**Car Rental System**

Software Engineering Project: Catherine Woods

Submitted By:

*William Moriarty*

*T00119587*

*TL\_KCPMM\_Y2*

Submitted on: 20 April 2015

**Table of Contents**

[**1.** **Introduction** 2](#_Toc417233352)

[**2.** **Functional Components** 4](#_Toc417233353)

[**3.** **User Requirements** 5](#_Toc417233354)

[**3.1.** **The system will oversee how car details are managed on the Car Rental System.** 5](#_Toc417233355)

[**3.2.** **The system will oversee how customer details are managed on the Car Rental System.** 5](#_Toc417233356)

[**3.3.** **The system will oversee how rental details are managed on the Car Rental System.** 5](#_Toc417233357)

[**4.** **System Requirements** 6](#_Toc417233358)

[**4.1.** **System Level Use Case Diagram** 6](#_Toc417233359)

[**4.2.** **User Case Diagrams** 7](#_Toc417233360)

[4.2.1. Add Car 7](#_Toc417233361)

[4.2.2. Amend Car 10](#_Toc417233362)

[4.2.3. List Cars 15](#_Toc417233363)

[4.2.4. Add Customer 20](#_Toc417233364)

[4.2.5. Amend Customer 23](#_Toc417233365)

[4.2.6. List Customers 28](#_Toc417233366)

[4.2.7. Rent Car 33](#_Toc417233367)

[4.2.8. Return Car 39](#_Toc417233368)

[4.2.9. List Rentals 42](#_Toc417233369)

[4.2.10. Rentals Analysis 47](#_Toc417233370)

[**5.** **System Model** 49](#_Toc417233371)

[**5.1.** **Data Flow Diagram Elements** 49](#_Toc417233372)

[**5.2.** **Level 0 DFD** 50](#_Toc417233373)

[**5.3.** **Level 1 DFD** 51](#_Toc417233374)

[**5.4.** **Level 2 DFDs** 52](#_Toc417233375)

[**6.** **Data Model** 55](#_Toc417233376)

[**7.** **Database Schema** 56](#_Toc417233377)

[**8.** **Conclusion** 57](#_Toc417233378)

# **Introduction**

This System is a proposed prototype for a Car Rental System that will manage hires for a Car Rental Company and provide basic feedback of income and rental trends.

The Car Rental System is expected to save both the details of its customers and its cars so that any rental transactions made by customers may be conveniently recorded and queried at any time.

Both customers and cars registered with the System may also be queried on demand, allowing for comprehensive management and tracking of the company’s records and assets.

The System is comprised of three main components which each have their own sub processes.

* The Cars process manages the details of vehicles that are owned by the company and that will eventually be rented by its customers.
* The Customers process manages the details of the company’s prospected customers who will have to register with the System if they would like to rent a car.
* The Rentals process manages the records of all rentals made by customers and includes details such as the date of the rental and what car was rented. Also featured is a rental analysis function, showing the average and total income of rentals over a particular period of time (month, or year).

As a prototype the System will be expected to be of high industry standards, however it will meet the essential requirements.

# **Functional Components**

*[Table of Contents](#Content)*

**Hierarchy Chart**

Car Rental System

Cars

Rentals

Customers

Add Car

Add Customer

Rent Car

Return Car

Amend Customer

Amend Car

List Rentals

List Cars

List Customers

Rentals Analysis

# **User Requirements**

[*Table of Contents*](#Content)

## **The system will oversee how car details are managed on the Car Rental System.**

* + 1. The User will be able to add a car’s details to the System.
    2. The User will be able to edit or remove a car’s details from the System.
    3. The User will receive a detailed list and will be able to query all cars in the System.

## **The system will oversee how customer details are managed on the Car Rental System.**

* + 1. The User will be able to add a customer’s details to the System
    2. The User will be able to edit or remove a customer’s details from the System.
    3. The User will receive a detailed list and will be able to query all customers in the System.

## **The system will oversee how rental details are managed on the Car Rental System.**

* + 1. The User will be able to add rental details to the System.
    2. The User will be able to record a returned car into the System.
    3. The User will receive a detailed list and will be able to query all rentals in the System.
    4. The User will be able to review Rental trends between specific dates in the System.

# **System Requirements**

[*Table of Contents*](#Content)

# **System Level Use Case Diagram**

The system requirements relating to the user requirements identified in section 3 are defined below:

**CAR RENTAL SYSTEM**

**Admin**

Administrator Queries

Staff Queries

**Staff**

Maintenance

Customer Services

**Customer**

# **User Case Diagrams**

[*Table of Contents*](#Content)

