Software Engineering Project

CarRentSYS

Submitted By: SL

Computing with Software Development

Date Submitted: 21/04/2024

**Table of Contents**

[1. Introduction/overview 3](#_Toc152949103)

[2. Functional Components 4](#_Toc152949104)

[3. User Requirements 5](#_Toc152949105)

[3.1. CarRentSYS will manage vehicle rental rates 5](#_Toc152949106)

[3.2. CarRentSYS will manage the fleet of vehicles 5](#_Toc152949107)

[3.3. CarRentSYS will handle rental processes 5](#_Toc152949108)

[3.4. CarRentSYS will provide data analysis functionalities 5](#_Toc152949109)

[4. System Requirements 6](#_Toc152949110)

[4.1. System Level Use Case Diagram 6](#_Toc152949111)

[4.2. Manage Rates 7](#_Toc152949112)

[4.2.1. Add Vehicle Type 7](#_Toc152949113)

[4.2.2. Update Rates 9](#_Toc152949114)

[4.2.3. Validate Vehicle Type Data 11](#_Toc152949115)

[4.3. Manage Fleet 12](#_Toc152949116)

[4.3.1. Add Vehicle 12](#_Toc152949117)

[4.3.2. Update Vehicle 15](#_Toc152949118)

[4.3.3. Discontinue Vehicle 18](#_Toc152949119)

[4.3.4. Validate Vehicle Data 19](#_Toc152949120)

[4.4. Process Rentals 20](#_Toc152949121)

[4.4.1. Create Reservation 20](#_Toc152949122)

[4.4.2. Cancel Reservation 23](#_Toc152949123)

[4.4.3. Process Rental 25](#_Toc152949124)

[4.4.4. Process Return 28](#_Toc152949125)

[4.5. Perform Data Analysis 30](#_Toc152949126)

[4.5.1. Generate Yearly Vehicle Type Analysis 30](#_Toc152949127)

[4.5.2. Generate Yearly Revenue Analysis 32](#_Toc152949128)

[5. System Model 35](#_Toc152949129)

[5.1. Level-0 DFD 35](#_Toc152949130)

[5.2. Level-1 DFD 36](#_Toc152949131)

[5.3. Level-2 DFD (Process P1: Manage Rates) 37](#_Toc152949132)

[5.4. Level-2 DFD (Process P2: Manage Fleet) 38](#_Toc152949133)

[5.5. Level-2 DFD (Process P3: Process Rentals) 39](#_Toc152949134)

[5.6. Level-2 DFD (Process P4: Process Data Analysis) 40](#_Toc152949135)

[6. Data Model (Class Diagram) 41](#_Toc152949136)

[6.1. Class Diagram 41](#_Toc152949137)

[6.2. Relational Schema 42](#_Toc152949138)

[6.3. Database Schema 42](#_Toc152949139)

[7. Conclusion 44](#_Toc152949140)

# Introduction/overview

CarRentSYS is a comprehensive car rental management system designed to offer robust functionality for every aspect of vehicle rental operations. The system is structured into four modules, each tailored to specific functionalities.

In the Rates module, CarRentSYS allows addition and modification of rates for different vehicle types, ensuring that rental rates remain current and reflective of the dynamic market.

The Fleet module lets businesses manage their vehicle offerings by adding new vehicles, update existing vehicle details for accuracy, and discontinuing vehicles.

With the Rentals module, CarRentSYS simplifies the reservation process, offering customers a user-friendly platform to create or cancel reservations, while also facilitating the smooth processing of both vehicle rentals and returns.

Finally, through the Data Analysis module, CarRentSYS provides invaluable business insights with reports such as "Yearly Car Type Analysis", allowing businesses to assess the performance of different vehicle types, and "Yearly Revenue Analysis", aiding in strategic decision-making by tracking and analysing revenue trends over time.

CarRentSYS is engineered to empower rental businesses of all sizes, offering an intuitive and scalable solution to manage rates, fleets, and rentals, while providing actionable data for informed decision-making.

# Functional Components

This section presents the functional components of the proposed software system.

