Ryde Now Rentals

Group 2

David Keith
Larry Ogunjobi
Jarvis Williams
Chris Foley

ITCS 3155 – Software Engineering
University of North Carolina at Charlotte

Table of Contents

Team 2 - Car Rental App
Initial Project Proposal and Plan

1. Introduction

Our group wants to provide an application that allows the user to reserve a car for rental. The sizes of the rental will be small (hold 4 people), medium (hold 5-6) people, and large (hold 7-8) people. This application will be available for use on mobile phones.

1.1 Project Overview and Statement of Proposal

Statement of Proposal: *We propose to create a computer application that allows people to reserve a car for rental.*

1.2 Project Scope and Objectives

The scope of the project will be for the city of Charlotte.
Objectives include:

- Project proposal (**Milestone 0)**
- Collection/Documentation of User Stories → Completion of Product Backlog (**Milestone 1A**)
    - Add "efforts" section to backlog to describe an estimate of how much effort or time each main task will require
        - how much time will designing exporting, importing, et cetera for database take?
- Document Projects Sprints for project in project backlog: Sprint 1, Sprint 2, Sprint 3
- Develop Gantt Chart to Have an Overview of the entire project
- Document Use Cases using Abstract Template (set High, Medium, Low priority)(**Milestone 1B**)
- Database Selection/Development
    - We will select a pre-made car database or design a simplified database of our own
- Code Application & Connect Car Database to App **(Milestone 2a)**
- Design Front-End of App → Sketch **(Milestone 2b)**
- Test application / Fix bugs **(Milestone 3)**

2. Plan

This section contains is a list of tasks and deliverables associated with the project, a Gantt chart depicting task durations, dependencies and completion dates, and a summary of resource requirements and assignments for each task.

2.1 Timeline Chart

Gantt Chart (with dates)

Production	Dates				Number of Days
Sprint 1	*Start: 7/10*	*Finish: 7/20*			10 days
Sprint 2		*Start: 7/17*	*Finish: 7/27*		10 days
Sprint 3			*Start: 7/24*	*Finish: 8/5*	12 days

2.2 Task Descriptions

Task 1 - User Stories

Task 2 - Abstract template

2.3 Resource Table

Task	People	Hardware & Software	Special
1	David Keith	Database Source Code Files	Team Moderator/Manager
2	Jarvis Williams	Java Source Code Files	Note Taker
1	Chris Foley	Database Source Code Files	
2	Larry Olumide	Java Source Code Files	Time keeper
3	David Keith, Jarvis Williams, Chris Foley, Larry Olumide	Entire team will work on application driver used to connect all the class file(java)	

3. Project Resources

3.1 Group Members

1. David Keith
2. Jarvis Williams
3. Chris Foley
4. Larry Olumide

3.2 Hardware and Software Resources

Indicate the anticipated software and hardware resources required to complete the project.

- Hardware: computer
- Software: Java IDE/VM/JDK, MSSQL server, PayPal payment system

3.3 Special Resources

There are no special resources needed to complete the project.

4. Appendices

There is no additional information to include.

Software Requirements Specifications

1. Introduction:

- *1.1 Purpose: The purpose of this application is to allow a new era of car renting, it cater to clients that are always on the go but need transportation at the push of a button.*

- *1.2 Document Conventions: This is used to describe any standards or typographical conventions being used. It often will include the meaning of the text styles, highlighting, and notations. Which makes it easier to understand and access. The language will be pretty standard terms such as "Rentals, Receipts, Late Fee, and Pick up" will be the common lingo.*

- *1.3 Project Scope: This provides a short description of the software being specified along with its purpose. This is then used to relate the product to the audience trying to be reached. Stick to plan in a sense, as more information and ideas begin flowing keep the scope to a state of incremental progress so as the product progress so does the size of the scope.*

- *1.4 References: This is just a list of resources and documents that the SRS has referred to throughout the entire process. This would consumer reports, car reviews, manufacturer catalogs, etc.*

2. Overall Description

- 2.1 Project Perspective:
  This is an application is designed to allow other the ability to reserve vehicles for rent. The software will draw the information on available cars from a connected database. This application can work on a computer or a mobile phone.

- 2.2 User Classes & Characteristics:
  The user classes are salesperson, manager, and customer. The salesperson is concerned with acquiring vehicle rental reservations, the manager is concerned with what the salesperson is concerned with and quality assurance also, the customer is concerned with being able to get the vehicle size of their choice for the specified amount of time.

- 2.3 Operating Environment:
  The car rental reservation system will be able to operate on any mobile device (hardware) with an android operating system and on any computer (hardware) with a windows operating system of windows XP or higher. The system will need to peacefully coexist with the car database.

- 2.4 Design & Implementation Constraints:

Considerable time and resources were placed into creating the cars database and the group must use that database because we do not have the necessary time to develop another from nothing. The car database was put together using SQL in a software called MySQL. If developers wanted to modify the database they would need to understand SQL and have equivalent software to open the database and make the modifications (i.e. MySQL). Time is another constraint; we only have three weeks to design this software system. We must code the application for the software system in a language that all member of the software engineering team are familiar with; we chose to use Java. We must code the program to connect to the database using the java database prewritten code for connecting databases written in SQL.

- 2.5 Assumptions & Dependencies:
  - All team members have a working advanced knowledge of the java programming language. The source code for the project will be a mixture of prewritten code and original code. In the software requirements specification (SRS) chart in the book on the page listed for question 2 on the test, it lists all of the things that we the developers of the software assume to be true (we don't have to have proven that it is or not) and others can assume to be true also, and it deals with the critical things in the project that depend on one or more of something else. As a team of software engineers, our group chose to program the vast majority of the project using the java programming language and we assume that each team member has a very good understanding of basic java and its uses in OOP as it relates to this car rental reservation software system that our team is designing, since we all learn java as part of the CS curriculum at UNCC. We also can assume that the actual source code for the app part of the car rental reservation system that we are designing will be made of code we have personally written ourselves and code that we have acquired the necessary permissions to use (prewritten code). For example, consider the source code files that David and Chris have posted on the group share website and also prewritten java classes from the java API. I did not think of anything I considered noteworthy of mentioning that depends on something else that cannot already be assumed.

3. System Features:

- 3.1 Search and Reserve
  - 3.1.1 Description
    Allows the customer to view and adjust search criteria for available rental cars. Also gives them the option to cancel rental reservation prior to rental pick up date, and shows the nearest rental location. Priority: High

  - 3.1.2 Functional Requirements
    The system needs to be able to sorted and searched through as well as have a database that can store the customer's information to the car reservation they picked out. If user inputs invalid search terms the search will display a message saying search criteria not valid, please try again.

- 3.2 Management tools
  - 3.2.1 Description
    Allows for manager to keep track of sales, cancellations, payments and payment validations. Priority: High

  - 3.2.2 Functional Requirements
    The system needs be able to allow certain users access to the database that holds the customers' reservations. System needs a search function for this database as well to make manager's job easier when trying to track information. Prompts will be made if a payment method was rejected.

- 3.3 Profile and Notifications
  - 3.3.1 Description
    Allows customers to create a profile and be able to receive notifications. Also allows for user to be able to view their past activities. Priority: High

  - 3.3.2 Functional Requirements
    System needs to have a database able to store customer profiles and notify them when important information is going to/has happened. System also needs to be able to store customer's history to their profile.

- 3.4 Inventory tools
  - 3.4.1 Description
    Allows for manager to view the inventory in terms of quantity, rental status, location, and size. Priority: High

  - 3.4.2 Functional Requirements
    System needs to allow certain users to access the inventory database and allow them to search through and update the database.

- 3.5 Penalty
  - 3.5.1 Description
    Allows manager to see which rentals are about to end and can send penalties to customers if they are late Priority: Medium

  - 3.5.2 Functional Requirements
    Manager tools need to be implemented to allow managers to see rental information. Also system needs to send out notifications to both manager and customer notifying them that their/a rental is late. The manager would also receive a late fee request to allow them to send a customer a late fee.

- 3.6 Quality Assurance
  - 3.6.1 Description

Serves as a medium for customers to report they satisfaction with the services provided. Also allows managers to see if the customer is enjoying the services and if something needs to be improved upon. Priority: High

- 3.6.2 Functional Requirements

System needs to allow customers to place their feedback either through survey, forum or by phone and store the results into a database. Mangers also need to have access to that database so they can look through it and see how customers view their system.

4. Data Requirements

- 4.1 Logical Data Model
  (see corresponding Logical Data Model file – Appendix B)

- 4.2 Data Dictionary
  (see corresponding Data Dictionary file in glossary – Appendix A)

- 4.3 Reports
  (see corresponding Reports file – Appendix B)

- 4.4 Data Acquisition, Integrity, Retention, and Disposal
  The Database runs off an SQL server allowing the addition, deletion, and updates required of the system. Additionally, with it's username and password requirements, the data is protected. The group is using Google Drive, which will also be used to store backup copies of the SQL files. New backups will be created after every editing session.

5. External Interface Requirements

- 5.1 User Interfaces

As far as user interface goes, the main design of the website came from taking into account just how users would interact with the site both on the front and back ends.

When choosing the look of the site, various rental websites were analyzed and their specific features were noted. The sites were approached from the aspect of a consumer looking for a rental car.

After comparing various sites a bulleted list was compiled based on feedback from group members and selected individuals on what should be key characteristics in the user interface for the site.

Font, Icons, Button Labels, and etc. Were left in a general form as they didn't hold much weight in the opinion of the selected individuals.

The background that was selected prior to gaining peer feedback proved to be a successful choice. Given the rental site would be based in the Charlotte area, using a background that displayed not only the nightlife but the beauty of Uptown Charlotte itself made the page very appealing to the eyes. Which in the case of car rental services is a very important component.

The templet that was selected was used in order to make navigation through the website easier however the color scheme was meant to create a correlation between itself and the background image allowing for flow and easy visibility. Car types are broken up into three categories Economy, Middle Class, and Luxury. The base rates are displayed below in order to give the consumer an idea of just how much they may have to pay.

- 5.2 Software Interfaces

  In this project we used SQL Server 2012 as the programming platform for the AutoRenter database. Afterward we paired CSS/HTML, java script and the SQL code to put together the actual product (the website). The users of this website will not not need any extra software other than an internet browser to access the site.

  SI-1.1: The website shall transmit the information (dates needed to rent vehicle, size of vehicle etc.) to the AutoRenter database through a programmatic interface.
  SI-1.2: The website shall poll the AutoRenter database to determine whether a requested vehicle for a requested period of time is available.

- 5.3 Hardware Interfaces

  There really is not any dedicated hardware or hardware interfaces used. The only hardware that is required is a computer, internet access, and a web browser.

- 5.4 Communications Interfaces

  The communications interfaces include a web browser. Due to the program's use of ActiveX, the browser must be compatible in order to include a functional database. However, the front end website will appear on any web browser. Email and phone numbers are provided for customer service reasons but they do not serve a key role in functionality.

6. Quality Attributes

- 6.1 Usability

  Usability was a key component in the development of this site. The structure of the site was made so simple that even a child could navigate through it.

  Keyword were expressed various time to make navigation smooth and simple Visually the color scheme and backdrop allowed users to feel important and exemplified as the look through car catalog

  Prices and Base Rates are displayed in both car rental and car purchasing pages to allow the user to quickly figure out which range they fall in.

  Contact forms are placed on multiple pages so that if user has question it can be emailed out and quickly answered.

  Information placed on each page was straight and to the point which makes the information easy to read and identify

The entire interface as previously stated was created with the mindset that even a child would be able to navigate through it using keywords and other given information.

- 6.2 Performance

  PER-1 the system shall accommodate a total of 100 users and a maximum of 25 concurrent users during the peak usage time window of 9:00 A.M. to 3:00 P.M. local time, with an estimated average session duration of 4 minutes.

  PER-2: 100% of webpage shall download completely within 5 seconds from the time the user requests the page over a 30 Mbps or faster Internet connection.

- 6.3 Security

  Security implementation was a low priority for our product due to time constraints and the overall security that is product has is limited to what the webhost has provided. In an ideal environment we would have used php to implement the proper security protocols to allow the database access to be secure from the front end. In the backend for the manager access the database would be simply secured with login credentials.

- 6.4 Safety

  The safety requirements would involve the vehicles. The vehicles will be from large automotive companies and meet the all legal standards for safety regulations before being rented to customers. In order to preserve the vehicle quality (including safety related qualities) vehicles will be assigned a routine maintenance schedule based around the vehicle age and mileage. Additionally, the vehicles will be inspected daily to ensure that conditions are adequate. When not in use, the vehicles have doors shut and locked, windows closed, and will be protected by an adequate security system in order to prevent damage or theft.

  The customer will be asked to provide valid driver's license, insurance, and sign a contract stating that the vehicle will be returned at a certain time and in good condition before purchasing a vehicle. Violations of the contract will result in a fine and possibly a ban from renting from the organization in the future. The customer's insurance policy must cover the costs of the potential damage from vehicle accidents and assure our organization that the customer will be held accountable for any wrongdoings related to company inventory. If the renters have small children, a car seat will be required before checkout.