# Add Car

**Staff**

<<includes>>

<<extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Add Car** | |
| **Use Case Id** | CH 1.1 | |
| **Priority** | 1 | |
| **Source** | Cars | |
| **Primary Business Actor** | Staff | |
| **Other Participating Actors** |  | |
| **Description** | This function records the details of a car into the System. | |
| **Preconditions** | The car must be in ownership and physically stored within the company’s garage before it can be added to the System.  The same car cannot be added into the System twice. | |
| **Trigger** | The Add Car function is triggered. | |
| **Typical Scenario** | **Actor Action** | **System Response** |
| The user adds a car to the system. | **Step 1:**  The user wishes to add a car or vehicle to the database and invokes the Add Car function.  **Step 3:**  The user member enters the vehicles necessary details to complete the entry:   * Registration No * Make * Model * Colour * Year\_of\_Make * Fuel\_Type (petrol or diesel)   **Step 4:**  The user member presses the ‘Confirm’ button to confirm the entered details.  **Step 9:**  The user acknowledges the confirmation message. | **Step 2:**  The System displays the UI for adding a car.  [*Table of Contents*](#Content)  **Step 5:**  The System validates the data entered:   * All fields are required * Data types entered are valid.   (The Model field takes both string and digits,  The Registration Serial field (shown by X -  i.e. 09-KY-XXXXX) takes only digits.  All other data is input via combo boxes.  **Step 6:**  The System assigns System Date as registration date, and assigns the vehicle a status of ‘Available’, and defaults its times Rented field to zero.  **Step 7:**  The Car details are stored in the Car\_File.  **Step 8:**  The System displays a confirmation message on the interface.  **Step 10:**  The System resets the user interface. |
| **Alternate Scenarios** | **Actor Action** | **System Response**  [*Table of Contents*](#Content) |
| Invalid Data Entered | **Step 7:**  The user clicks ‘Ok’.  **Step 8:**  The user re-enters the necessary details to compete the entry. | **Step 5:**  The System validates the data entered:   * Required field is not entered. * Valid data type is not entered.   (The Model field takes both string and digits,  The Registration Serial field (shown by X -  i.e. 09-KY-XXXXX) takes only digits.  All other data is input via combo boxes.  **Step 6:**  The System displays an appropriate error message on the user interface. |
| **Conclusions** | Details for a new car are now stored in the system. | |
| **Post conditions** | The Car may now be rented by an available Customer. | |
| **Business Rules** |  | |
| **Implementation Constraints** | The format of the vehicles registration plates must conform to standards of Irish Vehicle Registration and Motor Tax Office .  i.e. 09-KY-21298 (Year-County-Serial). | |

[*Table of Contents*](#Content)

# Amend Car

**Staff**

<<includes>>

<<extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Amend Car** | |
| **Use Case Id** | CH 1.2 | |
| **Priority** | 2 | |
| **Source** | Cars | |
| **Primary Business Actor** | Staff | |
| **Other Participating Actors** |  | |
| **Description** | This function allows for the details of a car to be edited or removed (status set to ‘D’(deleted)). | |
| **Preconditions** | Records must exist in the System before they can be edited.  The Car must have a status of ‘A’(available) before it can be edited (must not be out on hire or set to ‘D’ (deleted). | |
| **Trigger** | The Amend Car function is triggered. A list of all Cars that have the status of ‘A’(available) is loaded. | |
| **Typical Scenario** | **Actor Action** | **System Response** |
| The user edits the details of a car in the System | **Step 1:**  The user wishes to edit/delete the details of a car and invokes the Amend Car function.  **Step 3:**  The user may generate an alternative list by pressing the search button and inputting any search conditions he/she would like.  The user may search by:   * Registration * Year * Make * Colour   (where Status is set to ‘A’(available)).  **Step 4:**  The user clicks on the cell in the displayed Data Grid that contains the vehicle he or she wishes to be edited.  **Step 6:**  The user selects the radio button “Update”.  **Step 7:**  The user makes any changes he or she wishes.  Only the following details may be edited:   * Colour   **Step 8:**  The user presses the update button.  **Step 12:**  The user acknowledges the confirmation message and presses the ‘Ok’ button. | **Step 2:**  The System retrieves summary details of cars which have a status of ‘A’(available) from the Car\_File and displays them as a list on the user interface.  [*Table of Contents*](#Content)  **Step 5**  The System retrieves the details of the selected car from the Car\_File and displays them in their respective text boxes.  **Step 9:**  The system validates the data entered:   * All fields are required * Valid data types are entered.   **Step 10:**  The Car\_File is now updated by the System.  **Step 11:**  The System displays a confirmation message, notifying the user that the changes have been made.    **Step 13:**  [*Table of Contents*](#Content)  The System resets the user interface. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| The user removes a car from the System | **Step 1:**  The user wishes to remove a car (set its Status to ‘D’(deleted)) from the System and invokes the Amend Car function.  **Step 3:**  The user may generate an alternative list by pressing the search button and inputting any search conditions he/she would like.  The user may search by:   * Registration * Year * Make * Colour   (where Status is set to ‘A’(available))  **Step 4:**  The user clicks on the cell in the displayed Data Grid that contains the vehicle he or she wishes to be removed.  **Step 6:**  The user selects the radio button “Delete” (if it is not already selected – radio button Delete is set to default).  **Step 7:**  The user presses the “Delete” button.  **Step 9:**  The user confirms the deletion.  **Step 12:**  The user acknowledges the confirmation message and presses the ‘Ok’ button. | **Step 2:**  The System retrieves summary details of cars which have a status of ‘A’(available) from the Car\_File and displays them as a list on the user interface.  **Step 5**  The System retrieves the details of the selected car from the Car\_File and displays them on their respective text boxes.  **Step 8:**  The System displays a message on the user interface, asking the user if he or she really wants the car to be deleted.  **Step 10:**  The System deletes the entry (sets its status to ‘D’(deleted)).  **Step 11:**  The System displays a confirmation message, notifying the user that the changes have been made.  **Step 13:**  The System resets the user interface. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| Invalid Data Entered | **Step 12:**  The user re-enters the necessary details to compete the entry. | **Step 9:**  The System validates the data entered:   * Required field is not entered. * Invalid data type is not entered.   **Step 10:**  The System displays an appropriate error message on the user interface.  **Step 11:**  The System asks the user if they wish to try again. |
| **Conclusions** | Details are updated accordingly in the Car\_File. | |
| **Post conditions** | The overwritten details are permanently lost now that they are edited or removed.  Cars that are now deleted (status set to ‘D’(deleted)) may not be amended again. | |
| **Business Rules** | Amendments may only be made to reflect physical or documented changes to the vehicle (unless deleted).  The user cannot remove a vehicle that has the status of ‘O’ (out on hire) | |
| **Implementation Constraints** | The user cannot edit the details of an entry that has previously been removed.  The user cannot remove a car that has the status of ‘O’ (out on hire). | |