# User Requirements

## CarRentSYS will manage vehicle rental rates

* + 1. CarRentSYS will add a vehicle type
    2. CarRentSYS will update a vehicle type

## CarRentSYS will manage the fleet of vehicles

* + 1. CarRentSYS will add a new vehicle
    2. CarRentSYS will update the details of a vehicle
    3. CarRentSYS will discontinue a vehicle

## CarRentSYS will handle rental processes

* + 1. CarRentSYS will create reservation
    2. CarRentSYS will cancel an existing reservation
    3. CarRentSYS will process vehicle rental
    4. CarRentSYS will process return of a vehicle

## CarRentSYS will provide data analysis functionalities

* + 1. CarRentSYS will generate a yearly car type analysis
    2. CarRentSYS will generate a yearly revenue analysis

# System Requirements

The system requirements are presented in this section of the document.

The top-level modules in CarRentSYS are: Rates, Fleet, Rentals and Data Analysis.

## System Level Use Case Diagram

The following system-level use-case diagram illustrates the high-level system requirements.

Manager

Customer

Assistant

## Manage Rates

This module allows the manager to add new vehicle types and update existing types.

### Add Vehicle Type

This function allows the manager to input details for a new vehicle type such as: ECO (Economy), STD (Standard), SPT (Sport), LUX (Luxury), SUV (Sports Utility Vehicles), MVN (Minivans), GRN (Green).

<<includes>>

<<extends>>

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Add Vehicle Type | |
| **Use Case Id** | CR001 | |
| **Priority** | High | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function allows the manager to input details for a new vehicle type. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** Invoke the Add Vehicle Type function.  **Step 3:** Enter vehicle type details:   * TypeCode (TEXT 3) * Name (TEXT 15) * DailyRate (NUMERIC)   **Step 4:** Press confirmation. | **Step 2:** Display the UI.  **Step 5:** Invoke the Validate Vehicle Type Data function.  **Step 6:** Validate data:   * All fields must be entered * TypeCode must be unique * TypeCode must not be numeric * Name must not be numeric * DailyRate must be numeric and > 0   **Step 7:** Save vehicle type details to the Vehicle Types file:   * TypeCode * Name * Rate   **Step 8:** Display the confirmation message.  **Step 9:** Reset the UI. |
| **Alternate Scenarios** | **Manager** | **System** |
| **Invalid Data Input** | **Step 3:** The manager enters invalid data. | **Step 5:** Display an appropriate error message.  **Step 6:** Return to Step 3. |
| **Conclusions** | The vehicle type has been added successfully. | |
| **Post conditions** | Vehicles of this type can now be added. | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### Update Rates

This function allows the manager to modify existing rental rates for different vehicle types.

<<includes>>

<<extends>>

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Update Vehicle Type | |
| **Use Case Id** | CR002 | |
| **Priority** | High | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function allows the manager to update the details for an existing vehicle type. | |
| **Preconditions** | There is an existing vehicle type in the Vehicle Types file that the manager needs to update. | |
| **Trigger** |  | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** Invoke the Update Vehicle Type function.  **Step 4:** Select the vehicle type to update.  **Step 6:** Update required data:   * Type Name (TEXT 15) * Daily Rate (DECIMAL)   **Step 7:** Press confirmation. | **Step 2:** Retrieve Vehicle Type data from the Vehicle Types file.  **Step 3:** Display the UI.  **Step 5:** Retrieve current details for the selected vehicle type from the Vehicle Types file and display them on the UI.  **Step 8:** Invoke the Validate Vehicle Type Data function.  **Step 9:** Validate data:   * All fields must be entered * Name must not be numeric * DailyRate must be numeric and > 0   **Step 10:** Update vehicle type details in the Vehicle Types file:   * Name * DailyRate   **Step 11:** Display the Confirmation message.  **Step 12:** Reset the UI. |
| **Alternate Scenarios** | **Manager** | **System** |
| **Invalid Data Input** | **Step 6:** The manager enters invalid data. | **Step 7:** Display an appropriate error message.  **Step 8:** Return to Step 6. |
| **Validate Vehicle Data failed** |  | **Step 7:** Return to Step 6. |
| **Conclusions** | The vehicle type has been successfully updated. | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### Validate Vehicle Type Data

This function validates the Vehicle Type data input by the user.