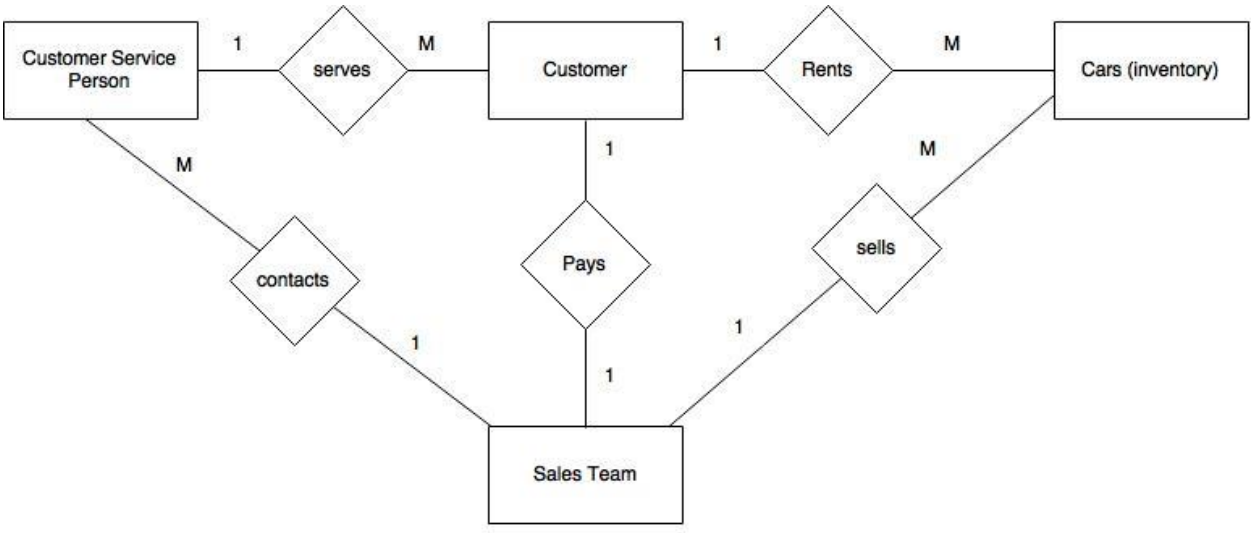
Appendix A: Glossary

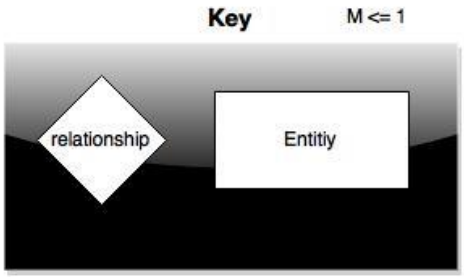
Data Dictionary

Data Element	Description	Composition or Data Type	Length	Values
Inventory Order	New cars shipped to the inventory location	VIN Number, Vehicle Make and Model, Cost, Date		
Inventory Reduction	Cars that are too old, have too many miles, or totaled are removed from the inventory.	VIN Number, Vehicle Make and Model, Revenue, Date		
Quantity	How many vehicles of each type are in the inventory	Positive integer		
Availability	Of the total inventory, which vehicles are currently available?	Positive integer, Date, Availability Status		

Appendix B: Analysis Models/Reports

Logical Data Model





Reports

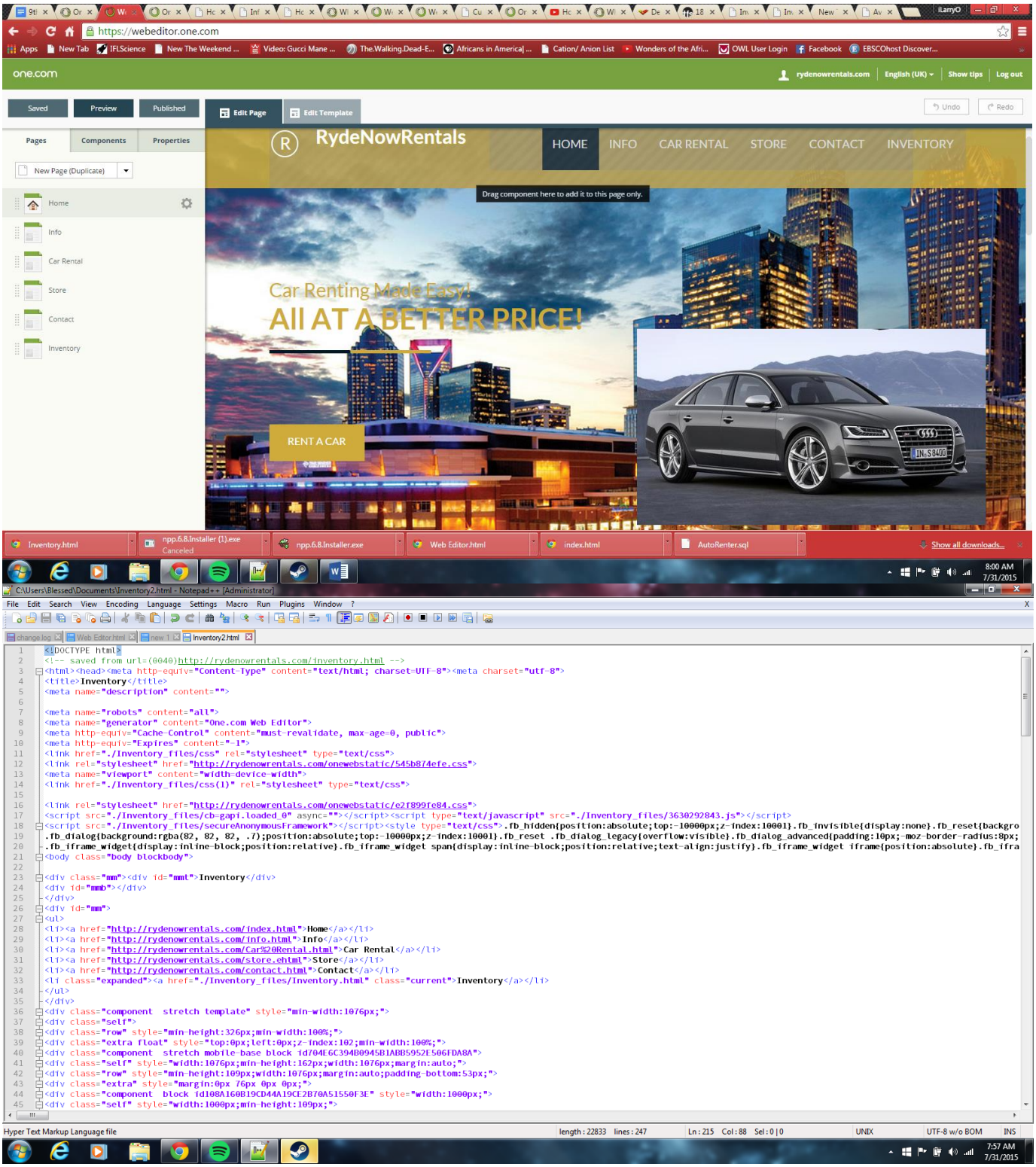
Report Element	Element Description
Report ID	Inventory_Report_001
Report Title	Inventory Report
Report Purpose	To give routine updates on the size of the inventory, the size of the available inventory, and the status of those vehicles in the inventory which are not currently available.
Decisions Made from the Report	To determine whether or not the inventory size is adequate and determine whether the quality of the inventory is optimal.
Priority	Medium
Report Users	Management
Data Sources	SQL databases and physical locations.
Frequency and Disposition	Dynamic Report, filed weekly, all data from the inventory tables of the database is included, the report is generated weekly, report is received by management
Latency	Report must be delivered within two hours of request. Data must be updated daily.
Visual Layout	Portrait, standard size printer paper, no graphs
Header and Footer	
Report Body	Fields: Inventory Size, Available Inventory Size, Unavailable Vehicle Status
End of Report Indicator	
Interactivity	
Security Access Restrictions	Access for management and Inventory personnel only.

Appendix C: Project Product (Website)

Throughout the duration of this course we were given a task, this task was to brainstorm an idea and then create a functional and tangible product. After brainstorming the group decide on a car rental product that would be able to compete with all other rental services. As task were being delegated the task of website design fell on me. Going through various websites in order to find one that would not only produce a functional but a presentable I set my eyes on One.com and webhosting website that would allow me to create my own ".com" domain and allow instant access to me and my colleagues.

The domain name that was used was entitled "Rydenowrentals.com", the site from my colleagues eyes looked great, the only issue at this point was functionality. When it came to the looks of the website it was top tier in design, flow, and appeal but once again the issue was functionality. Having a somewhat moderate understanding of HTML coding it was interesting to see just how I was going to correct the issue of implementing the database into the website. Finding a solution was not easy, after hours with technical support I was told by several different people that it wasn't possible on this site; that if I wanted to implement a personal database I'd have to use something along the lines of WordPress.

After going through numerous trials and errors, small victories were riddle with big defeats. I swallowed my pride and went on to word press to see if it was possible to replicate what I had already done in “RydeNowRentals.com” it seemed prominent but functionality was still an issue. At this point it was either throw away the entire idea or find a solution. I found a way to retrieve the HTML codes from the Ryde Now website which will allow me to recreate the website on any form or hosting site, my colleague emailed me last night saying he made a breakthrough and that what he found could be the underlying piece to solve our problems. (Figures 1-1 & 1-2)