# List Cars

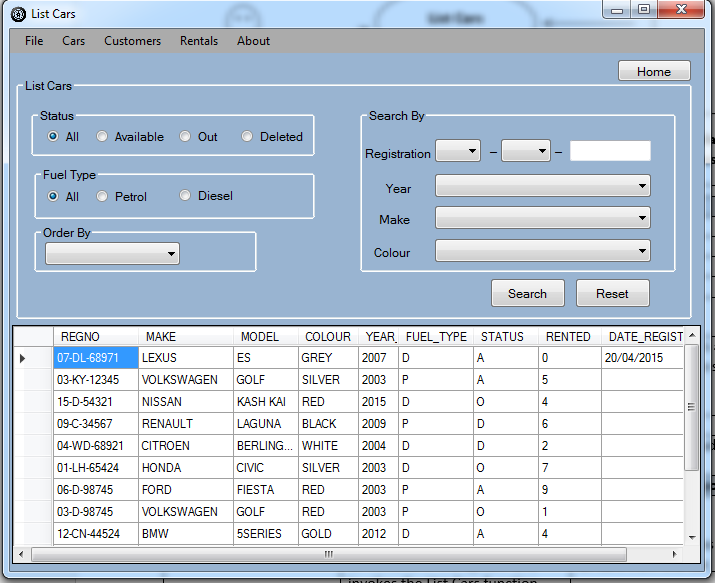
**Customer**

[*Table of Contents*](#Content)

**Staff**

<<includes>>

<<extends>>



|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **List Cars** | |
| **Use Case Id** | CH 1.3 | |
| **Priority** | 3 | |
| **Source** | Cars | |
| **Primary Business Actor** | Staff | |
| **Other Participating Actors** | Customer | |
| **Description** | This function will display a comprehensive list of all cars and their details on the user interface and will be able to retrieve specific queries on demand. | |
| **Preconditions** |  | |
| **Trigger** | The List Cars function is triggered. A list of all Cars is loaded. | |
| **Typical Scenario** | **Actor Action** | **System Response** |
| The user views a list of all Cars. | **Step 1:**  The user wishes to view a list of all vehicles in the system and invokes the List Cars function. | **Step 2:**  The System retrieves details of all cars from the Car\_File.  **Step 3:**  The System displays the car details as a list in no specific order on the user interface. Details include:   * RegistrationNo * Make * Model   [*Table of Contents*](#Content)   * Colour * Year\_of\_Make * Fuel\_Type * Times Rented * Date\_Registered * Status |
| **Alternative Scenarios** | **Actor Action** | **System Response** |
| A user needs to query a specific vehicle. | **Step 1:**  The user wishes to view the details of a specific vehicle in the system and invokes the List Cars function.  **Step 4:**  The user selects from a range of conditions for his/her query based on:   * Registration * Make * Colour * Year\_of\_Make * Status * Fuel\_Type   The user selects (if necessary) the order in which they will be listed.  **Step 5:**  The user pushes the “Search” button. | **Step 2:**  The System retrieves details of all cars from the Car\_File.  **Step 3:**  The System displays the car details as a list in no specific order on the user interface. Details include:   * RegistrationNo * Make * Model * Colour * Year\_of\_Make * Fuel\_Type * Times Rented * Date\_Registered * Status   **Step 6:**  The System validates the data entered:   * Valid data types are entered.   **Step 7:**  The System retrieves all details from the Car\_File of cars that have met the criteria of the selected query.  **Step 8:**  The System displays the list on the user interface. |
| **Alternative Scenarios** | **Actor Action** | **System Response** |
|  | **Step 9:**  The user re-enters the necessary details to compete the entry. | **Step 6**  The System validates the data entered:   * Input data is not valid.   (The Registration Serial field (shown by X -  i.e. 09-KY-XXXXX) takes only digits.  All other data is input via combo boxes.)  **Step 7:**  The System displays an appropriate error message on the user interface.  **Step 8:**  The System asks the user if they wish to try again. |
| **Conclusions** | A list of all Cars or a list for a specific query is generated for display on the user interface.  [*Table of Contents*](#Content) | |
| **Post conditions** | The user interface will remain showing the details of the query until otherwise prompted for another action. | |
| **Business Rules** | The Customer may view the staff member using this function if he wishes to select a car to rent. | |
| **Implementation Constraints** | The list is generated from a single table and does not include joins.  Date\_Registered cannot be queried. | |

# Add Customer

**Staff**

[*Table of Contents*](#Content)

**Staff**

<<includes>>

<<extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Add Customer** | |
| **Use Case Id** | CH 2.1 | |
| **Priority** | 4 | |
| **Source** | Customers | |
| **Primary Business Actor** | Staff | |
| **Other Participating Actors** | Customer | |
| **Description** | This function records the details of a customer into the System. | |
| **Preconditions** | The same customer cannot be added into the System twice. | |
| **Trigger** | The Add Customer function is triggered. | |
| **Typical Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:**  The user wishes to add a customer to the database and invokes the ‘Add Customer’ function.  **Step 3:**  The user enters the customers necessary details to compete the entry:   * Forename * Surname * Street * Town * County * Phone\_No * Email   **Step 4:**  The staff member presses the ‘Confirm’ button to confirm the entered details.  **Step 10:**  The user acknowledges the confirmation message. | **Step 2:**  The System loads the “Add Customer” form.  [*Table of Contents*](#Content)  **Step 5:**  The System validates the data entered:   * All fields are required * Valid data is entered   (The Surname and Forename fields take only String values, as does the Town filed.  Street may take a digit but only for Street numbers.  Phone No may only take digits.  Email requires valid email format (i.e. @hotmail.com))  **Step 6:**  The System determines the next Cust\_Id.  The System assigns System Date as registration date, and assigns the sets the Rental\_Out value to ‘N’.  **Step 7:**  The Customer details are stored in the Customer\_File.  **Step 9:**  The System displays a confirmation message on the interface.  **Step 11:**  The System resets the user interface. |
| **Alternate Scenarios** | **Actor Action** | **System Response**  [*Table of Contents*](#Content) |
| Invalid Data Entered | **Step 8:**  The user re-enters the necessary details to compete the entry. | **Step 5:**  The System validates the data entered:   * Required field is not entered. * Invalid data type is entered.   (The Surname and Forename fields take only String values, as does the Town filed.  Street may take a digit but only for Street numbers.  Phone No may only take digits.  Email requires valid email format (i.e. @hotmail.com))  **Step 6:**  The System displays an appropriate error message on the user interface.  **Step 7:**  The System asks the user if they wish to try again. |
| **Conclusions** | Details for a new customer are now stored in the system. | |
| **Post conditions** | The Customer may now rent an available vehicle. | |
| **Business Rules** | A customer may only be added once. | |
| **Implementation Constraints** | A foreign resident cannot be added to the system in this version. | |