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Validate Vehicle Type Data | |
| **Use Case Id** | CR003 | |
| **Priority** | High | |
| **Source** | System | |
| **Primary Business Actor** |  | |
| **Other Participating Actors** |  | |
| **Description** | This function validates the Vehicle Type data input by the user. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Manager** | **System** |
|  |  | **Step 1:** Invoke the Validate Vehicle Type Data function.  **Step 2:** Validate data:   * All fields must be entered * TypeCode must be unique * TypeCode must not be numeric * Name must not be numeric * DailyRate must be numeric and > 0 |
| **Alternate Scenarios** | **Manager** | **System** |
| **Invalid Data Input** |  | **Step 3:** Display an appropriate error message. |
| **Conclusions** | The vehicle type data has been successfully validated. | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

## Manage Fleet

This module allows the manager to add new vehicles, update, or discontinue them.

### Add Vehicle

This function enables the manager to input a vehicle details.

<<includes>>

Manager

<<extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Add Vehicle | |
| **Use Case Id** | CR004 | |
| **Priority** | High | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function allows the Manager to add a new vehicle to the Vehicle file. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** Invoke the Add Vehicle function.  **Step 4:** Enter new vehicle details:   * RegNum (TEXT 12)   **Step 5:** Select:   * Vehicle Type * Make * Model * Transmission * Fuel Type   **Step 6:** Press confirmation. | **Step 2:** Retrieve Vehicle Type data from the Vehicle Types file and Model data from the Models file.  **Step 3:** Display the UI.  **Step 7:** Invoke the Validate Vehicle Data function. Validate RegNum:   * Must be unique * The length must be between 8 and 9 characters * The first two characters must represent a year between 21 and 24 * The third character must be either "1" or "2" * The next 1 or 2 characters representing county code must be letters * The characters after the county code must be numeric   **Step 8:** Retrieve ModelID data from the Models file.  **Step 9:** The status is set to ‘A’ in the Vehicles file.  **Step 10:** Save vehicle details to the Vehicle file:   * RegNum * TypeCode * ModelID * Availability   **Step 11:** Display the Confirmation message.  **Step 12:** Reset the UI. |
| **Alternate Scenarios** | **Manager** | **System** |
| **Invalid data input** | **Step 4:** The manager enters invalid data. | **Step 5:** Display an appropriate error message.  **Step 7:** Return to Step 4. |
| **Validate Vehicle Data failed** |  | **Step 5:** Return to Step 4. |
| **Conclusions** | A new vehicle has been added to the Vehicles file. | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### Update Vehicle

This function allows the manager to update the details for an existing vehicle in the Vehicles file.

<<includes>>

Manager

<<extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Update Vehicle | |
| **Use Case Id** | CR005 | |
| **Priority** | High | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function allows the manager to update the details for an existing vehicle in the Vehicles file. | |
| **Preconditions** | There is an existing vehicle in the system that the manager wants to update. | |
| **Trigger** |  | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** Invoke the Update Vehicle Function.  **Step 3:** Enter the RegNum or select the vehicle from the grid.  **Step 5:** Update Vehicle Details.  Select:   * Vehicle Type * Make * Model * Transmission * Fuel Type   **Step 6:** Press confirmation. | **Step 2:** Display the UI.  **Step 4:** Retrieve current details for the vehicle and display on the UI for updating.  **Step 7:** Retrieve ModelID data from the Models file.  **Step 9:** Save vehicle details to the Vehicle file:   * TypeCode * ModelID * Trans * Fuel   **Step 10:** Display the Confirmation message.  **Step 11:** Reset the UI. |
| **Alternate Scenarios** | **Manager** | **System** |
|  |  |  |
| **Conclusions** | The vehicle has been updated in the Vehicles file. | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### Discontinue Vehicle

This function allows the manager to discontinue existing vehicles.

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Discontinue Vehicle | |
| **Use Case Id** | CR006 | |
| **Priority** | High | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function allows the manager to discontinue existing vehicles. | |
| **Preconditions** | There is an existing vehicle in the Vehicles file that the manager needs to discontinue. | |
| **Trigger** |  | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** Invoke the Discontinue Vehicle Function.  **Step 4:** Enter RegNum or select vehicle from the grid.  **Step 6:** Press “Discontinue”. | **Step 2:** Retrieve Vehicle Type data from the Vehicle Types file.  **Step 3:** Display the UI.  **Step 5:** Retrieve current details for the vehicle and display on the UI.  **Step 7:** Check if there are no existing reservations with status Reserved or Picked Up.  **Step 8:** Ask Manager to confirm decision to discontinue vehicle.  **Step 9:** Set the status to ‘D’ in the Vehicle file.  **Step 10:** Display the confirmation message.  **Step 11:** Reset the UI. |
| **Alternate Scenarios** | **Manager** | **System** |
| **There are existing reservations with status Reserved or Picked Up** |  | **Step 8:** Display the error. |
| **Conclusions** | The selected vehicles status has been set to discontinued. | |
| **Post conditions** | Once discontinued the vehicle is no longer visible for reservation. | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### Validate Vehicle Data

This function validates the Vehicle data input by the user.