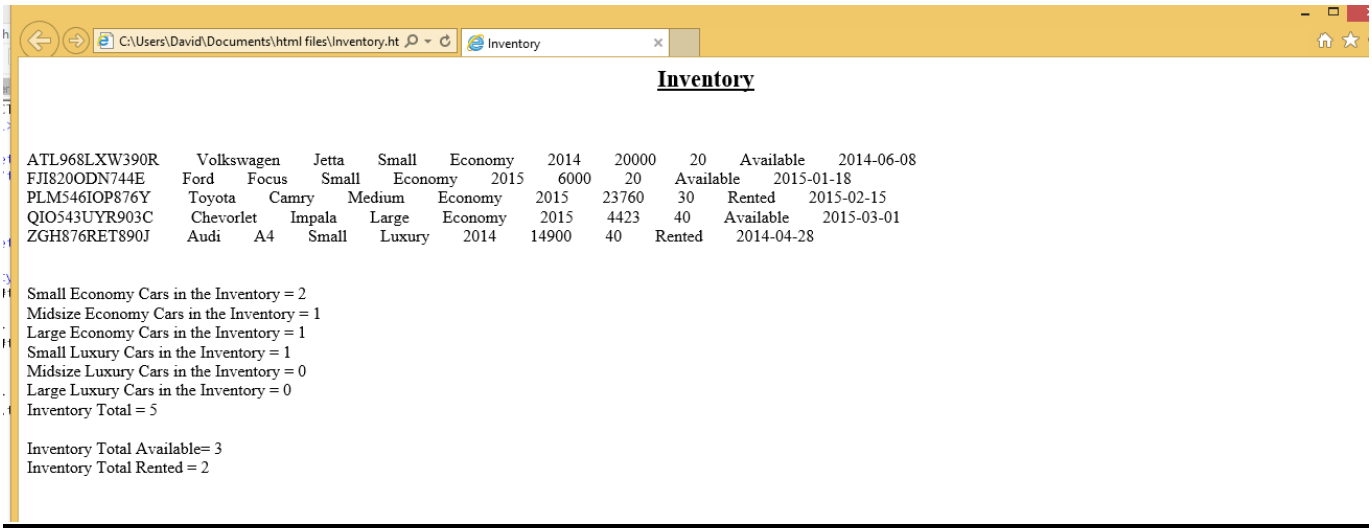


Figure 2-1: Report of the available cars from Database connected to website

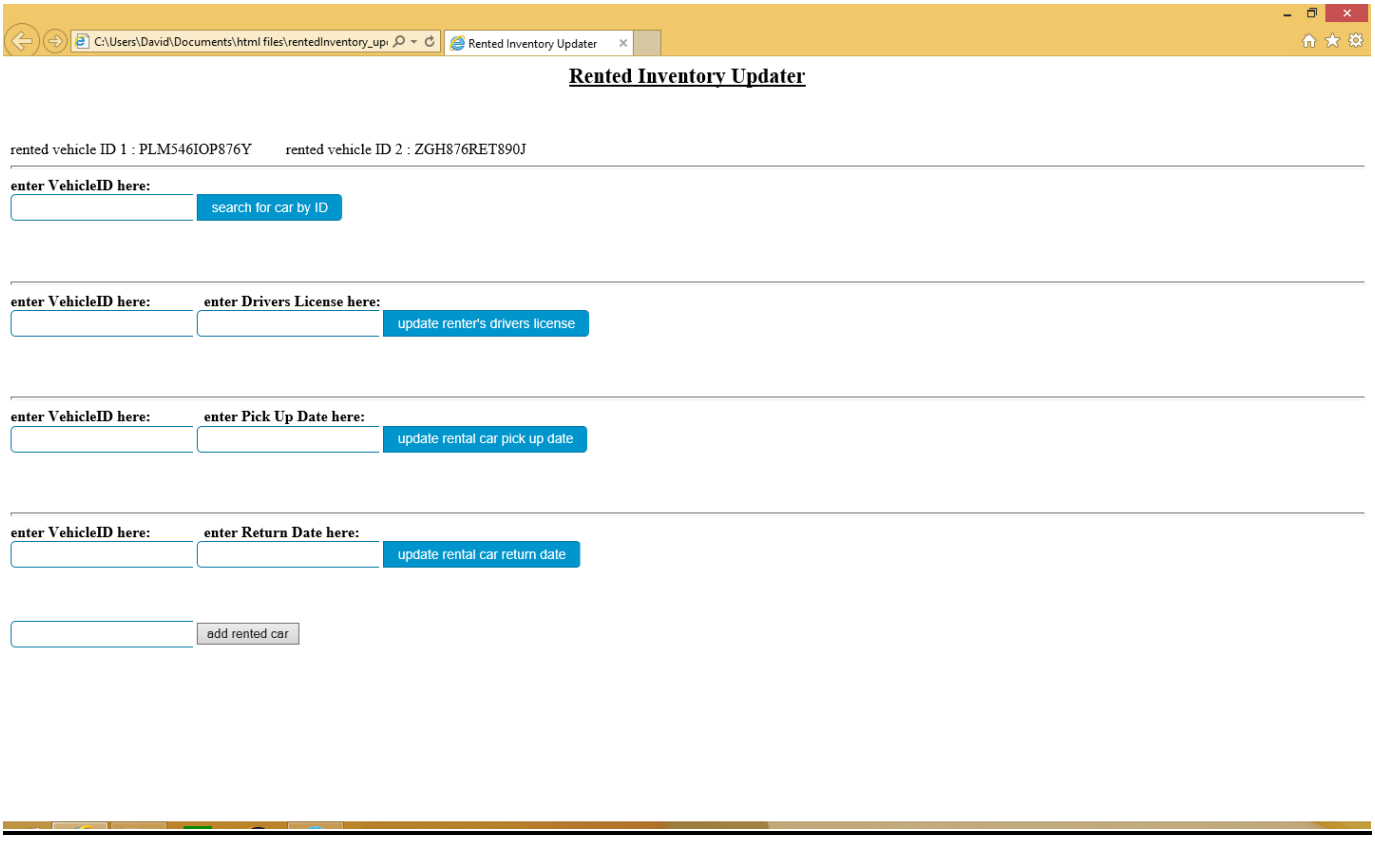


Figure 2-2: Screen shot showing how inventory can be updated

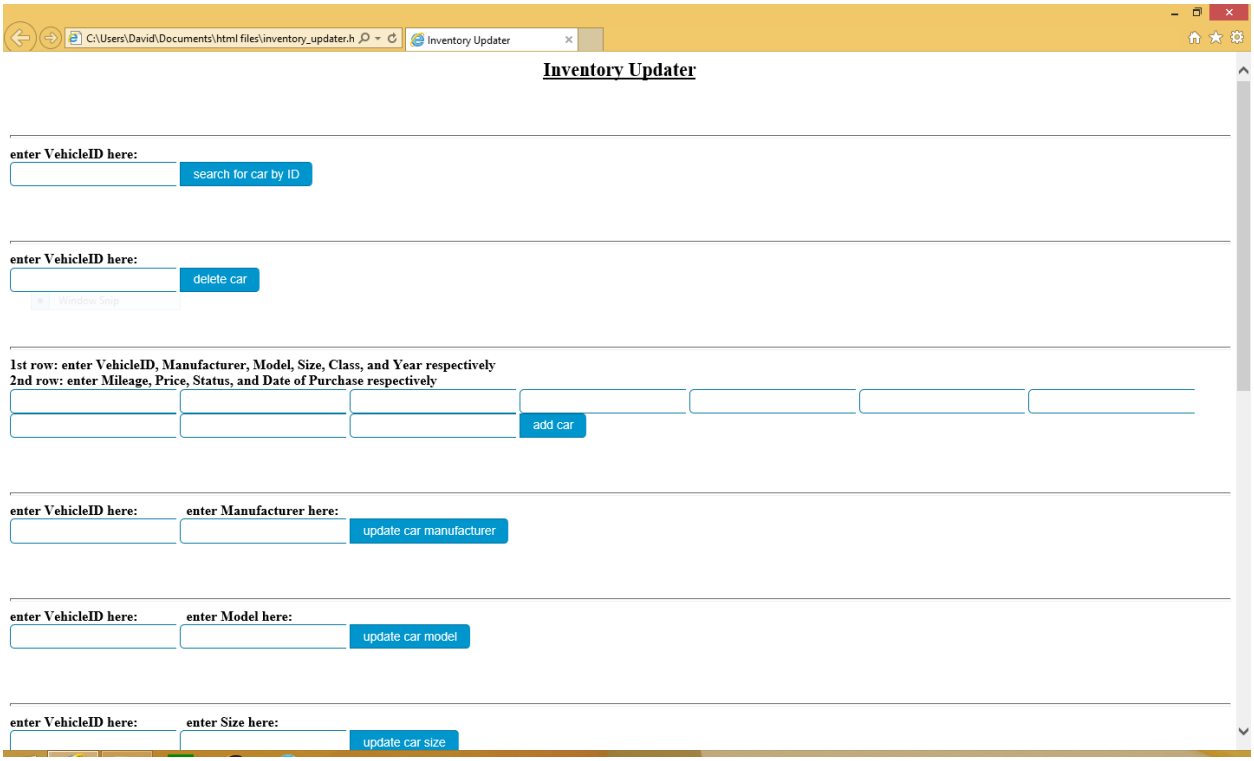


Figure 2-3-1: Showing how each individual field (column) of the inventory can be updated

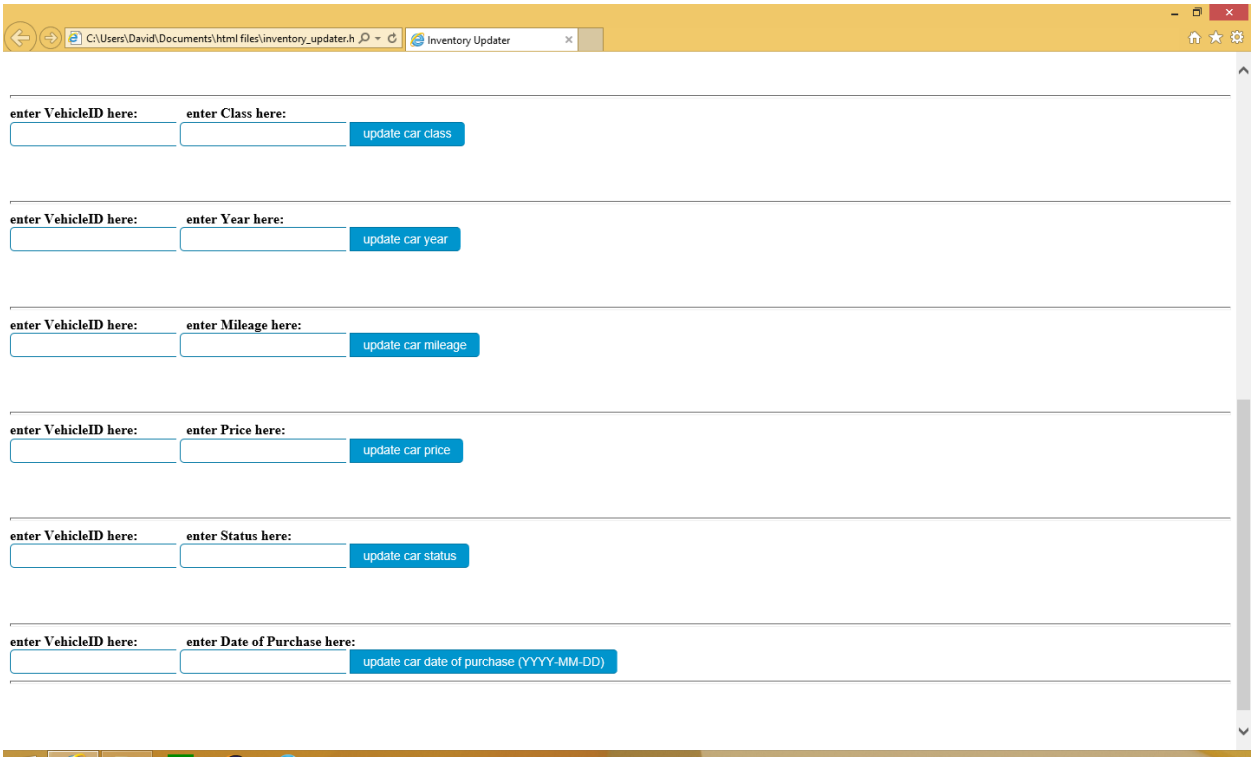


Figure 2-3-2: Showing more updates to the inventory fields

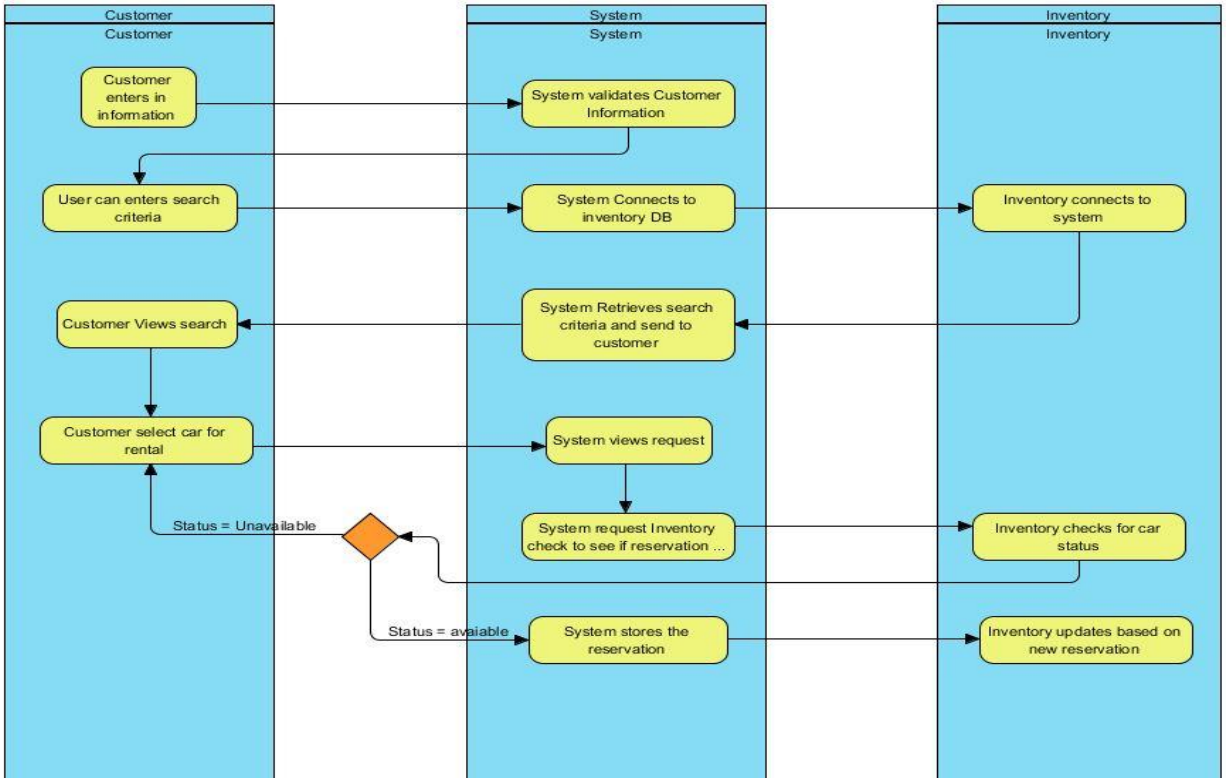


Figure 3-1: Swimlane diagram showing how a reservation is made in the system

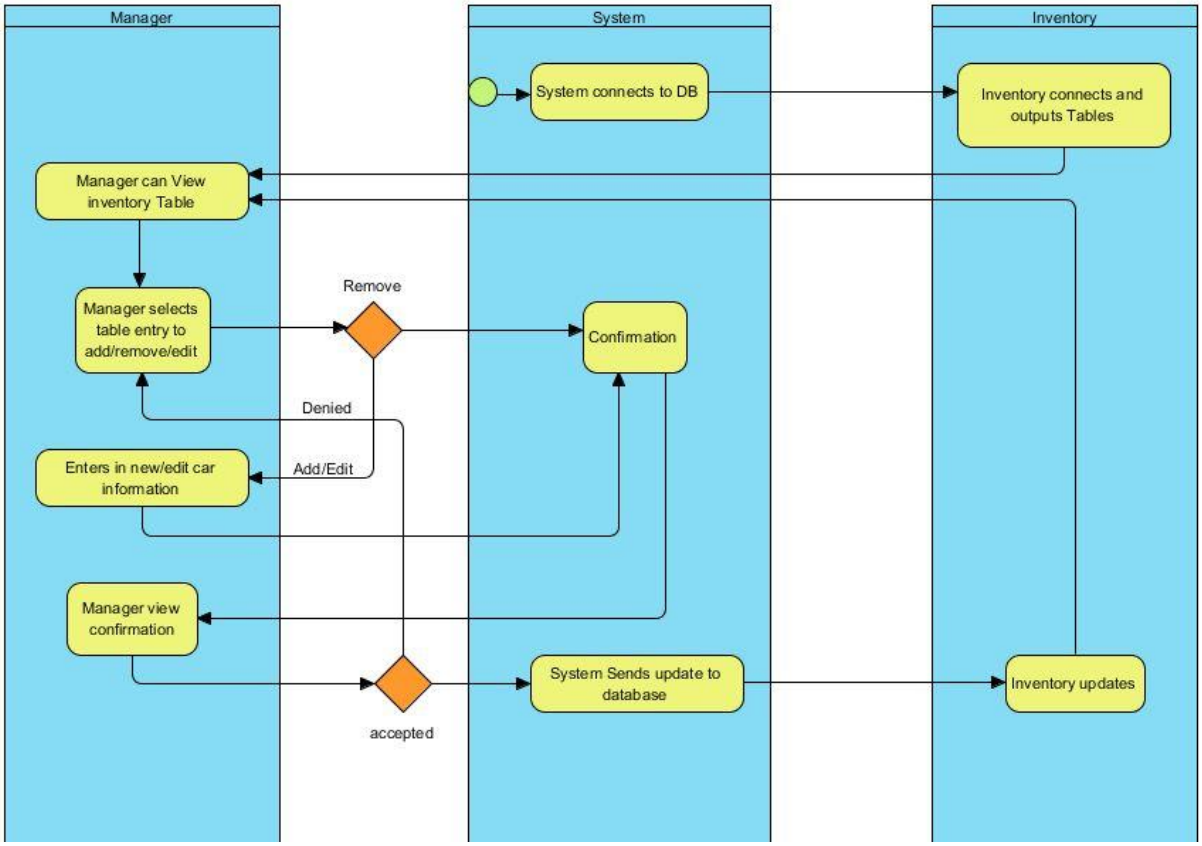


Figure 3-2: Swimlane diagram showing how vehicles can be added and removed from the database