# Amend Customer

[*Table of Contents*](#Content)

**Customer**

**Staff**

<<includes>>

<<extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Amend Customer** | |
| **Use Case Id** | CH 2.2 | |
| **Priority** | 5 | |
| **Source** | Customers | |
| **Primary Business Actor** | Staff | |
| **Other Participating Actors** |  | |
| **Description** | This function allows for the details of a customer to be edited or removed (status set to ‘D’(deleted)). | |
| **Preconditions** | Records of a customer must exist in the System before it can be edited or removed.  The Car must have a status of ‘A’(available) before it can be edited (must not be out on hire or set to ‘D’ (deleted). | |
| **Trigger** | The Amend Customer function is triggered. A list of all Customers that have the status of ‘A’(available) and the Rental\_Out value ‘N’ is loaded. | |
| **Typical Scenario** | **Actor Action** | **System Response** |
| The user edits the details of a customer in the System | **Step 1:**  The user wishes to edit the details of a customer and invokes the Amend Customer function.  **Step 3:**  The user may generate an alternative list by pressing the search button and inputting any search conditions he/she would like.  The user may search by:   * Cust\_Id * Forename * Surname   (where Status is set to ‘A’(available)).  **Step 4:**  The user clicks on the cell in the displayed Data Grid that contains the customer he or she wishes to be edited.  **Step 6:**  The user selects the radio button “Update”.  **Step 7:**  The user makes any changes he or she wishes.  Only the following details may be edited:   * Street * Town * County * Phone * Email   **Step 8:**  The user presses the update button.  **Step 13:**  The user acknowledges the confirmation message. | **Step 2:**  The System retrieves the details of all customers from the Customer\_File that have the status of ‘A’(available) and the Rental\_Out value ‘N’ and displays them as a list on the user interface.  [*Table of Contents*](#Content)  **Step 5:**  The System retrieves the details of the selected customer from the Customer\_File and displays them in their respective text boxes.  **Step 9:**  The system validates the data entered:   * All fields are required * Valid data types are entered.   [*Table of Contents*](#Content)  **Step 10:**  The Customer\_File is now updated by the System.  **Step 12:**  The System displays a confirmation message, notifying the user that the changes have been made.  **Step 14:**  The System resets the user interface. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| The user removes a customer from the System | **Step 1:**  The user wishes to remove a customer (set its Status to ‘D’(deleted)) from the System and invokes the Amend Customer function.  **Step 3:**  The user may generate an alternative list by pressing the search button and inputting any search conditions he/she would like.  The user may search by:   * Cust\_Id * Forename * Surname   (where Status is set to ‘A’(available)).  **Step 4:**  The user clicks on the cell in the displayed Data Grid that contains the vehicle he or she wishes to be removed.  **Step 6:**  The user selects the radio button “Delete” (if it is not already selected – radio button Delete is set to default).  **Step 7:**  The user presses the “Delete” button.  **Step 9:**  The user confirms the deletion.  **Step 12:**  The user acknowledges the confirmation message and presses the ‘Ok’ button. | **Step 2:**  The System retrieves the details of all customers from the Customer\_File that have the status of ‘A’(available) and the Rental\_Out value ‘N’ and displays them as a list on the user interface.  **Step 5**  The System retrieves the details of the selected customer from the Customer and displays them on their respective text boxes.  **Step 8:**  The System displays a message on the user interface, asking the user if he or she really wants the car to be deleted.  **Step 10:**  The System deletes the entry (sets its status to ‘D’(deleted)).  **Step 11:**  The System displays a confirmation message, notifying the user that the changes have been made.  **Step 13:**  The System resets the user interface. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| Invalid Data Entered | **Step 12:**  The user re-enters the necessary details to compete the entry. | **Step 9:**  The System validates the data entered:   * Required field is not entered. * Invalid data type is not entered.   **Step 10:**  The System displays an appropriate error message on the user interface.  **Step 11:**  The System asks the user if they wish to try again. |
| **Conclusions** | Details are updated accordingly in the Customer\_File. | |
| **Post conditions** | The overwritten details are permanently lost now that they are edited or removed.  Customers that are now deleted (status set to ‘D’(deleted)) may not be amended again. | |
| **Business Rules** | Amendments may only be made to the customer’s details after the customer themselves inform Staff of their life changes. | |
| **Implementation Constraints** | The user cannot edit the details of an entry that has previously been removed.  The user cannot remove the details of a customer who has the status of ‘Yes’ in Rental\_Out. | |