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Validate Vehicle Data | |
| **Use Case Id** | CR007 | |
| **Priority** | High | |
| **Source** | System | |
| **Primary Business Actor** |  | |
| **Other Participating Actors** |  | |
| **Description** | This function validates the Vehicle data input by the user. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Manager** | **System** |
|  |  | **Step 1:** The system invokes the Validate Vehicle function.  **Step 2:** Validate RegNum:   * Must be unique * The length must be between 8 and 9 characters * The first two characters must represent a year between 21 and 24 * The third character must be either "1" or "2" * The next 1 or 2 characters representing county code must be letters * The characters after the county code must be numeric |
| **Alternate Scenarios** | **Manager** | **System** |
| **Invalid Data Input** |  | **Step 3:** Display an appropriate error message. |
| **Conclusions** | The vehicle data has been validated. | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

## Process Rentals

This module allows the manager to or assistant to create reservations, cancel reservations, process rentals and return rented vehicles.

### Create Reservation

This function enables a customer to create a reservation for a vehicle.

Staff

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Create Reservation | |
| **Use Case Id** | CR008 | |
| **Priority** | High | |
| **Source** | Customer | |
| **Primary Business Actor** | Assistant | |
| **Other Participating Actors** | Manager | |
| **Description** | This function enables Staff to create an vehicle reservation for a Customer. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Staff** | **System** |
|  | **Step 1:**  Invoke the Create Reservation function.  **Step 4:** Choose the rental dates and type of a vehicle.  **Step 7:** Choose the vehicle on the grid.  **Step 9:** Enter reservation details   * FName (TEXT 20) * SName (TEXT 20) * Email (TEXT 100) * Phone (NUMERIC)   **Step 9:** Press confirmation. | **Step 2:** Retrieve data from the Vehicle Types file.  **Step 3:** Display the UI.  **Step 5:** Retrieve information of available cars from the Vehicles and Reservations files for the specified dates and vehicle type and display available vehicles on the UI.  **Step 6:** Calculate the cost and display on the UI.  **Step 8:** Display fields for customer reservation details on the UI  **Step 10:** Validate reservation details:   * All fields must be entered * FName must not be numeric * SName must not be numeric * Email must be a valid email * Phone must be a valid phone number   **Step 11:** Assign ResID the next Primary Key value.  **Step 13:** Save data to Reservations File:   * ResID * FName * SName * Email * Phone * RegNum * ResDate * PickupDate * ReturnDate * Cost * Status   **Step 14:** Display the Confirmation message.  **Step 16:** Reset the UI. |
| **Alternate Scenarios** | **Customer** | **System** |
| **No available vehicles** |  | **Step 6:** Display appropriate error message.  **Step 7:** Return to Step 3. |
| **Invalid data input** | **Step 8:** Enter invalid data. | **Step 9:** Display appropriate error message.  **Step 10:** Return to Step 8. |
| **Conclusions** | The customer has successfully created a reservation, and the reservation details have been recorded in the Reservations file. | |
| **Post conditions** | The reservation details are stored in the Reservations file.  The system is ready to process new reservation requests. | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

|  |  |
| --- | --- |
| Staff | System |

Retrieve information of vehicle types

Invoke the Create Reservation function

Display the UI

Choose the rental dates and type of a vehicle

Retrieve information of available cars

No

Display Error Message

Available?

Yes

Choose the vehicle on the grid

Calculate cost

Enter reservation details

Display fields for reservation details

Press confirmation

Validate

Display Error Message

No

Valid?

Assign ResID

Display the success message

Save data to the database

Reset the UI

### Cancel Reservation

This function allows the customer or staff to cancel an existing reservation.