Appendix C: Diagrams

Product Backlog (Figure 4: Showing the user stories used to develop the product)

1	Rental Location	1	Customer	*As a customer, I want to know the nearest place to rent a car.*			1 (Highest)	
2	Payment	1	Customer	*As a customer, I want to be able to cancel a transaction/rental reservation prior to my rental pickup date.*			1 (Highest)	
3	Search	1	Customer	*As a customer, I want to be able to view and adjust search criteria the cars available for rental.*			1 (Highest)	
4	Cancellations	2	manager	*As a customer service person, I want to have a system in place for if a customer pays for a service that they no longer need.*			1 (Highest)	
5	Bookkeeping	2	manager	*As an accountant, I want to have validated transactions automatically sent to the company's financial statements.*			1 (Highest)	
6	Receipts	2	manager, customer, sales person	*As an accountant, customer, sales person, or customer service person, I want to have varification that this transaction took place.*			1 (Highest)	
7	Payment Validation	2	manager, sales person	*As an accountant or sales person, I want to have varification from the financial institution representing the customer that the transaction was processed.*			1 (Highest)	
8	Payment Validation Request	2	manager, sales person	*As an accountant or sales person, I want to be able to send a request to the financial institution representing the customer in order to confirm that the transaction was processed.*			1 (Highest)	
9	Payment	2	manager, sales person	*As an accountant or sales person, I want to know basic information on payments such as who is paying, information on the financial intitutions involved, when was the payment made, and how much was the money was in the payment.*			1 (Highest)	
10	Notifcation	3	Customer	*As a customer, I want to recieve notifications letting me know that my rental is almost*			2 (High)	

				over or about to start				
11	profile	3	Customer	*As a customer, I want to have a profile regarding my activities with the company.*			2 (High)	
12	Inventory Reduction	4	manager, Sales Person	*As an inventory manager, I want to have a way of knowing if a car no longer meets our company standards, have a way to take it off the inventory, and sell or scrap.*			2 (High)	
13	Orders	4	manager	*As an inventory manager, I want to be able to order cars or parts and track orders in progress.*			2 (High)	
14	Maintainence	4	manager	*As an inventory manager, I want to utilize up to date information such as mileage on the cars as they are used in order to maintain them better.*			2 (High)	
15	Location	4	manager	*As an inventory manager, I want to know where to find each of my companies cars.*			2 (High)	
16	Count	4	manager	*As an inventory manager, I want to have a list of our inventory rented and available for rent.*			2 (High)	
17	Customer Penalty	5	manager, Sales Person	*As an inventory manager or sales person, I want to have a system in which the customer must pay a fine if the car is returned late or in terrible condition.*			3 (Medium)	
18	Time	5	manager, Sales Person	*As an inventory manager or sales person, I want to be able keep track of how long a customer has until the customer must return the car.*			3 (Medium)	
19	Complaints	6	manager	*As a customer service person, I want to be able to address legitimate issues that customers have with our service*			4 (Low)	
20	Feedback	6	manager	*As a customer service person, I want to be able to get valuable feedback from the customer regarding our service.*			4 (Low)	
ID	**Theme**	**Use Case**	**Role**	**Story**	**Notes**	**Effort**	**Priority**	**Sprint**
Larry = green, Chris = yellow, Jarvis = blue, David = red								

Figure 4 (continued): User Stories

Decomposition Functional Diagram

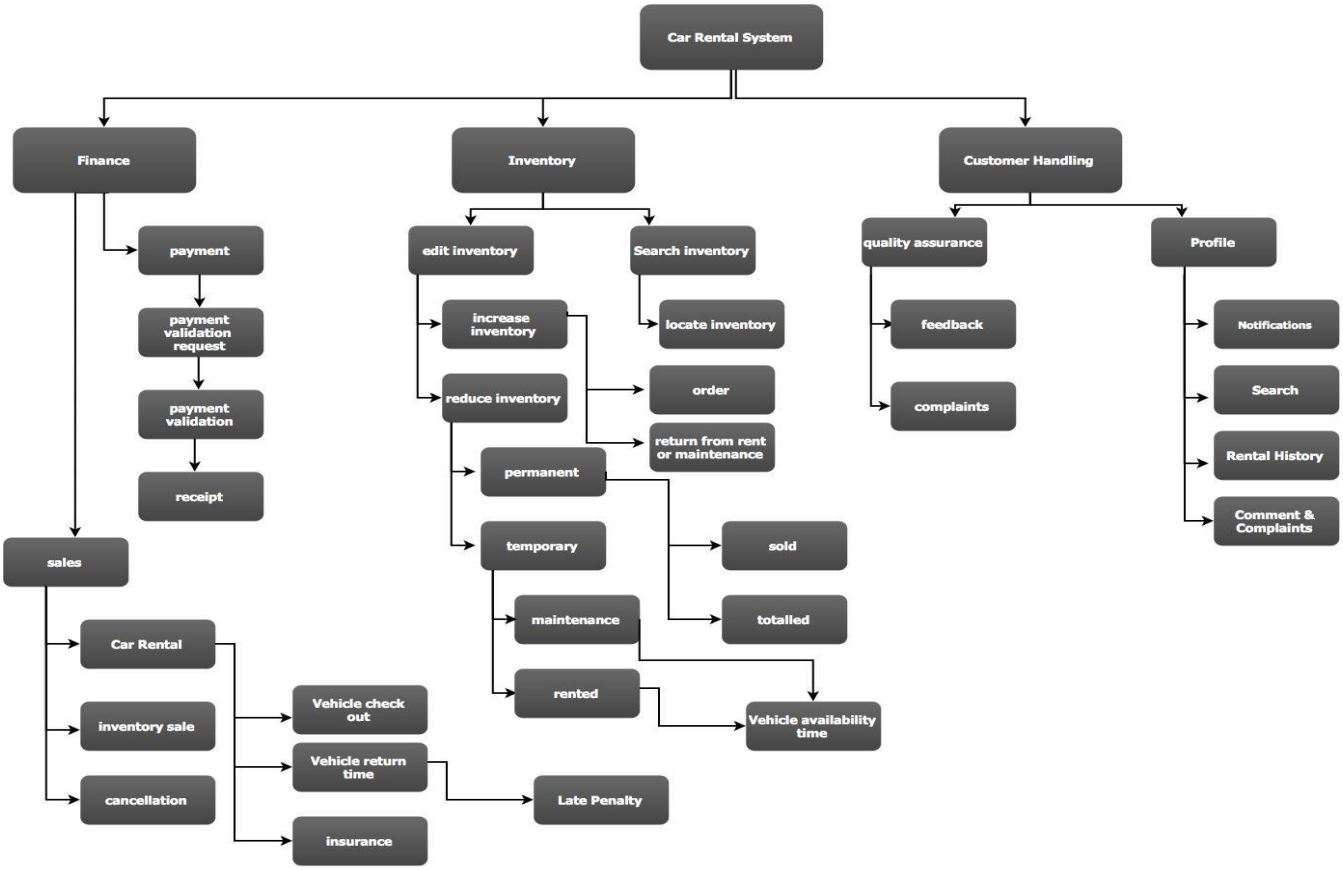


Figure 5-1: Decomposition Diagram showing each individual part of the product. This is more in-depth than the context diagram.

Context Diagram

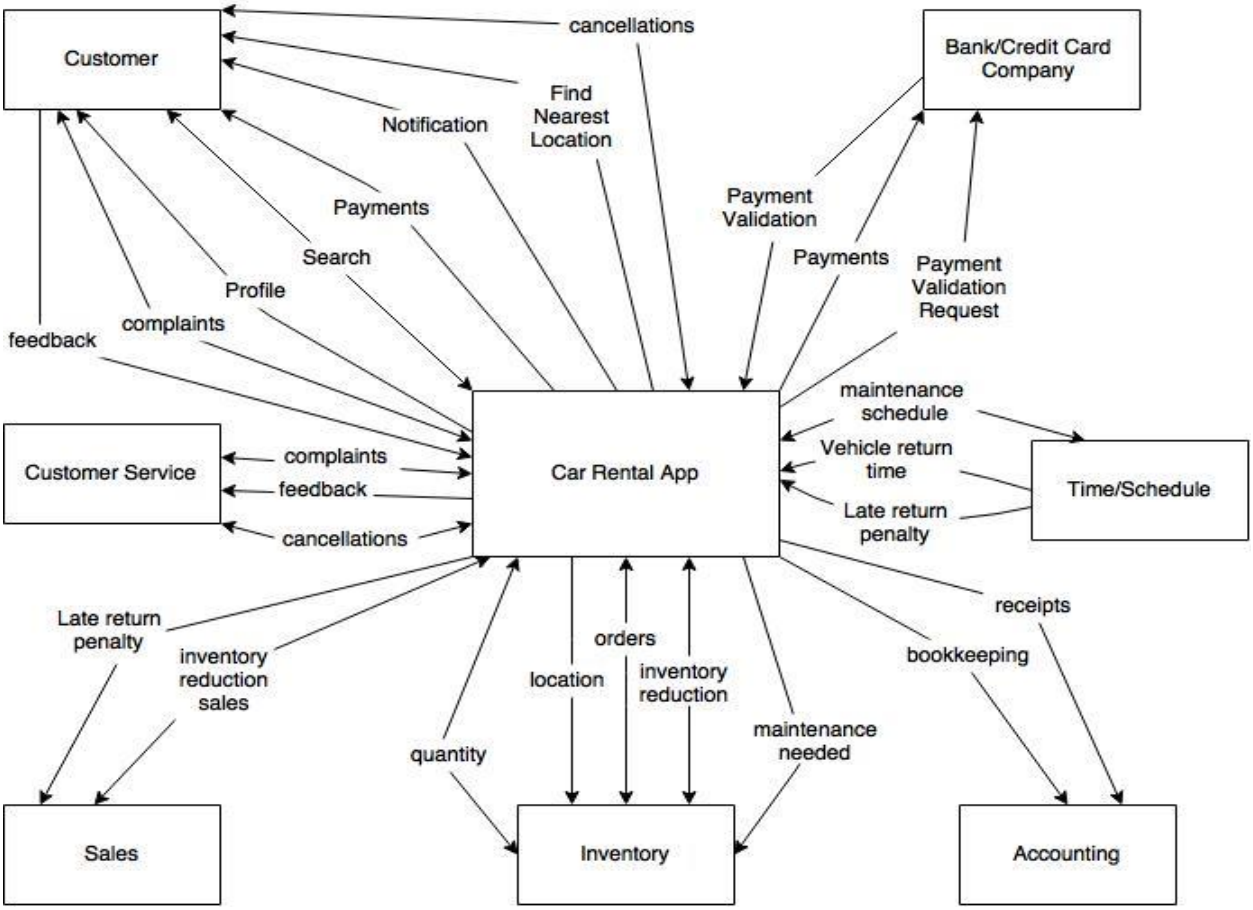


Figure 5-2: Context diagram showing the planning of the car rental reservation system.