# List Customers

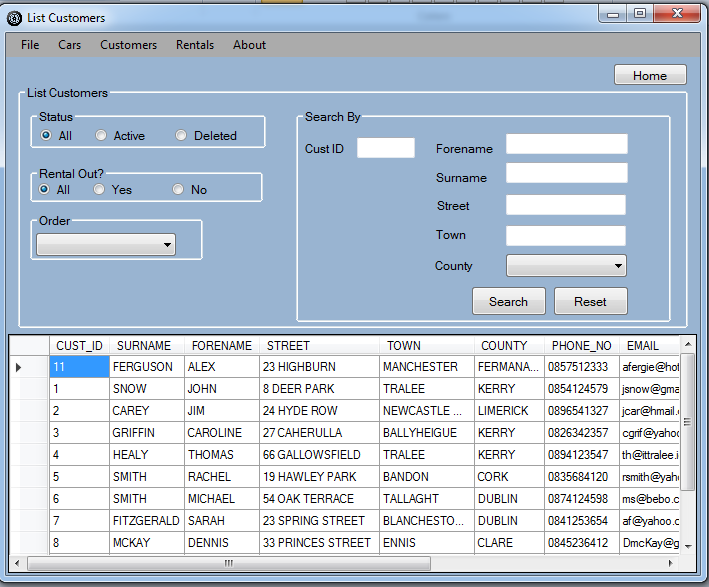
**Customer**

[*Table of Contents*](#Content)

**Staff**

<<includes>>

<<extends>>



|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **List Customers** | |
| **Use Case Id** | CH 2.3 | |
| **Priority** | 6 | |
| **Source** | Customers | |
| **Primary Business Actor** | Staff | |
| **Other Participating Actors** |  | |
| **Description** | This function will display a comprehensive list of all customers and their details on the user interface and will be able to retrieve specific queries on demand. | |
| **Preconditions** |  | |
| **Trigger** | The List Customers function is triggered. A list of all Customers is loaded. | |
| **Typical Scenario** | **Actor Action** | **System Response** |
| The user views a list of all Customers. | **Step 1:**  The user wishes to view a list of all customers in the system and invokes the List Customers function. | **Step 2:**  The System retrieves details of all customers from the Customer\_File.  **Step 3:**  The System displays the customer details as a list in no specific order on the user interface. Details include:   * Cust\_Id * Surname * Forename   [*Table of Contents*](#Content)   * Street * Town * County * Phone\_No * Email * Status * Rental\_Out * Num\_Rentals * Date\_Registered |
| **Alternative Scenarios** | **Actor Action** | **System Response**  [*Table of Contents*](#Content) |
| A user needs to query a specific customer. | **Step 1:**  The user wishes to view the details of a specific customer in the system and invokes the List Customers function.  **Step 4:**  The user selects from a range of conditions for his/her query based on:   * Cust\_Id * Surname * Forename * Street * Town * County * Status * Rental\_Out   The user selects (if necessary) the order in which they will be listed.  **Step 5:**  The user pushes the “Search” button. | **Step 2:**  The System retrieves details of all customers from the Customer\_File.  **Step 3:**  The System displays the customer details as a list in no specific order on the user interface. Details include:   * Cust\_Id * Surname * Forename * Street * Town * County * Phone\_No * Email * Status * Rental\_Out * Num\_Rentals * Date\_Registered   **Step 6:**  The System validates the data entered:   * Valid data types are entered.   **Step 7:**  The System retrieves all details from the Customer\_File of customers that have met the criteria of the selected query.  **Step 8:**  The System displays the list on the user interface. |
| **Alternative Scenarios** | **Actor Action** | **System Response** |
| Invalid data input | **Step 9:**  The user re-enters the necessary details to compete the entry. | **Step 6**  The System validates the data entered:   * Input data is not valid.   (The Forename, Surname, Town, Street, and Town fields require String only inputs. The Cust\_ID requires number only input)  **Step 7:**  The System displays an appropriate error message on the user interface.  [*Table of Contents*](#Content)  **Step 8:**  The System asks the user if they wish to try again. |
| **Conclusions** | A list of all Customers or a list for a specific query is generated for display on the user interface. | |
| **Post conditions** | The user interface will remain showing the details of the query until otherwise prompted for another action. | |
| **Business Rules** |  | |
| **Implementation Constraints** | The list is generated from a single table and does not include joins.  Date\_Registered cannot be queried. | |

# Rent Car

[*Table of Contents*](#Content)

<<extends>>

**Staff**

<<extends>>

**Customer**

**Activity Diagram – Rent Car**

(Zoom to 200% for best view)



[*Table of Contents*](#Content)