Staff

Customer

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Cancel Reservation | |
| **Use Case Id** | CR009 | |
| **Priority** | High | |
| **Source** | Staff | |
| **Primary Business Actor** | Staff | |
| **Other Participating Actors** | Customer | |
| **Description** | This function allows staff to cancel existing reservation for the customer. | |
| **Preconditions** | There is an existing reservation in the Reservations file that the customer wants to cancel. | |
| **Trigger** |  | |
| **Expected Scenario** | **Customer** | **System** |
|  | **Step 1:** Invoke the Cancel Reservation Function.  **Step 3:** Enter a ResID or select from a grid.  **Step 5:** Press cancel reservation.  **Step 7:** Press confirmation. | **Step 2:** Retrieve Reservation data from the Reservations file and display on the UI.  **Step 4:** Display reservation information on the UI.  **Step 6:** Ask confirmation.  **Step 8:** Set the status to ‘C’ (Cancelled) in the Reservations file.  **Step 9:** Display the Confirmation message.  **Step 10:** Reset the UI. |
| **Alternate Scenarios** | **Customer** | **System** |
|  |  |  |
| **Conclusions** | The selected reservation’s status has been set to cancelled. | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### Process Rental

This function allows Assistant or a Manager to process the rental of a reserved vehicle when a customer arrives at the rental location.

Customer

Staff

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Process Rental | |
| **Use Case Id** | CR010 | |
| **Priority** | High | |
| **Source** | Assistant | |
| **Primary Business Actor** | Assistant | |
| **Other Participating Actors** | Manager, Customer | |
| **Description** | This function allows an Assistant or a Manager to process the rental of a reserved vehicle when a customer arrives at the rental location. | |
| **Preconditions** | Customer has a valid reservation. | |
| **Trigger** |  | |
| **Expected Scenario** | **Assistant** | **System** |
|  | **Step 1:**  Invoke the Process Rental function.  **Step 3:** Select reservation from the grid.  **Step 5:** Enter customers driver license number.  **Step 6:** Press confirmation. | **Step 2:** Retrieve Reservations with today’s date of pick up date from the Reservations file and display on the UI.  **Step 4:** Display Reservation details information on the UI.  **Step 7:** Validate driver’s license data:   * The length must be between 5 and 9 characters * Must be alphanumeric   **Step 8:** Display the success message.  **Step 9:** Set the status to ‘P’ (Picked Up) in the Reservations file.  **Step 10:** Reset the UI. |
| **Alternate Scenarios** | **Assistant** | **System** |
| **Invalid data input** | **Step 5:** Enter invalid data. | **Step 7:** Display appropriate error message.  **Step 8:** Return to Step 5. |
| **Conclusions** | The rental process has been successfully completed. | |
| **Post conditions** | The status in the Reservations file updated to “P” (Picked Up). | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### Process Return

This function allows an Assistant or a Manager to process the return of a picked up vehicle when a customer arrives at the rental location.

Assistant

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Process Return | |
| **Use Case Id** | CR011 | |
| **Priority** | High | |
| **Source** | Assistant | |
| **Primary Business Actor** | Assistant | |
| **Other Participating Actors** | Manager | |
| **Description** | This function allows an Assistant or a Manager to process the return of a picked up vehicle when a customer arrives at the rental location. | |
| **Preconditions** | Customer has rented a vehicle and is returning it. | |
| **Trigger** |  | |
| **Expected Scenario** | **Assistant** | **System** |
|  | **Step 1:**  Invoke the Process Return function.  **Step 3:** Select reservation from the grid.  **Step 7:** Press confirmation. | **Step 2:** Retrieve Reservations with Picked Up Status from the Reservations file and display on the UI.  **Step 6:** Calculate the final charges.  **Step 4:** Display Reservation details information on the UI.  **Step 8:** Process the Payment.  **Step 9:** Display the success message.  **Step 10:** Set the status to ‘D’ (Dropped Off) in the Reservations file.  **Step 11:** Reset the UI. |
| **Alternate Scenarios** | **Assistant** | **System** |
| **Conclusions** | The return process has been successfully completed. | |
| **Post conditions** | Status in the Reservations file updated to “D” (Dropped Off) | |
| **Business Rules** | Late return fees are calculated based on predefined rates and the number of late days. | |
| **Implementation Constraints** |  | |

## Perform Data Analysis

### Generate Yearly Vehicle Type Analysis

This function allows the manager to generate an annual vehicle type analysis.