Appendix D: Use Cases

CarRentalApp_01

Author (s): ___Group 2_____ **Date:** ____7/15/15

Version: _____

USE CASE NAME:	User Interface	USE CASE TYPE
USE CASE ID:	CarRentalApp_01	Abstract: ☑
PRIORITY:	High	Extension: □
INVOKED BY:	User	
PARTICIPATING ACTORS:	Customer	
DESCRIPTION:	Allows Customer to view and adjust search criteria for available rental cars, gives them the option to cancel rental reservation prior to rental pick up date, and shows the nearest rental location.	
PRE-CONDITION:	None	
TYPICAL COURSE OF EVENTS:	**Step 1**: Customer searches through vast catalog of cars, then adjusts search criteria to find a specific car. Step 2: After car selection, interface shows the Customer the nearest location for rental pick up Step 3: Interface also notifies Customer that if they wish to cancel their reservation it must be 2 days prior to pick up or a late cancelation fee will be accessed.	
ALTERNATE COURSES:	None	
POST-CONDITION:	None	

CarRentalApp_002

Author (s): ___Group 2_____ **Date:** __07/14/2015__

Version: ___1.0_____

USE CASE NAME:	Finance	USE CASE TYPE
USE CASE ID:	CarRentalApp_02	Abstract: ☑
PRIORITY:	High	Extension: □
INVOKED BY:	User	
PARTICIPATING ACTORS:	Manager, Salesperson	
DESCRIPTION:	Allows the manager/sales person to keep track of sales, cancellation, payments and payment validations	
PRE-CONDITION:	None	
TYPICAL COURSE OF EVENTS:	**Step 1**: Sales person or manager will enter the customers payment information **Step 2**: The system will send a validation request to the bank and validate the payment method **Step 3**: If payment method is accepted the system to generate a receipt for record and store that record in the system for the manager to reference	
	Step 4: Customer or manager will have the option to cancel at anytime	
ALTERNATE COURSES:	Payment Method Rejected	
POST-CONDITION:		

CarRentalApp_3

Author (s): _____ **Date:** _____

Version: _____

USE CASE NAME:	Profile and Notifications	**USE CASE TYPE**
USE CASE ID:	CarRentalApp_3	Abstract: ☑
PRIORITY:	High	**Extension:** □
INVOKED BY:	User	
PARTICIPATING ACTORS:	Customer	
DESCRIPTION:	Allows the user to create a profile and receive notifications	
PRE-CONDITION:	none	
TYPICAL COURSE OF EVENTS:	**Step 1**: User creates a profile Step 2: User can choose which contact method they prefer Step 3: System will send a test notification showing they made changes	
	Step 4: User will receive notifications based on promotions, Rental start/stop dates	
ALTERNATE COURSES:		
POST-CONDITION:		

CarRentalApp_04

Author (s): **Group 2** **Date:** **7/15/15**

Version: _____

USE CASE NAME:	Inventory	**USE CASE TYPE**
USE CASE ID:	CarRentalApp_04	Abstract: ☑
PRIORITY:	High	Extension: □
INVOKED BY:	User	
PARTICIPATING ACTORS:	Manager, Sales Person	
DESCRIPTION:	How the participating actors would be able to view the inventory quantity, rental status, the location, maintain, increase, or reduce the inventory size.	
PRE-CONDITION:	none	
TYPICAL COURSE OF EVENTS:	**Step 1**: actor can view a database cars with details such as make, model, color, year, mileage and VIN. **Step 2:** actor can view the rental status and location of each and every car in the database. **Step 3:** System alerts the user if a car has reached the age or mileage in which it will require routine maintenance or expiration. **Step 4:** If a car is damaged beyond reasonable repair or does not meet the standards for age or mileage, it is then sold or scrapped **Step 5:** If the amount of inventory is below the desired threshold, more cars will be ordered.	
ALTERNATE COURSES:	none	
POST-CONDITION:	none	

CarRentalApp_05

Author (s): _____ **Date:** _____

Version: _____

USE CASE NAME:	Time and Penalty	USE CASE TYPE
USE CASE ID:	CarRentalApp_05	Abstract: ☑
PRIORITY:	Medium	Extension: □
INVOKED BY:	User	
PARTICIPATING ACTORS:	Manager	
DESCRIPTION:	Allows manger to see when a rental starts/ends and can send penalties to customers if they are late	
PRE-CONDITION:	Notifications	
TYPICAL COURSE OF EVENTS:	**Step 1**: System creates timestamps for start and end dates of rentals Step 2: System sends notifications to customer based on start/end dates Step 3: System will notify manger if end date has past and customer hasn't returned the car	
	Step 4: Manager will charge a penalty(late fee) to customer	
ALTERNATE COURSES:		
POST-CONDITION:		

CarRentalApp_06

Author (s): Group 2 **Date:** 7/15/15

Version: _____

USE CASE NAME:	Quality Assurance	USE CASE TYPE
USE CASE ID:	CarRentalApp_06	Abstract: ☑
PRIORITY:	Medium	Extension: □
INVOKED BY:	User	
PARTICIPATING ACTORS:	Manager, Sales Person, Customer	
DESCRIPTION:	Serves as a medium for the Customers to not only ensure that their experience with the company is a pleasant, but also protecting them from any issues or malpractice that might be going within the company by receiving feedback and reviews. Serves as a resource for Manager/Sales Person in order to not only rectify any issues or discrepancies a customer may have but also to make the company better by taking reviews and feedback into careful consideration	
PRE-CONDITION:	None	
TYPICAL COURSE OF EVENTS:	**Step 1**: Customers will contact Quality Assurance either after or during their experience with the company. They express either helpful feedback or complain about a discrepancy that has occurred Step 2: Manager will contact Sales person associated with specific Customer, then will take the necessary steps to rectify the situation. Step 3: After situation has been rectified, customer will leave a review based on service received. Step 4: Manger and Sales person will take gathered review information and work to improve based on information received.	
ALTERNATE COURSES:	None	
POST-CONDITION:	None	

Appendix E: Source Code for Project

The following is the source code for the AutoRenter.sql database of vehicles

```
USE [master]
GO
CREATE DATABASE [AutoRenter3]
GO
USE AutoRenter3
GO

/****************************************Cars****************************************/
CREATE PROCEDURE [dbo].[procAddCars]
   @VehicleID char(15),
   @Manufacturer varchar(20),
   @Model varchar(20),
   @Size varchar(10),
   @Class varchar(10),
   @Year int,
   @Mileage int,
   @Price decimal(12, 2),
   @Status varchar(20),  /*whether it's available for rent or it's temporarily unavailable*/
   @DateOfPurchase date
AS
BEGIN
   INSERT INTO Cars(VehicleID, Manufacturer, Model, Year, Mileage, Price, Status,
DateOfPurchase)
   VALUES (@VehicleID, @Manufacturer, @Model, @Size, @Class, @Year, @Mileage,
@Price, @Status, @DateOfPurchase)
END

GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[procDeleteCars]
   @VehicleID char(15),
   @Manufacturer varchar(20),
   @Model varchar(20),
   @Size varchar(10),
   @Class varchar(10),
   @Year int,
   @Mileage int,
   @Price decimal(12, 2),
   @Status varchar(20),
   @DateOfPurchase date
AS
BEGIN
   DELETE
   FROM Cars
   WHERE Cars.VehicleID = @VehicleID
END
```

```
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[procUpdateCars]
  @VehicleID char(15),
  @Manufacturer varchar(20),
  @Model varchar(20),
  @Size varchar(10),
  @Class varchar(10),
  @Year int,
  @Mileage int,
  @Price decimal(12, 2),
  @Status varchar(20), /*whether it's available for rent or it's temporarily unavailable*/
  @DateOfPurchase date
AS

BEGIN
  UPDATE Cars
  SET
        VehicleID = @VehicleID,
        Manufacturer = @Manufacturer,
        Model = @Model,
        Size = @Size,
        Class = @Class,
        Year = @Year,
        Mileage = @Mileage,
        Price = @Price,
        Status = @Status,
        DateOfPurchase = @DateOfPurchase
  WHERE Cars.VehicleID = @VehicleID
END

GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[procGetCarsByID]
  @VehicleID varchar(30)
AS
BEGIN
  SELECT      *
  FROM        Cars
  WHERE Cars.VehicleID = @VehicleID
END

GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
```

```
CREATE PROCEDURE [dbo].[procCountCars]
  @VehicleID varchar(30)
AS
BEGIN
  SELECT      count(*)
  FROM        Cars
  WHERE Cars.VehicleID = @VehicleID
END


GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Cars](
  [VehicleID] [char](15) NOT NULL,
  [Manufacturer] [varchar](20),
  [Model] [varchar](20),
  [Size] [varchar](10),
  [Class] [varchar](10),
  [Year] [int],
  [Mileage] [int],
  [Price] [decimal](12, 2),
  [Status] [varchar](20),
  [DateOfPurchase][date],
PRIMARY KEY CLUSTERED
(
  [VehicleID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY =
OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]


GO
INSERT [dbo].[Cars] ([VehicleID], [Manufacturer], [Model], [Size], [Class], [Year], [Mileage],
[Price], [Status]) VALUES
(N'ATL968LXW390R  ', N'Volkswagen', N'Jetta', N'Small', N'Economy', 2014, 20000,
CAST(20.00 AS Decimal(12, 2)), N'Available', N'8-Jun-2014')
GO
INSERT [dbo].[Cars] ([VehicleID], [Manufacturer], [Model], [Size], [Class], [Year], [Mileage],
[Price], [Status]) VALUES
(N'FJI820ODN744E  ', N'Ford', N'Focus', N'Small', N'Economy', 2015, 6000, CAST(20.00 AS
Decimal(12, 2)), N'Available', N'18-Jan-2015')
GO
INSERT [dbo].[Cars] ([VehicleID], [Manufacturer], [Model], [Size], [Class], [Year], [Mileage],
[Price], [Status]) VALUES
(N'QIO543UYR903C  ', N'Chevorlet', N'Impala', N'Large', N'Economy', 2015, 4423, CAST(40.00
AS Decimal(12, 2)), N'Available', N'1-Mar-2015')
GO
INSERT [dbo].[Cars] ([VehicleID], [Manufacturer], [Model], [Size], [Class], [Year], [Mileage],
[Price], [Status]) VALUES
```

```
(N'ZGH876RET890J  ', N'Audi', N'A4', N'Small', N'Luxury', 2014, 14900, CAST(40.00 AS Decimal(12, 2)), N'Rented', N'28-APR-2014')
GO
INSERT [dbo].[Cars] ([VehicleID], [Manufacturer], [Model], [Size], [Class], [Year], [Mileage], [Price], [Status]) VALUES
(N'PLM546IOP876Y  ', N'Toyota', N'Camry', N'Medium', N'Economy', 2015, 23760, CAST(30.00 AS Decimal(12, 2)), N'Rented', N'15-Feb-2015')
GO
INSERT [dbo].[Cars] ([VehicleID], [Manufacturer], [Model], [Size], [Class], [Year], [Mileage], [Price], [Status]) VALUES
(N'TGV095YHU254T  ', N'Ford', N'Fusion', N'Medium', N'Economy', 2015, 26000, CAST(30.00 AS Decimal(12, 2)), N'Available', N'28-Jan-2015')
GO
INSERT [dbo].[Cars] ([VehicleID], [Manufacturer], [Model], [Size], [Class], [Year], [Mileage], [Price], [Status]) VALUES
(N'UIN789EWR449L  ', N'Honda', N'Accord',  N'Medium', N'Economy', 2015, 7600, CAST(30.00 AS Decimal(12, 2)), N'Available', N'10-Jul-2015')
GO
INSERT [dbo].[Cars] ([VehicleID], [Manufacturer], [Model], [Size], [Class], [Year], [Mileage], [Price], [Status]) VALUES
(N'WCF906GVC880O  ', N'Honda', N'Civic',  N'Small', N'Economy', 2014, 17300, CAST(20.00 AS Decimal(12, 2)), N'Available', N'13-Oct-2014')
GO
INSERT [dbo].[Cars] ([VehicleID], [Manufacturer], [Model], [Size], [Class], [Year], [Mileage], [Price], [Status]) VALUES
(N'SDF778YTE121W  ', N'BMW', N'5-Series',  N'Medium', N'Luxury', 2015, 3900, CAST(50.00 AS Decimal(12, 2)), N'Available', N'12-May-2015')
GO
INSERT [dbo].[Cars] ([VehicleID], [Manufacturer], [Model], [Size], [Class], [Year], [Mileage], [Price], [Status]) VALUES
(N'RSP494GHM346I  ', N'Hundai', N'Equus', N'Large', N'Luxury', 2014, 6890, CAST(60.00 AS Decimal(12, 2)), N'Available', N'1-Dec-2014')
GO
INSERT [dbo].[Cars] ([VehicleID], [Manufacturer], [Model], [Size], [Class], [Year], [Mileage], [Price], [Status]) VALUES
(N'NBH324ENF399U  ', N'Toyota', N'Carolla', N'Small', N'Economy', 2015, 16000, CAST(20.00 AS Decimal(12, 2)), N'Available', N'20-Mar-2015')


/****************************************RentedCars****************************************/

GO
SET ANSI_PADDING OFF
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE PROCEDURE [dbo].[procAddRentedCars]
@VIN char(15),
@DriversLicense int,
```

```
@DateRented date,
@ReturnDate date

AS
BEGIN
  INSERT INTO Cars(VIN, DriversLicense, DateRented, ReturnDate)
  VALUES (@VIN, @DriversLicense, @DateRented, @ReturnDate)
END

GO

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[procDeleteRentedCars]
@VIN char(15),
@DriversLicense int,
@DateRented date,
@ReturnDate date

AS
BEGIN
  DELETE
  FROM RentedCars
  WHERE RentedCars.VIN = @VIN
END

GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[procUpdateRentedCars]
@VIN char(15),
@DriversLicense int,
@DateRented date,
@ReturnDate date

AS

BEGIN
  UPDATE RentedCars
  SET
        VIN = @VIN,
        DriverLicense = @DriversLicense,
        DateRented = @DateRented,
        ReturnDate = @Return
  WHERE Cars.VIN = @VIN
END

GO
SET ANSI_NULLS ON
```

```
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[procGetRentedCarsByID]
  @VIN varchar(30)
AS
BEGIN
  SELECT      *
  FROM        RentedCars
  WHERE RentedCars.VIN = @VIN
END


GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[RentedCars](
[VIN] [char](15) NOT NULL,
[DriversLicense] [int],
[DateRented] [date],
[ReturnDate] [date]
)
PRIMARY KEY CLUSTERED
(
  [VIN] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY =
OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
```