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Rent Car**  [*Table of Contents*](#Content) | |
| **Use Case Id** | CH 3.1 | |
| **Priority** | 7 | |
| **Source** | Rentals | |
| **Primary Business Actor** | Staff | |
| **Other Participating Actors** | Customer | |
| **Description** | This function records the details of a car rental into the System. | |
| **Preconditions** | The car must be in ownership and physically stored within the company’s garage before it can be hired to a customer. It must also have a status of ‘Available’ (from the Car\_File).  A customer must be registered in the System before he or she can hire a vehicle. The customer must have a Status of ‘A’(available) and have no Rental\_Out(Set to ‘N’). | |
| **Trigger** | The Rent Car function is triggered. | |
| **Typical Scenario** | **Actor Action** | **System Response** |
| A customer wishes to rent a car | **Step 1:**  A customer informs a staff member that he or she wishes to rent a car. The staff member invokes the Rent Car function.  **Step 3:**  The user pushes the Find Customer button to generate a query of all customers in the System.  **Step 5:**  The staff member locates and selects the customer who is requesting a rental from the list.  **Step 7:**  The staff decides to use the ‘Find’ button to generate a list of all cars in the System.  **Step 9:**  The staff member locates and selects the car requested for the rental from the list.  **Step 11:**  The staff member enters the rate of the rental into the Rate text field.  **Step 12:**  The staff member selects the date of the rental from the date time picker (will be set by default to current day of system time).  **Step 13:**  The staff member enters the due date of the rental’s return.  **Step 14:**  The staff member presses the ‘Confirm’ button to confirm the entered details.  **Step 21:**  The user acknowledges the confirmation message. | **Step 2:**  The System displays the UI for renting a car.  **Step 4:**  The System displays the UI for the List Customers component.  **Step 6:**  The System now closes(hides) the List Customers Component and displays the Cust\_Id of the selected customer within the Customer text field.  **Step 8:**  [*Table of Contents*](#Content)  The System displays the UI for the List Cars component.  **Step 10:**  The System now closes (hides) the List Cars component and now displays the registration of the selected car within the Car text field.  **Step 15:**  The System validates the data entered:   * All fields are required * Data types entered are valid.   **Step 16:**  The System assigns a Rent\_Id to the rental.  [*Table of Contents*](#Content)  The System assigns a ‘Condition on Return’ field with a default String value of “NOT YET RETURNED”. (This will be assigned a value when the car is returned by the ‘Return Car’ function).  The System assigns a ’Returned’ field with a value of ‘N’.  The System assigns an ’Is\_Late’ field with a value of ‘N’.  **Step 17:**  The System stores the Rental details in the Rent\_File  **Step 18:**  The System updates the Car status to ‘Unavailable’ in the Car\_File.  **Step 19:**  The System updates the Rental\_Out status to ‘Y’ for the associated Cust\_Id in the Customer\_File.  **Step 20:**  The System displays a confirmation message on the interface.  **Step 23:**  The System resets the user interface. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| Invalid Cust\_Id or details | **Step 17:**  The staff member re-enters the Cust\_Id | **Step 15:**  The System validates the Cust\_Id entered:   * Cust\_Id is not entered * Cust\_Id is not numeric   **Step 16:**  The System displays the appropriate error message on the user interface |
| **Conclusions** | Details for a new rental are recorded in the system. | |
| **Post conditions** | The Car status is updated to ‘U’(unavailable).  The Car Rented value is incremented by 1.  The Customer Rental\_Out value is set to ‘Y’.  The Car may not be rented again until it is returned.  If a car has the status of ‘Unavailable’ AFTER its return date the value of Is\_Late is changed to ‘Y’ (for ‘Yes”). | |
| **Business Rules** | Customer must be registered in the system.  Customer may not exceed limit of more than one Car rented at a time.  Car must be ‘Available’ for a rental to commence.  When a car is returned from its rental its status is amended to ‘Available’. | |
| **Implementation Constraints** | Proper validation for setting dates is not implemented. | |

[*Table of Contents*](#Content)

# Return Car

[*Table of Contents*](#Content)

**Staff**

**Customer**

<<includes>>

<<extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Return Car** | |
| **Use Case Id** | CH 3.2 | |
| **Priority** | 7 | |
| **Source** | Rentals | |
| **Primary Business Actor** | Staff | |
| **Other Participating Actors** | Customer | |
| **Description** | This function records the details of a car return into the System after a customer has returned a rented car. | |
| **Preconditions** | The Customer must return with the Rented car. | |
| **Trigger** | The Return Car function is triggered. The system generates a list of all Rentals where the Returned value is ‘N’. | |
| **Typical Scenario** | **Actor Action** | **System Response** |
| A customer wishes to rent a car | **Step 1:**  A customer informs a staff member that he or she is returning a car that has been rented. The staff member invokes the Return Car function.  **Step 3:**  The user may generate an alternative list by pressing the search button and inputting any search conditions he/she would like.  The user may search by:   * Rent\_Id * Cust\_Id * Registration No * Date From * Date Due   (where Returned is set to ‘N’)  **Step 4:**  The user clicks on the cell in the displayed Data Grid that contains the Rentals he or she wants returned.  **Step 6:**  The user selects the Date\_Returned(System Date of today by default).  **Step 7:**  The user inputs the Condition\_on\_return field.  **Step 8:**  The user presses the confirm button.  **Step 13:**  The user acknowledges the confirmation message. | **Step 2:**  The System retrieves the details of all Rentals from the Rent\_File where the Returned value is ‘N’ and displays them as a list on the user interface.  [*Table of Contents*](#Content)  **Step 5:**  The System retrieves the details of the selected Rental from the Rental\_File and displays its Rental Id in its respective text boxes.  **Step 9:**  The system validates the data entered:   * All fields are required * Valid data types are entered.   **Step 10:**  The Rent\_File is now updated by the System.  **Step 12:**  The System displays a confirmation message, notifying the user that the Car has been returned.  [*Table of Contents*](#Content)  **Step 14:**  The System resets the user interface. |
| **Conclusions** | The Rent\_File is updated with the information of a return date and whether the car was returned damaged or not.  The value of Returned is set to ‘Y’.  The Car status is updated to ‘Available’, now that it has been returned.  The Customer’s Rental\_Out status has been updated to ‘N’ now that he has returned the car and has no rentals out. | |
| **Post conditions** | A rental has been returned and is recorded.  The Car may now be rented by a customer again.  The Customer may rent another vehicle if he or she wishes. | |
| **Business Rules** | If the date returned is after the due date in the Rent\_File the rental is considered overdue and a fine may be implemented.  Minimum time for a rental is one day. | |
| **Implementation Constraints** |  | |

