempting to do this practical**):

- **Task 1: Search (1 Mark)**
- **Task 2: Sort (1 Mark)**
- **Task 3: Review (1 Mark)**
- **Task 4: Booking (1 Mark)**
- **Task 5: Payment (1 Mark)**

Task 1: Search (1 Mark)

The application will allow users to search for existing vehicles based on their desirable **vehicle category**, **body style, pick-up/drop-off city,** and **pick-up/drop-off date**. A list of available vehicles including the manufacturer's name and model's code together with the daily rate of rental will then be displayed to the user in the **display results page**.

Task 2: Sort (1 Mark)

In the **display results page**, users should be able to sort the results by the following options: **Manufacturer's name, Manufacturing date** and **Daily hire rate**. By default, the results are displayed in the increasing order of daily hire rate.

Task 3: Review (1 Mark)

Upon a vehicle selection, in the **display results page**, the user will be redirected to a **review page** to view all the information of the vehicle. You are required to display **all** the information about the vehicle, including the registration number, manufacturer name, model code, body style, automatic, passenger capacity, vehicle category description, manufacturing date, availability of the vehicle, daily rental rate, and **total rental cost** (based on the pick-up date, return date and daily rate).

Task 4: Booking (1 Mark)

After reviewing the selected vehicle, the user can go back to the results list (**display results page**) or confirm the selection to proceed to the booking procedure. Upon booking, the user will be redirected to a **booking page**, where he/she can enter the booking details including the following details:

- Passport number / ID card number
- First name
- Last name
- Gender
- Age
- Email
- Phone number
- Address
- City (where the customer lives)

You should have the basic validations (required field validations) for all of the above fields. Note that age should be at least 18 or above. After the user has finished entering the information, he/she will be redirected to the **payment page**.

Task 5: Payment (1 Mark)

In the payment page, the user will only be able to pay by using a credit card. The detailed vehicle information, customer information and total hiring cost will be displayed to the user.

The user has to enter the following details to make the payment:

- Select the credit card type (Visa, Master, ...)
- Name on the card
- Credit card number
- Security number (3 digits)
- Expiry Date

After entering the credit card information, the credit number will be checked against a **credit card web service** to see if the credit card number is valid. Once the validation of the credit card succeeds, all the information of the customer and his/her booking have to be submitted to the database and the user is being redirected to the **receipt page,** where he/she can view the vehicle information, customer information, and hiring cost information. The receipt should also be sent to the customer via email.

If the credit card validation is unsuccessful, the user will be requested to input the payment details again. You are required to either write a credit card number checking web service to check the credit card number (refer to http://wiki.cdyne.com/index.php/Credit_Card_Verification on credit card number checking algorithm) or call an existing external credit card number checking web service. **Please note** that you are **not** required to contact the credit card company for real validation.