The following is the HTML source code for the Inventory

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html><head>
 <meta content="text/html; charset=ISO-8859-1" http-equiv="content-type">
 <title>Inventory</title>

 <meta name="ROBOTS" content="NOINDEX, NOFOLLOW">
 <style type="text/css">
  #tfheader{
        background-color:#white;
  }
  #tfnewsearch{
        float:right;
        padding:20px;
  }
  .tftextinput{
        margin: 0;
        padding: 5px 15px;
        font-family: Arial, Helvetica, sans-serif;
```

```
        font-size:14px;
        border:1px solid #0076a3; border-right:0px;
        border-top-left-radius: 5px 5px;
        border-bottom-left-radius: 5px 5px;
  }
  .tfbutton {
        margin: 0;
        padding: 5px 15px;
        font-family: Arial, Helvetica, sans-serif;
        font-size:14px;
        outline: none;
        cursor: pointer;
        text-align: center;
        text-decoration: none;
        color: #ffffff;
        border: solid 1px #0076a3; border-right:0px;
        background: #0095cd;
        background: -webkit-gradient(linear, left top, left bottom, from(#00adee), to(#0078a5));
        background: -moz-linear-gradient(top,  #00adee,  #0078a5);
        border-top-right-radius: 5px 5px;
        border-bottom-right-radius: 5px 5px;
  }
  .tfbutton:hover {
        text-decoration: none;
        background: #007ead;
        background: -webkit-gradient(linear, left top, left bottom, from(#0095cc), to(#00678e));
        background: -moz-linear-gradient(top,  #0095cc,  #00678e);
  }
  /* Fixes submit button height problem in Firefox */
  .tfbutton::-moz-focus-inner {
    border: 0;
  }
  .tfclear{
        clear:both;
  } </style>
</head><body style="color: rgb(0, 0, 0);" alink="#ee0000" link="#0000ee" vlink="#551a8b">
<div style="text-align: center;"><big style="font-weight: bold; text-decoration:
underline;"><big>Inventory<br>
</big></big></div>
<br>
<br>
<br>



<!--connection to the SQL server**********************************************************-->
<script>
var objConnection = new ActiveXObject("adodb.connection");
var strConn = "driver={SQL
SERVER};server=127.0.0.1\\MSSQLSERVER2012;database=AutoRenter3;uid=sa;password=it
cs3160";

objConnection.Open(strConn);
```

```
<!--recordset for Cars Table--
*******************************************************************************>
var rs = new ActiveXObject("ADODB.Recordset");
var strQuery1 = "SELECT * FROM Cars WHERE Cars.VehicleID = VehicleID";
var total = 0;
var available = 0;
var rented = 0;
var small_econ = 0;
var med_econ = 0;
var large_econ = 0;
var small_lux = 0;
var med_lux = 0;
var large_lux = 0;
rs.Open(strQuery1, objConnection);
rs.MoveFirst();
/*This loop writes all fields from the Cars table in the database into the html program, gives
values to the variables
representing the number of each subcategory of car size and class, the number of vehicles
rented and available, and the total
number of vehicles in the database*/
while (!rs.EOF)
{
        total++
        document.write(rs.fields(0) + "        ");
        document.write(rs.fields(1) +
"        ");
        document.write(rs.fields(2) + "        ");
        document.write(rs.fields(3) + "        ");
        document.write(rs.fields(4) + "        ");

        if(rs.fields(3).Value == "Small" && rs.fields(4).Value == "Economy")
  {
         small_econ++;
  }
        else if(rs.fields(3).Value == "Medium" & rs.fields(4).Value == "Economy")
  {
         med_econ++;
  }
        else if(rs.fields(3).Value == "Large" && rs.fields(4).Value == "Economy")
  {
         large_econ++;
  }
        if(rs.fields(3).Value == "Small" && rs.fields(4).Value == 'Luxury')
  {
         small_lux++;
  }
        else if(rs.fields(3).Value == "Medium" && rs.fields(4).Value == 'Luxury')
  {
         med_lux++;
  }
        else if(rs.fields(3).Value == "Large" && rs.fields(4).Value == 'Luxury')
  {
         large_lux++;
```

```
  }
        document.write(rs.fields(5) + "        ");
        document.write(rs.fields(6) + "        ");
        document.write(rs.fields(7) + "        ");
        document.write(rs.fields(8) + "        ");

        if(rs.fields(8).Value == "Available")
  {
         available++;
  }
         else if(rs.fields(8).Value == "Rented")
  {
         rented++;
  }
        document.write(rs.fields(9) + "<br/>");
        rs.movenext();
}
document.write("<br/>" + "<br/>" + "Small Economy Cars in the Inventory = " + small_econ +
"<br/>");
document.write("Midsize Economy Cars in the Inventory = " + med_econ + "<br/>");
document.write("Large Economy Cars in the Inventory = " + large_econ + "<br/>");
document.write("Small Luxury Cars in the Inventory = " + small_lux + "<br/>");
document.write("Midsize Luxury Cars in the Inventory = " + med_lux + "<br/>");
document.write("Large Luxury Cars in the Inventory = " + large_lux + "<br/>");
document.write("Inventory Total = " + total + "<br/>");
document.write("<br/>" +"Inventory Total Available= " + available + "<br/>");
document.write("Inventory Total Rented = " + rented + "<br/><br/>");
rs.Close();


var today = new Date();
var dd = today.getDate();
var mm = today.getMonth()+1; //January is 0!
var yyyy = today.getFullYear();

if(dd<10) {
        dd='0'+dd
}

if(mm<10) {
        mm='0'+mm
}

today = yyyy+'-'+mm+'-'+dd;

document.write("Rented Cars:" + "<br/>");

/*here we begin to take in the RentedCars table in the database*/
var strQuery2 = "SELECT * FROM RentedCars WHERE RentedCars.VIN = VIN";
rs.Open(strQuery2, objConnection);
rs.MoveFirst();
var count = 0;
```

```
/*This loop writes all the fields from the RentedCars table in the database and performs a count
function*/
while (!rs.EOF)
{
	document.write(rs.fields(0) + "        ");
	document.write(rs.fields(1) +
"        ");
	document.write(rs.fields(2) + "        ");
	document.write(rs.fields(3) + "<br/>");
	count++;
	rs.movenext();
}
/*
var daysInContract = new Array(count);
var daysLate = new Array(count);
var vinArray = new Array(count);
count = 0;
while (!rs.EOF)
{
	 rs.fields(0).Value;
	 vinArray(count) = rs.fields(0).Value;
	rs.fields(2).Value;
	rs.fields(3).Value;
	daysInContract[count] = rs.fields(3).Value - rs.fields(2).Value;
	if(rs.fields(3).Value > today)
	{
		daysLate(count) = today - rs.fields(3).Value;
	}
	else
	{
		daysLate(count) = 0;
	}
	count++;
	rs.movenext();
}
var paymentEstimate = new Array(count);
rs.Close();


<!--recordset for Cars Table--
*******************************************************************************>
var rs = new ActiveXObject("ADODB.Recordset");
rs.Open(strQuery1, objConnection);
count = 0;
rs.MoveFirst();
while (!rs.EOF)
{
  rs.fields(0).Value;
  rs.fields(7).Value;
  rs.fields(8).Value;
  if(rs.fields(8).Value == "Rented")
  {
```