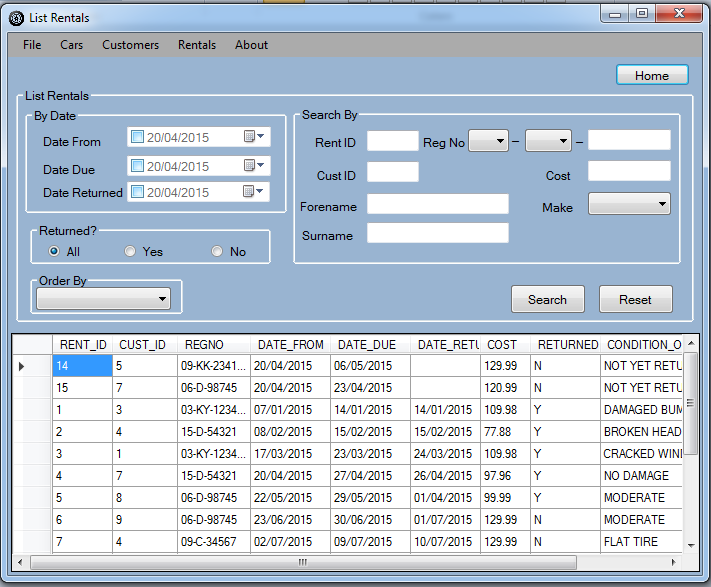
# List Rentals

[*Table of Contents*](#Content)

**Staff**

<<includes>>

<<extends>>



[*Table of Contents*](#Content)

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **List Rentals** | |
| **Use Case Id** | CH 3.3 | |
| **Priority** | 12 | |
| **Source** | Rentals | |
| **Primary Business Actor** | Staff | |
| **Other Participating Actors** |  | |
| **Description** | This function will display a comprehensive list of all rentals and their details on the user interface and will be able to retrieve specific queries on demand. | |
| **Preconditions** | Records must exist in the System before they can be listed on the user interface. | |
| **Trigger** | The List Rentals function is triggered. A list of all Rentals is loaded. | |
| **Typical Scenario** | **Actor Action** | **System Response** |
| The user views a list of all Rentals. | **Step 1:**  The user wishes to view a list of all rentals in the System and invokes the List Rentals function.  **Step 4:**  The user can now view a list of all Rentals and its details. | **Step 2:**  The System retrieves details of all Rentals from the Rent\_File.  **Step 3:**  The System displays the rental details as a list in no specific order on the user interface. Details include:   * RentId * CustId * RegNo * Date\_From * Date\_Due * Date\_Returned * Rate * Returned * Condition\_On\_Return * Is\_Late |
| **Alternative Scenario** | **Actor Action** | **System Response**  [*Table of Contents*](#Content) |
| A user needs to query a specific Rental. | **Step 1:**  The user wishes to view the details of a specific rental in the system and invokes the List Rentals function.  **Step 4:**  The user selects from a range of conditions for his/her query based on:   * RentId * CustId * RegNo * Date\_From * Date\_Due * Date\_Returned * Rate * Returned   The user selects (if necessary) the order in which they will be listed.  **Step 5:**  The user pushes the “Search” button. | **Step 2:**  The System retrieves details of all customers from the Rent\_File.  **Step 3:**  The System displays the rental details as a list in no specific order on the user interface. Details include:   * RentId * CustId * RegNo * Date\_From * Date\_Due * Date\_Returned * Rate * Returned * Condition\_On\_Return * Is\_Late   **Step 6:**  The System validates the data entered:   * Valid data types are entered.   **Step 7:**  The System retrieves all details from the Rent\_File of rentals that have met the criteria of the selected query.  **Step 8:**  The System displays the list on the user interface. |
| **Alternative Scenarios** | **Actor Action** | **System Response** |
|  | **Step 9:**  The user re-enters the necessary details to compete the entry. | **Step 6**  The System validates the data entered:   * Input data is not valid.   (The RentId, CustId, Reg Serial field and Rate fields must all take numbers.)  **Step 7:**  The System displays an appropriate error message on the user interface.  **Step 8:**  The System asks the user if they wish to try again. |
| **Conclusions** | A list of all Rentals or a list for a specific query is generated for display on the user interface. | |
| **Post conditions** | The user interface will remain showing the details of the query until otherwise prompted for another action. | |
| **Business Rules** |  | |
| **Implementation Constraints** | The list is generated from a single table and does not include joins.  Customer Names cannot be shown in queries. | |

*[Table of Contents](#Content)*

# Rentals Analysis

[*Table of Contents*](#Content)

**Admin**

<<extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Rental Analysis** | |
| **Use Case Id** | CH 3.5 | |
| **Priority** | 10 | |
| **Source** | Rentals | |
| **Primary Business Actor** | Admin | |
| **Other Participating Actors** |  | |
| **Description** | The function allows for the administrator to analysis rental trends in the system. | |
| **Preconditions** | There must be at least one rental recorded in the System to analysis rentals within the System. | |
| **Trigger** | The Rental Analysis function is triggered. | |
| **Typical Scenario** | **Actor Action** | **System Response** |
| The admin wants to analysis rental trends between two months of a year. | **Step 1:**  The Admin wishes to analyse the rental trends within the System and invokes the Rentals Analysis function.  **Step 3:**  The Admin selects what months and year he wishes to analyse income from by manipulating the combo boxes.  **Step 4:**  The Admin presses the ‘Analyse’ button. | **Step 2:**  The System displays the Rental Analysis function on the user interface. The System loads a line chart which displays income between Jan 01 and Dec 31 for the current year (default setting).  [*Table of Contents*](#Content)  **Step 5:**  The System retrieves the details of the query from the Rent\_File.  **Step 6:**  The System displays the updated Line Chart for the selected year.  The System also displays – via labels – the number of rentals, the most rented car make, the most rented car model, the average income, and the total income between the selected time frames. |
| **Conclusions** | A list of analytical information for rentals is displayed as a list on the user interface. | |
| **Post conditions** | The user interface will remain showing the details of the query until otherwise prompted for another action. | |
| **Business Rules** | This feature is primarily for the Administrator by no restrictions are made on staff to observer it. | |
| **Implementation Constraints** | Line chart will not display where values are null between certain time periods. | |