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Generate Yearly Vehicle Type Analysis** | |
| **Use Case Id** | CR012 | |
| **Priority** | Low | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function allows the manager to generate an annual vehicle type analysis. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** Invokes the Generate Yearly Vehicle Type Analysis function.  **Step 3:** Select the year to generate an analysis for. | **Step 2:** Retrieve reservation data from the Reservations file for the current year.  **Step 4:** Group reservation data by vehicle types for a year.  **Step 5:** Display the pie chart on the UI, showing the most rented vehicle type. |
| **Alternate Scenarios** | **Manager** | **System Response** |
|  |  |  |
| **Conclusions** | Yearly Vehicle Type Analysis is generated and displayed. | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

A screenshot of a graph

Description automatically generated

Figure . Sample Yearly Vehicle Type Analysis

### Generate Yearly Revenue Analysis

This function allows the manager to generate yearly revenue analysis.

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Generate Yearly Revenue Analysis** | |
| **Use Case Id** | CR013 | |
| **Priority** | Low | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function allows the manager to generate an annual revenue analysis. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** Invokes the Generate Yearly Revenue Analysis function.  **Step 3:** Select the year to generate an analysis for. | **Step 2:** Retrieve Reservation data from the Reservations file.  **Step 4:** Generate Yearly Revenue Analysis for each month of the year.  **Step 5:** Display Yearly Revenue Analysis on the grid. |
| **Alternate Scenarios** | **Manager** | **System Response** |
|  |  |  |
| **Conclusions** | Annual Revenue Analysis is generated and displayed. | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

A screenshot of a graph

Description automatically generated

Figure . Sample Yearly Revenue Analysis

# System Model

The following dataflow diagrams have been produced for the system:

## Level-0 DFD

Invoice

CarRentSYS

Customer

Reservation Details

## Level-1 DFD

Reservation Details

Customer

Rate Details

Rate Details

Process Rentals

P3

P1

Reservation Confirmation

Manage

Rates

Rates File

Reservation Details

Vehicle

Reservations File

Rate Details

v

Vehicles File

Reservation Details

Vehicle Details

Vehicle Type Details

Process Data Analysis

P2

P4

Manage

Fleet

Vehicle Type Details

Rates File

## Level-2 DFD (Process P1: Manage Rates)

Update Vehicle Type

P1.2

P1.1

Add Vehicle Type

Vehicle Type Details

Vehicle Type Details

Vehicle Type Details

Rates File

## Level-2 DFD (Process P2: Manage Fleet)

Vehicle Type Details

Vehicle Type Details

Rates File

P2.1

Add Vehicle

Vehicle Details

Vehicles File

P2.2

Update Vehicle

Vehicle Details

Vehicle Details

Vehicle Details

Vehicle Details

P2.3

Discontinue Vehicle

## Level-2 DFD (Process P3: Process Rentals)

P4.2

Cancel Reservation

Reservation Details

Reservation Details

Customer

Refund

Reservation ID

Reservation ID

Vehicle Type Details

Reservation

Details

Rates File

P4.1

Create Reservation

Vehicles File

Reservation Details

Vehicle Status

Reservation Details

Reservations File

Reservation Details

P4.4

Process Return

Reservation

Status

Reservation

Status

Vehicle Details

Reservation

Details

Vehicles File

Receipt

Key

Reservation ID

P4.3

Process Rental

Vehicle Status

Customer

Key

## Level-2 DFD (Process P4: Process Data Analysis)

Vehicle Type Details

P4.1

Yearly Car Type Analysis

Reservation Details

Reservation Details

Rates File

P4.2

Yearly Revenue Analysis

Reservations File

# Data Model (Class Diagram)

This data model includes a UML class diagram, a relational schema, and a database scheme for the car rental system.

|  |
| --- |
| **Model** |
| * \_make: string (15) * \_model: string (15) |
| + GetModelID(string, string): int  + GetAllMakes(): DataSet  + GetAllModels(string): DataSet |