```
        paymentEstimate(count) = daysInContract(count) * rs.fields(7).Value +
daysLate(count)*rs.fields(7).Value*2;
   }
}
*/
</script>
</body></html>
```

The following is the HTML source code for the Inventory Updater

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html><head>


 <meta content="text/html; charset=ISO-8859-1" http-equiv="content-type"><title>Inventory
Updater</title>

 <meta name="ROBOTS" content="NOINDEX, NOFOLLOW">
 <style type="text/css">
  #tfheader{
        background-color:#white;
  }
  #tfnewsearch{
        float:right;
        padding:20px;
  }
  .tftextinput{
        margin: 0;
        padding: 5px 15px;
        font-family: Arial, Helvetica, sans-serif;
        font-size:14px;
        border:1px solid #0076a3; border-right:0px;
        border-top-left-radius: 5px 5px;
        border-bottom-left-radius: 5px 5px;
  }
  .tfbutton {
        margin: 0;
        padding: 5px 15px;
        font-family: Arial, Helvetica, sans-serif;
        font-size:14px;
        outline: none;
        cursor: pointer;
        text-align: center;
        text-decoration: none;
        color: #ffffff;
        border: solid 1px #0076a3; border-right:0px;
        background: #0095cd;
        background: -webkit-gradient(linear, left top, left bottom, from(#00adee), to(#0078a5));
        background: -moz-linear-gradient(top,  #00adee,  #0078a5);
        border-top-right-radius: 5px 5px;
        border-bottom-right-radius: 5px 5px;
  }
  .tfbutton:hover {
```

```
            text-decoration: none;
            background: #007ead;
            background: -webkit-gradient(linear, left top, left bottom, from(#0095cc), to(#00678e));
            background: -moz-linear-gradient(top,  #0095cc,  #00678e);
    }
    /* Fixes submit button height problem in Firefox */
    .tfbutton::-moz-focus-inner {
      border: 0;
    }
    .tfclear{
            clear:both;
    } </style></head><body style="color: rgb(0, 0, 0);" alink="#ee0000" link="#0000ee"
vlink="#551a8b">
<div style="text-align: center;"><big style="font-weight: bold; text-decoration:
underline;"><big>Inventory Updater<br>
</big></big></div>
<br>
<br>
<br>


<script>

/*takes in the information to perform a search function*/
function getQuery()
{
    var carId = document.getElementById("carInput").value;
    getCar(carId);
}

/*takes in the information to update the manufacturer field for the car*/
function updateQuery1()
{
    var carId = document.getElementById("carInputMa").value;
    var manuf = document.getElementById("carInputManuf").value;
    updateManufacturer(carId, manuf);
}

/*takes in the information to update the model field for the car*/
function updateQuery2()
{
    var carId = document.getElementById("carInputMo").value;
    var carMod = document.getElementById("carInputModel").value;
    updateModel(carId, carMod);
}

/*takes in the information to update the size field for the car*/
function updateQuery3()
{
    var carId = document.getElementById("carInputSi").value;
    var sizes = document.getElementById("carInputSizes").value;
    updateSize(carId, sizes);
}
```

```
/*takes in the information to update the class field for the car*/
function updateQuery4()
{
   var carId = document.getElementById("carInputC").value;
   var classes = document.getElementById("carInputClasses").value;
   updateClass(carId, classes);
}

/*takes in the information to update the year field for the car*/
function updateQuery5()
{
   var carId = document.getElementById("carInputY").value;
   var modYear = document.getElementById("carInputYear").value;
   updateYear(carId, modYear);
}

/*takes in the information to update the mileage field for the car*/
function updateQuery6()
{
   var carId = document.getElementById("carInputMi").value;
   var mi = document.getElementById("carInputMiles").value;
   updateMileage(carId, mi);
}

/*takes in the information to update the price field for the car*/
function updateQuery7()
{
   var carId = document.getElementById("carInputP").value;
   var carPrice = document.getElementById("carInputPrice").value;
   updatePrice(carId, carPrice);
}

/*takes in the information to update the status field for the car*/
function updateQuery8()
{
   var carId = document.getElementById("carInputSt").value;
   var avblty = document.getElementById("carInputStatus").value;
   updateStatus(carId, avblty);
}

/*takes in the information to update the date of purchase field for the car*/
function updateQuery9()
{
   var carId = document.getElementById("carInputD").value;
   var DOP = document.getElementById("carInputDOP").value;
   updateDOP(carId, DOP);
}

/*takes in the information to delete the car from the database*/
function deleteQuery()
{
   var carId = document.getElementById("carInput2").value;
```

```
   deleteCar(carId);
}

/*takes in the information to add a new car to the database*/
function addQuery()
{
   var carId = document.getElementById("carInput3").value;
   var manuf = document.getElementById("carInput4").value;
   var carMod = document.getElementById("carInput5").value;
   var sizes = document.getElementById("carInput6").value;
   var classes = document.getElementById("carInput7").value;
   var modYear = document.getElementById("carInput8").value;
   var mi = document.getElementById("carInput9").value;
   var carPrice = document.getElementById("carInput10").value;
   var avblty = document.getElementById("carInput11").value;
   var DOP = document.getElementById("carInput12").value;
   addCar(carId, manuf, carMod, sizes, classes, modYear, mi, carPrice, avblty, DOP);
}

/*connects to the database to perform a search function for the Cars table*/
function getCar(carNo)
{
   var objConnection = new ActiveXObject("adodb.connection");
   var strConn = "driver={SQL
SERVER};server=127.0.0.1\\MSSQLSERVER2012;database=AutoRenter3;uid=sa;password=it
cs3160";

   objConnection.Open(strConn);
   <!--recordset--**********************************************************************************>
   var rs = new ActiveXObject("ADODB.Recordset");
   var strQuery = "SELECT * FROM Cars WHERE Cars.VehicleID ='" + carNo+"'";
   rs.Open(strQuery, objConnection);
   rs.MoveFirst();
   var queryResult = document.getElementById("SearchArea");
   while (!rs.EOF)
   {
                  queryResult.innerText += rs.fields(0) + "\t\t\t";
                  queryResult.innerText += rs.fields(1) + "\t\t\t";
                  queryResult.innerText += rs.fields(2) + "\t\t\t";
                  queryResult.innerText += rs.fields(3) + "\t\t\t";
                  queryResult.innerText += rs.fields(4) + "\t\t\t";
                  queryResult.innerText += rs.fields(5) + "\t\t\t";
                  queryResult.innerText += rs.fields(6) + "\t\t\t";
                  queryResult.innerText += rs.fields(7) + "\t\t\t";
                  queryResult.innerText += rs.fields(8) + "\t\t\t";
                  queryResult.innerText += rs.fields(9) + "\t\t\t";
                  rs.movenext();
   }
   rs.Close();
}

/*connects to the database to update the manufacturer field in the Cars table*/
function updateManufacturer(carNo, manu)
```

```
{
   var objConnection = new ActiveXObject("adodb.connection");
   var strConn = "driver={SQL
SERVER};server=127.0.0.1\\MSSQLSERVER2012;database=AutoRenter3;uid=sa;password=it
cs3160";

   objConnection.Open(strConn);
   <!--recordset--**********************************************************************************>
   var rs = new ActiveXObject("ADODB.Recordset");
   var strQuery2 = "UPDATE Cars SET Cars.Manufacturer ='"+ manu + "'WHERE
Cars.VehicleID ='" + carNo+"'";
   rs.Open(strQuery2, objConnection);
   alert("manufacturer has been updated");
   rs.Close();
}

/*connects to the database to update the model field in the Cars table*/
function updateModel(carNo, mod)
{
   var objConnection = new ActiveXObject("adodb.connection");
   var strConn = "driver={SQL
SERVER};server=127.0.0.1\\MSSQLSERVER2012;database=AutoRenter3;uid=sa;password=it
cs3160";

   objConnection.Open(strConn);
   <!--recordset--**********************************************************************************>
   var rs = new ActiveXObject("ADODB.Recordset");
   var strQuery3 = "UPDATE Cars SET Cars.Model ='"+ mod + "'WHERE Cars.VehicleID ='" +
carNo+"'";
   rs.Open(strQuery3, objConnection);
   alert("model has been updated");
   rs.Close();
}

/*connects to the database to update the size field in the Cars table*/
function updateSize(carNo, siz)
{
   var objConnection = new ActiveXObject("adodb.connection");
   var strConn = "driver={SQL
SERVER};server=127.0.0.1\\MSSQLSERVER2012;database=AutoRenter3;uid=sa;password=it
cs3160";

   objConnection.Open(strConn);
   <!--recordset--**********************************************************************************>
   var rs = new ActiveXObject("ADODB.Recordset");
   var strQuery4 = "UPDATE Cars SET Cars.Size ='"+ siz + "'WHERE Cars.VehicleID ='" +
carNo+"'";
   rs.Open(strQuery4, objConnection);
   alert("size has been updated");
   rs.Close();
}

/*connects to the database to update the class field in the Cars table*/
```

```
function updateClass(carNo, clas)
{
   var objConnection = new ActiveXObject("adodb.connection");
   var strConn = "driver={SQL
SERVER};server=127.0.0.1\\MSSQLSERVER2012;database=AutoRenter3;uid=sa;password=it
cs3160";

   objConnection.Open(strConn);
   <!--recordset--**********************************************************************************>
   var rs = new ActiveXObject("ADODB.Recordset");
   var strQuery5 = "UPDATE Cars SET Cars.Class ='"+ clas + "'WHERE Cars.VehicleID ='" +
carNo+"'";
   rs.Open(strQuery5, objConnection);
   alert("class has been updated");
   rs.Close();
}

/*connects to the database to update the year field in the Cars table*/
function updateYear(carNo, yr)
{
   var objConnection = new ActiveXObject("adodb.connection");
   var strConn = "driver={SQL
SERVER};server=127.0.0.1\\MSSQLSERVER2012;database=AutoRenter3;uid=sa;password=it
cs3160";

   objConnection.Open(strConn);
   <!--recordset--**********************************************************************************>
   var rs = new ActiveXObject("ADODB.Recordset");
   var strQuery6 = "UPDATE Cars SET Cars.Year ='"+ yr + "'WHERE Cars.VehicleID ='" +
carNo+"'";
   rs.Open(strQuery6, objConnection);
   alert("year has been updated");
   rs.Close();
}

/*connects to the database to update the miles field in the Cars table*/
function updateMileage(carNo, miles)
{
   var objConnection = new ActiveXObject("adodb.connection");
   var strConn = "driver={SQL
SERVER};server=127.0.0.1\\MSSQLSERVER2012;database=AutoRenter3;uid=sa;password=it
cs3160";

   objConnection.Open(strConn);
   <!--recordset--**********************************************************************************>
   var rs = new ActiveXObject("ADODB.Recordset");
   var strQuery7 = "UPDATE Cars SET Cars.Mileage ='"+ miles + "'WHERE Cars.VehicleID ='" +
carNo+"'";
   rs.Open(strQuery7, objConnection);
   alert("mileage has been updated");
   rs.Close();
}
```

```
/*connects to the database to update the price field in the Cars table*/
function updatePrice(carNo, cost)
{
  var objConnection = new ActiveXObject("adodb.connection");
  var strConn = "driver={SQL
SERVER};server=127.0.0.1\\MSSQLSERVER2012;database=AutoRenter3;uid=sa;password=it
cs3160";

  objConnection.Open(strConn);
  <!--recordset--**********************************************************************************>
  var rs = new ActiveXObject("ADODB.Recordset");
  var strQuery8 = "UPDATE Cars SET Cars.Price ='"+ cost + "'WHERE Cars.VehicleID ='" +
carNo+"'";
  rs.Open(strQuery8, objConnection);
  alert("price has been updated");
  rs.Close();
}

/*connects to the database to update the status field in the Cars table*/
function updateStatus(carNo, availability)
{
  var objConnection = new ActiveXObject("adodb.connection");
  var strConn = "driver={SQL
SERVER};server=127.0.0.1\\MSSQLSERVER2012;database=AutoRenter3;uid=sa;password=it
cs3160";

  objConnection.Open(strConn);
  <!--recordset--**********************************************************************************>
  var rs = new ActiveXObject("ADODB.Recordset");
  var strQuery9 = "UPDATE Cars SET Cars.Status ='"+ availability + "'WHERE Cars.VehicleID
='" + carNo+"'";
  rs.Open(strQuery9, objConnection);
  alert("status has been updated");
  rs.Close();
}

/*connects to the database to update the date of purchase field in the Cars table*/
function updateDOP(carNo, DOP)
{
  var objConnection = new ActiveXObject("adodb.connection");
  var strConn = "driver={SQL
SERVER};server=127.0.0.1\\MSSQLSERVER2012;database=AutoRenter3;uid=sa;password=it
cs3160";

  objConnection.Open(strConn);
  <!--recordset--**********************************************************************************>
  var rs = new ActiveXObject("ADODB.Recordset");
  var strQuery10 = "UPDATE Cars SET Cars.DateOfPurchase ='"+ DOP + "'WHERE
Cars.VehicleID ='" + carNo+"'";
  rs.Open(strQuery10, objConnection);
  alert("date of purchase has been updated");
  rs.Close();
}
```

```
/*connects to the database to delete a row from the Cars table*/
function deleteCar(carNo)
{
  var objConnection = new ActiveXObject("adodb.connection");
  var strConn = "driver={SQL
SERVER};server=127.0.0.1\\MSSQLSERVER2012;database=AutoRenter3;uid=sa;password=it
cs3160";

  objConnection.Open(strConn);
  <!--recordset--**********************************************************************************>
  var rs = new ActiveXObject("ADODB.Recordset");
  var strQueryDelete = "DELETE FROM Cars WHERE Cars.VehicleID ='" + carNo+"'";
  rs.Open(strQueryDelete, objConnection);
  alert("The car has been deleted from the system")
  rs.Close();
}

/*connects to the database to add a row to the Cars table*/
function addCar(carNo, manu, mod, siz, clas, yr, miles, cost, availability, datePurchased)
{
  var objConnection = new ActiveXObject("adodb.connection");
  var strConn = "driver={SQL
SERVER};server=127.0.0.1\\MSSQLSERVER2012;database=AutoRenter3;uid=sa;password=it
cs3160";

  objConnection.Open(strConn);
  <!--recordset--**********************************************************************************>
  var rs = new ActiveXObject("ADODB.Recordset");
  var strQueryAdd = "INSERT INTO Cars (VehicleID) VALUES('"+carNo+ "') INSERT INTO
Cars (Manufacturer) VALUES('"+manu+ "')"
  "INSERT INTO Cars (Model) VALUES('"+mod+"') INSERT INTO Cars (Size)
VALUES('"+siz+"')"
  "INSERT INTO Cars (Class) VALUES('"+clas+"') INSERT INTO Cars (Year) VALUES
('"+yr+"')"
  "INSERT INTO Cars (Mileage) VALUES('"+miles+"') INSERT INTO Cars (Price)
VALUES('"+cost+"')"
  "INSERT INTO Cars (Status) VALUES('"+availability+"') INSERT INTO Cars
(DateOfPurchase) VALUES('"+datePurchased+"')";
  rs.Open(strQueryAdd, objConnection);
  alert("The car has been added to the system")
  rs.Close();
}

</script>

<!-- CSS styles for standard search box****************************************************** -->
<hr style="width: 100%; height: 2px;"><span style="font-weight: bold;">enter VehicleID
here:</span>
<div id="tfheader">
<div style="text-align: left;">
<strong></strong><strong>
```

```
<input id="carInput" class="tftextinput" name="q" size="21" maxlength="120"
type="text"></strong>
<input value="search for car by ID" class="tfbutton" onclick="getQuery();" type="submit">
</div>
<br>
<br>
<br>
<hr style="width: 100%; height: 2px;"><span style="font-weight: bold;">enter VehicleID
here:</span>
<div id="tfheader">
<div style="text-align: left;">
<input id="carInput2" class="tftextinput" name="q" size="21" maxlength="120" type="text">
<input value="delete car" class="tfbutton" onclick="deleteQuery();" type="submit">
</div>
<br>
<br>
<br>
<hr style="width: 100%; height: 2px;"><span style="font-weight: bold;">1st row: enter
VehicleID, Manufacturer, Model, Size, Class, and Year respectively
<br>
2nd row: enter Mileage, Price, Status, and Date of Purchase respectively</span>
<div id="tfheader">
<div style="text-align: left;">
<input id="carInput3" class="tftextinput" name="q" size="21" maxlength="120" type="text">
<input id="carInput4" class="tftextinput" name="q" size="21" maxlength="120" type="text">
<input id="carInput5" class="tftextinput" name="q" size="21" maxlength="120" type="text">
<input id="carInput6" class="tftextinput" name="q" size="21" maxlength="120" type="text">
<input id="carInput7" class="tftextinput" name="q" size="21" maxlength="120" type="text">
<input id="carInput8" class="tftextinput" name="q" size="21" maxlength="120" type="text">
<input id="carInput9" class="tftextinput" name="q" size="21" maxlength="120" type="text">
<input id="carInput10" class="tftextinput" name="q" size="21" maxlength="120" type="text">
<input id="carInput11" class="tftextinput" name="q" size="21" maxlength="120" type="text">
<input id="carInput12" class="tftextinput" name="q" size="21" maxlength="120" type="text">
<input value="add car" class="tfbutton" onclick="addQuery();" type="submit">
</div>
<br>
<br>
<br>
<hr style="width: 100%; height: 2px;"><span style="font-weight: bold;">enter VehicleID
here:</span>
               <span style="font-weight:
bold;">  enter Manufacturer here:</span><br>
<div id="tfheader">
<div style="text-align: left;">
<input id="carInputMa" class="tftextinput" name="q" size="21" maxlength="120" type="text">
<input id="carInputManuf" class="tftextinput" name="q" size="21" maxlength="120" type="text">
<input value="update car manufacturer" class="tfbutton" onclick="updateQuery1();"
type="submit">
</div>
<br>
<br>
<br>
<hr style="width: 100%; height: 2px;"><span style="font-weight: bold;">enter VehicleID
here:</span>
```

```
               <span style="font-weight:
bold;">  enter Model here:</span>
<div id="tfheader">
<div style="text-align: left;">
<input id="carInputMo" class="tftextinput" name="q" size="21" maxlength="120" type="text">
<input id="carInputModel" class="tftextinput" name="q" size="21" maxlength="120" type="text">
<input value="update car model" class="tfbutton" onclick="updateQuery2();" type="submit">
</div>
<br>
<br>
<br>
<hr style="width: 100%; height: 2px;"><span style="font-weight: bold;">enter VehicleID
here:</span>
               <span style="font-weight:
bold;">  enter Size here:</span>
<div id="tfheader">
<div style="text-align: left;">
<input id="carInputSi" class="tftextinput" name="q" size="21" maxlength="120" type="text">
<input id="carInputSizes" class="tftextinput" name="q" size="21" maxlength="120" type="text">
<input value="update car size" class="tfbutton" onclick="updateQuery3();" type="submit">
</div>
<br>
<br>
<br>
<hr style="width: 100%; height: 2px;"><span style="font-weight: bold;">enter VehicleID
here:</span>
               <span style="font-weight:
bold;">  enter Class here:</span><br>
<div id="tfheader">
<div style="text-align: left;">
<input id="carInputC" class="tftextinput" name="q" size="21" maxlength="120" type="text">
<input id="carInputClasses" class="tftextinput" name="q" size="21" maxlength="120"
type="text">
<input value="update car class" class="tfbutton" onclick="updateQuery4();" type="submit">
</div>
<br>
<br>
<br>
<hr style="width: 100%; height: 2px;"><span style="font-weight: bold;">enter VehicleID
here:</span>
               <span style="font-weight:
bold;">  enter Year here:</span>
<div id="tfheader">
<div style="text-align: left;">
<input id="carInputY" class="tftextinput" name="q" size="21" maxlength="120" type="text">
<input id="carInputYear" class="tftextinput" name="q" size="21" maxlength="120" type="text">
<input value="update car year" class="tfbutton" onclick="updateQuery5();" type="submit">
</div>
<br>
<br>
<br>
<hr style="width: 100%; height: 2px;"><span style="font-weight: bold;">enter VehicleID
here:</span>
```

```
               <span style="font-weight:
bold;">  enter Mileage here:</span>
<div id="tfheader">
<div style="text-align: left;">
<input id="carInputMi" class="tftextinput" name="q" size="21" maxlength="120" type="text">
<input id="carInputMiles" class="tftextinput" name="q" size="21" maxlength="120" type="text">
<input value="update car mileage" class="tfbutton" onclick="updateQuery6();" type="submit">
</div>
<br>
<br>
<br>
<hr style="width: 100%; height: 2px;"><span style="font-weight: bold;">enter VehicleID
here:</span>
               <span style="font-weight:
bold;">  enter Price here:</span><br>
<div id="tfheader">
<div style="text-align: left;">
<input id="carInputP" class="tftextinput" name="q" size="21" maxlength="120" type="text">
<input id="carInputPrice" class="tftextinput" name="q" size="21" maxlength="120" type="text">
<input value="update car price" class="tfbutton" onclick="updateQuery7();" type="submit">
</div>
<br>
<br>
<br>
<hr style="width: 100%; height: 2px;"><span style="font-weight: bold;">enter VehicleID
here:</span>
               <span style="font-weight:
bold;">  enter Status here:</span>
<div id="tfheader">
<div style="text-align: left;">
<input id="carInputSt" class="tftextinput" name="q" size="21" maxlength="120" type="text">
<input id="carInputStatus" class="tftextinput" name="q" size="21" maxlength="120" type="text">
<input value="update car status" class="tfbutton" onclick="updateQuery8();" type="submit">
</div>
<br>
<br>
<br>
<hr style="width: 100%; height: 2px;"><span style="font-weight: bold;">enter VehicleID
here:</span>
               <span style="font-weight:
bold;">  enter Date of Purchase here:</span>
<div id="tfheader">
<div style="text-align: left;">
<input id="carInputD" class="tftextinput" name="q" size="21" maxlength="120" type="text">
<input id="carInputDOP" class="tftextinput" name="q" size="21" maxlength="120" type="text">
<input value="update car date of purchase (YYYY-MM-DD)" class="tfbutton"
onclick="updateQuery9();" type="submit">
<hr style="width: 100%; height: 2px;"><br>
<p>
<br>
</p></div>
</div>
```

```
<div id="SearchArea"></div>

</div></div></div></body></html>
```