[*Table of Contents*](#Content)

# **System Model**

# **Data Flow Diagram Elements**

**External Entities:**

Customer

**Data Stores:**

D1 Car\_File

D2 Customer\_ File

D3 Rent\_File

**Processes:**

**P1 Process Cars**

P1.1 Add Car

P1.2 Edit/Remove Car

P1.3 List Cars

**P2 Process Customers**

P2.1 Add Customer

P2.2 Edit/Remove Customer

P2.3 List Customers

**P3 Process Rentals**

P3.1 Rent Car

P3.2 Return Car

P3.3 List Rentals

P3.4 Rental Analysis

# **Level 0 DFD**

[*Table of Contents*](#Content)

Rental

**Car Rental System**

**Customer**

Car

# **Level 1 DFD**

[*Table of Contents*](#Content)

**D3**

**Rent\_File**

Rental Details

**P1**

**Process**

**Cars**

**P3**

**Process**

**Rentals**

Rental Details

Car Details

Car Details

**D1**

**Car\_File**

**Customer**

Customer Details

Customer Details

Customer Details

**P2**

**Process**

**Customers**

**D2**

**Customer\_File**

# **Level 2 DFDs**

[*Table of Contents*](#Content)

* + 1. **Process Cars**

**P1.1**

**Add Car**

Car Details

Car Details

**P1.2**

**Amend Car**

Car Details

**D1**

**Car\_File**

**P1.3**

**List Car**

Car Details

* + 1. **Process Customers**

[*Table of Contents*](#Content)

**P2.1**

**Add Customer**

Customer Details

Customer Details

Customer Details

**P2.2**

**Amend Customer**

Customer Details

**Customer**

**D2**

**Customer\_File**

Customer Details

**P2.3**

**List Customers**

* + 1. **Process Rentals**

[*Table of Contents*](#Content)

**Customer**

Returns a Car

Rents a Car

Car Details

**D1**

**Car\_File**

Rental Details

Return Details

**P3.1**

**Rent Car**

Car Details

**P3.2**

**Return Car**

Rental Details

Return Details

**D2**

**Customer\_File**

Customer Details

Customer Details

Rental Details

Return Details

Rental Details

**D3**

**Rent\_File**

Rental Details

Rental Details

**P3.4**

**Rental**

**Analysis**

**P3.3**

**List Rentals**

# **Data Model**

[*Table of Contents*](#Content)

Car

Registration {pk}

Make

Model

Colour

Year\_of\_Make

Fuel\_Type

Status

Rented

Date\_Registered

**1**

**0..\***

Is for

Rental

Rent\_Id {pk}

Rate

Date\_From

Date\_Due

Date\_Returned

Returned

Condition\_On\_Return

Is\_Late

Customer

Cust\_Id {pk}

Forename

Surname

Street

Town

County

Phone

Email

Status

Rental\_Out

Num\_Rentals

Date\_Registered

**0..\***

Is rented by

**1**

# **Database Schema**

[*Table of Contents*](#Content)

**Schema:** Car Rental System

**Relation: Cars**

Attributes: Registration char(13)

Make char(10)

Model char(10)

Colour char(8)

Year\_of\_Make char(4)

Fuel\_Type char (1) { ‘P’,’D’}

Status char (1) {‘A’}

Rented numeric(3) {0}

Date\_Registered Date

Primary Key: Registration

**Relation: Customers**

Attributes: Cust\_Id numeric(5)

Forename char(15)

Surname char(25)

Street varchar(25)

Town varchar(15)

Count char(10)

Phone\_No char(10)

Email varchar (30)

Rental\_Out char(1){‘N’}

Num\_Rentals numeric(3) {0}

Date\_Registered Date

Primary Key: Cust\_Id

**Relation: Rentals**

Attributes: Rent\_Id numeric (5)

Cust\_Id numeric(5)

RegNo char(13)

Date\_From Date

Date\_Due Date

Date\_Returned Date

Rate decimal(6,2)

Returned char(1) {‘N’}

Condition\_On\_Return char (20)

Is\_Late char(1) {‘N’}

Primary Key: Rent\_Id

Foreign Key: Registration REFERENCES Cars

Foreign Key: Cust\_Id REFERENCES Customers

# **Conclusion**

[*Table of Contents*](#Content)

In conclusion to the project, there are many aspects of the System that could have been, upon reflection, improved.

Primarily, more could have been done in terms of functionality. Extra features could have been included, such as logins for staff and a special login for the administrator.

A broader Customer process could also have been implemented to help the company take more business initiative with their customers.

Such ideas that have been given consideration are:

* A Customer Analysis function.
* More detailed customer information such as date of birth, nationality, and whether they have a license to drive on Irish roads. This would give the company a clearer projection of who their target market is, and what age group they belong to, and whether they are a tourist etc
* A process for controlling automating newsletters and emailing for receipts and could also have been beneficial for promotion and communication to customers.

Other functions that have been considered:

* Staff and administrator processes to keep track of wages, bonuses, commission etc..
* A specific fine payment function under the rental process that could keep track of fines for late returns.
* A process that could keep track of damaged cars, repairs, spare parts and salvage.

Due to time constraints these features were omitted. However, the System could also be updated to include these features in future versions, providing there is time.

Throughout the duration of this project, time has been the biggest challenge. In retrospect it can be said that there has been much learned about the management of time from this project, but it will always be considered one of the biggest factors. It was important not to commit too much time to on a feature that was considered too difficult, especially if it was unnecessary to the System.

One difficulty was in approaching how the Return Car function was to be implemented, whether to make a separate file for returns and rentals or to update a rental file upon a cars return, much like amending a customer or car file. The latter was eventually decided.

As for finishing touches to the project, the interface of the prototype done in visual studio could have been much cleaner and consistent in the measurements of its windows and panes.