## Class Diagram

1

has

1..\*

|  |
| --- |
| **Vehicle** |
| * RegNum: char (12) * TypeCode: char (3) * ModelID: numeric * Avail: char (1) |
| + RegNumExists(string): bool  + HasReservedOrPickedUpReservations(string): bool  + GetAllAvailableVehicles(): DataTable  + GetVehicleDetails(string): DataTable  + AddVehicle(): void  + UpdateVehicle(): void  + DiscontinueVehicle(string): void  + GetAvailableVehiclesForType(string, string, string): DataTable |

|  |
| --- |
| **Reservation** |
| * ResID: int (5) * FName: string (20) * SName: string (20) * Email: string (100) * Phone: string (12) * RegNum: string (12) * ResDate: DateTime * PickupDate: DateTime * ReturnDate: DateTime * ActReturnDate: DateTime * Cost: decimal * Status: char (1) * License: string (20) |
| + GetNextResID(): int  + CalculateCost(VehicleType, int): decimal  + CreateReservation(): void  + CancelReservation(int): void  + GetTodaysPickedUpReservations(): DataTable  + GetReservationsWithReservedVehicles(): DataTable  + GetReservationsWithPickedUpVehicles(): DataTable  + AddDriverLicense(int, string): void  + ChangeReservationStatusToPickedUp(int): void  + CalculatePenalty(int): decimal  + ProcessReturn(int, DateTime): void  + GetReservationYears(): DataSet  + GetYearlyRevenueData(int): DataSet  + GetYearlyVehicleTypeData(int): DataSet |

for

1

0..\*

0..\*

has

1

|  |
| --- |
| **Rate** |
| * TypeCode: char (3) * Name: char (15) * DailyRate: numeric |
| + TypeCodeExists(string): bool  + GetAllVehicleTypes(): DataSet  + AddVehicleType(): void  + UpdateVehicleType(): void  + GetVehicleTypeByCode(string): VehicleType |

## Relational Schema

Relational schema for the data requirements.

Rates (TypeCode, Name, DailyRate)

Models (ModelID, Make, Model)

Vehicles (RegNum, TypeCode, Trans, Fuel, ModelID, Avail)

Reservation (ResID, FName, SName, Email, Phone, RegNum, ResDate, PickupDate, ReturnDate, ActReturnDate, Cost, Status, License)

## Database Schema

A definition of the database to be implemented.

This includes primary key, foreign key and other constraints to be implemented.

**Relation Rates**

**Attributes:**

TypeCode: char (3) NOT NULL UNIQUE

Name: varchar2 (15) NOT NULL

DailyRate: number (5,2) NOT NULL

**Primary Key:** TypeCode

**Relation Models**

**Attributes:**

ModelID: int (3) NOT NULL UNIQUE

Make: varchar2 (15) NOT NULL

Model: varchar2 (15) NOT NULL

**Primary Key:** ModelID

**Relation Vehicles**

**Attributes:**

RegNum: varchar2 (12) UNIQUE

ModelID: int (3) NOT NULL

TypeCode: char (3)

Trans: char (1) CHECK (Trans IN ('M', 'A')

Fuel: char (1) CHECK (Fuel IN ('P', 'D', 'H', 'E')

Avail: char (1)

**Primary Key:** RegNum

**Foreign Key:** TypeCode **References** Rates

**Foreign Key:** ModelID **References** Models

**Relation Reservations**

**Attributes:**

ResID: number (5) NOT NULL

FName: varchar2 (20) NOT NULL

SName: varchar2 (20) NOT NULL

Email: varchar2 (100) NOT NULL

Phone: number NOT NULL

RegNum: varchar2 (12) NOT NULL

ResDate: date NOT NULL

PickupDate: date NOT NULL

ReturnDate: date NOT NULL

ActReturnDate: date

Cost: decimal NOT NULL

Status: char(1) NOT NULL

License: varchar2 (10)

**Primary Key:** ResID

**Foreign Key:** RegNum **References** Vehicles

# Conclusion

The development of this CarRentSYS requirements document has delineated the essential processes and functionalities needed for the system's creation. By constructing detailed use-case narratives for each function, I've outlined how user interfaces can be designed, what data will be presented to users, and what data will be stored. This approach was crucial in understanding the data storage requirements, informing the creation of the class diagram, relational schema, and database schema associated with CarRentSYS. Simultaneously, a prototype of the system was developed, providing a practical glimpse into how CarRentSYS might operate upon completion. This concurrent development with the requirements document proved immensely beneficial, shedding light on potential operational issues and allowing for real-time adjustments to enhance user experience.