The following is the HTML source code for the inventory of the cars that are rented already

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html><head>


 <meta content="text/html; charset=ISO-8859-1" http-equiv="content-type"><title>Rented
Inventory Updater</title>

 <meta name="ROBOTS" content="NOINDEX, NOFOLLOW">
 <style type="text/css">
  #tfheader{
        background-color:#white;
  }
  #tfnewsearch{
        float:right;
        padding:20px;
  }
  .tftextinput{
        margin: 0;
        padding: 5px 15px;
        font-family: Arial, Helvetica, sans-serif;
        font-size:14px;
        border:1px solid #0076a3; border-right:0px;
        border-top-left-radius: 5px 5px;
        border-bottom-left-radius: 5px 5px;
  }
  .tfbutton {
        margin: 0;
        padding: 5px 15px;
        font-family: Arial, Helvetica, sans-serif;
        font-size:14px;
        outline: none;
        cursor: pointer;
        text-align: center;
        text-decoration: none;
        color: #ffffff;
        border: solid 1px #0076a3; border-right:0px;
        background: #0095cd;
        background: -webkit-gradient(linear, left top, left bottom, from(#00adee), to(#0078a5));
        background: -moz-linear-gradient(top,  #00adee,  #0078a5);
        border-top-right-radius: 5px 5px;
        border-bottom-right-radius: 5px 5px;
  }
  .tfbutton:hover {
        text-decoration: none;
        background: #007ead;
```

```
        background: -webkit-gradient(linear, left top, left bottom, from(#0095cc), to(#00678e));
        background: -moz-linear-gradient(top,  #0095cc,  #00678e);
  }
  /* Fixes submit button height problem in Firefox */
  .tfbutton::-moz-focus-inner {
    border: 0;
  }
  .tfclear{
        clear:both;
  } </style></head><body style="color: rgb(0, 0, 0);" alink="#ee0000" link="#0000ee"
vlink="#551a8b">
<div style="text-align: center;"><big style="font-weight: bold; text-decoration:
underline;"><big>Rented Inventory Updater<br>
</big></big></div>
<br>
<br>
<br>

<script>

  var objConnection = new ActiveXObject("adodb.connection");
  var strConn = "driver={SQL
SERVER};server=127.0.0.1\\MSSQLSERVER2012;database=AutoRenter3;uid=sa;password=it
cs3160";

  objConnection.Open(strConn);
  var strQueryCarList = "SELECT * FROM Cars WHERE Cars.VehicleID = VehicleID";
  var rs = new ActiveXObject("ADODB.Recordset");
  rs.Open(strQueryCarList, objConnection);
  var count = 0;
  rs.MoveFirst();
  /*returns a list of all the VIN numbers of cars currently rented*/
  while (!rs.EOF)
  {
        if(rs.fields(8).Value == "Rented")
        {
              count++;
              document.write("rented vehicle ID " + count + " : " + rs.fields(0) +
"        ");
        }
        rs.movenext();
  }
  rs.Close()

/*takes in the information to perform a search function*/
function getQuery()
{
  var carId = document.getElementById("carInput").value;
  getCar(carId);
}

/*takes in the information to update the drivers license of the person renting the car in field for
the rented car*/
```

```
function updateQuery1()
{
   var carId = document.getElementById("carInputL").value;
   var license = document.getElementById("carInputLicense").value;
   updateDL(carId, license);
}

/*takes in the information to update the rent date in field for the rented car*/
function updateQuery2()
{
   var carId = document.getElementById("carInputP").value;
   var pickUp = document.getElementById("carInputPickUp").value;
   updatePU(carId, pickUp);
}

/*takes in the information to update the return date in field for the rented car*/
function updateQuery3()
{
   var carId = document.getElementById("carInputR").value;
   var carReturn = document.getElementById("carInputReturn").value;
   updateReturn(carId, carReturn);
}

/*takes in the information to delete the car from the database*/
function deleteQuery()
{
   var carId = document.getElementById("carInput2").value;
   deleteCar(carId);
}

/*takes in the information to add the car to the database*/
function addQuery()
{
   var carId = document.getElementById("carInput3").value;
   addCar(carId);
}

/*connects to the database to delete a row from the RentedCars table*/
function deleteCar(carNo)
{
   var objConnection = new ActiveXObject("adodb.connection");
   var strConn = "driver={SQL
SERVER};server=127.0.0.1\\MSSQLSERVER2012;database=AutoRenter3;uid=sa;password=it
cs3160";

   objConnection.Open(strConn);
   <!--recordset--**********************************************************************************>
   var rs = new ActiveXObject("ADODB.Recordset");
   var strQueryDelete = "DELETE FROM Cars WHERE Cars.VehicleID ='" + carNo+"'";
   rs.Open(strQueryDelete, objConnection);
   alert("The car has been deleted from the system")
   rs.Close();
}
```

```
/*connects to the database to update the drivers license field in the RentedCars table*/
function updateDL(carNo, dL)
{
   var objConnection = new ActiveXObject("adodb.connection");
   var strConn = "driver={SQL
SERVER};server=127.0.0.1\\MSSQLSERVER2012;database=AutoRenter3;uid=sa;password=it
cs3160";

   objConnection.Open(strConn);
   <!--recordset--**********************************************************************************>
   var rs = new ActiveXObject("ADODB.Recordset");
   var strQuery1 = "UPDATE RentedCars SET RentedCars.DriversLicense ='"+ dL + "'WHERE
RentedCars.VIN ='" + carNo+"'";
   rs.Open(strQuery1, objConnection);
   alert("drivers license has been updated");
   rs.Close();
}

/*connects to the database to update the date rented field in the RentedCars table*/
function updatePU(carNo, rentDate)
{
   var objConnection = new ActiveXObject("adodb.connection");
   var strConn = "driver={SQL
SERVER};server=127.0.0.1\\MSSQLSERVER2012;database=AutoRenter3;uid=sa;password=it
cs3160";

   objConnection.Open(strConn);
   <!--recordset--**********************************************************************************>
   var rs = new ActiveXObject("ADODB.Recordset");
   var strQuery2 = "UPDATE RentedCars SET RentedCars.DateRented ='"+ rentDate +
"'WHERE RentedCars.VIN ='" + carNo+"'";
   rs.Open(strQuery2, objConnection);
   alert("pick up date has been updated");
   rs.Close();
}

/*connects to the database to update the return date field in the RentedCars table*/
function updateReturn(carNo, returnDate)
{
   var objConnection = new ActiveXObject("adodb.connection");
   var strConn = "driver={SQL
SERVER};server=127.0.0.1\\MSSQLSERVER2012;database=AutoRenter3;uid=sa;password=it
cs3160";

   objConnection.Open(strConn);
   <!--recordset--**********************************************************************************>
   var rs = new ActiveXObject("ADODB.Recordset");
   var strQuery3 = "UPDATE RentedCars SET RentedCars.ReturnDate ='"+ returnDate +
"'WHERE RentedCars.VIN ='" + carNo+"'";
   rs.Open(strQuery3, objConnection);
   alert("return date has been updated");
   rs.Close();
```

```
}

/*connects to the database to add a row to the RentedCars table*/
function addCar(carNo)
{
  var objConnection = new ActiveXObject("adodb.connection");
  var strConn = "driver={SQL
SERVER};server=127.0.0.1\\MSSQLSERVER2012;database=AutoRenter3;uid=sa;password=it
cs3160";

  objConnection.Open(strConn);
  <!--recordset--**********************************************************************************>
  var rs = new ActiveXObject("ADODB.Recordset");

  var strQueryAdd = "INSERT INTO RentedCars (VIN) VALUES('"+ carNo+ "')";
  rs.Open(strQueryAdd, objConnection);
  alert("car added to the rental car database");
  rs.Close();
}


</script>


<hr style="width: 100%; height: 2px;"><span style="font-weight: bold;">enter VehicleID
here:</span>
<div id="tfheader">
<div style="text-align: left;">
<strong></strong><strong>
<input id="carInput" class="tftextinput" name="q" size="21" maxlength="120"
type="text"></strong>
<input value="search for car by ID" class="tfbutton" onclick="getQuery();" type="submit">
</div>
<br>
<br>
<br>
<hr style="width: 100%; height: 2px;"><span style="font-weight: bold;">enter VehicleID
here:</span>
               <span style="font-weight:
bold;">  enter Drivers License here:</span><br>
<div id="tfheader">
<div style="text-align: left;">
<input id="carInputL" class="tftextinput" name="q" size="21" maxlength="120" type="text">
<input id="carInputLicense" class="tftextinput" name="q" size="21" maxlength="120"
type="text">
<input value="update renter's drivers license" class="tfbutton" onclick="updateQuery1();"
type="submit">
</div>
<br>
<br>
<br>
<hr style="width: 100%; height: 2px;"><span style="font-weight: bold;">enter VehicleID
here:</span>
```

```
&amp;nbsp;&amp;nbsp;  &amp;nbsp;&amp;nbsp;  &amp;nbsp;&amp;nbsp;  &amp;nbsp;&amp;nbsp;&amp;nbsp;<span style="font-weight: bold;">  enter Pick Up Date here:</span>
<div id="tfheader">
<div style="text-align: left;">
<input id="carInputP" class="tftextinput" name="q" size="21" maxlength="120" type="text">
<input id="carInputPickUp" class="tftextinput" name="q" size="21" maxlength="120" type="text">
<input value="update rental car pick up date" class="tfbutton" onclick="updateQuery2();" type="submit">
</div>
<br>
<br>
<br>
<hr style="width: 100%; height: 2px;"><span style="font-weight: bold;">enter VehicleID here:</span>
&amp;nbsp;&amp;nbsp;  &amp;nbsp;&amp;nbsp;  &amp;nbsp;&amp;nbsp;  &amp;nbsp;&amp;nbsp;&amp;nbsp;<span style="font-weight: bold;">  enter Return Date here:</span>
<div id="tfheader">
<div style="text-align: left;">
<input id="carInputR" class="tftextinput" name="q" size="21" maxlength="120" type="text">
<input id="carInputReturn" class="tftextinput" name="q" size="21" maxlength="120" type="text">
<input value="update rental car return date" class="tfbutton" onclick="updateQuery3();" type="submit">
</div>
<br>
<br>
<br>
<div id="tfheader">
<div style="text-align: left;">
<strong></strong><strong>
<input id="carInput3" class="tftextinput" name="q" size="21" maxlength="120" type="text">
<input type=BUTTON VALUE="add rented car" onclick="addQuery();"></form>

<div id="SearchArea"></div>

</div></div></div></body></html>
```