**Task**

**Screen/Page**

**entity**

**input field**

***Button***

‘$session variable’

**Relationship**

**Table of Contents:**

**Task Decomposition with Abstract Code:**

[Search page/Display Main Menu](#_i0ogo8skan04)

[View Vehicle Details](#_saqzw13fp1tr)

[Login](#_g0xqcagl2tso)

[Sales order](#_o2z30i6b6e7d)

[Repair](#_1hz99rwds4mi)

[Sales by Color](#_x9xztswvlfne)

[Sales by Type](#_x9xztswvlfne)

[Sales by Manufacturer](#_l2da8golj1n1)

[Gross Customer Income](#_2clos2apolcn)

[View Drill-Down Customers](#_dqkew8iy8laz)

[Repairs by Manufacturer/Type/Model](#_w2c3zyugh6gm)

[View Drill-Down Manufacturer](#_ikvgofdujxa3)

[Below Cost Sales](#_euh19qn8p1xo)

[Average Time in Inventory](#_hfmeex3y5mau)

[Parts Statistics](#_9swvts5mj3b)

[Monthly Sales](#_58nienjvv55z)

[Monthly Sales Drill-Down](#_wugpz6pmnb09)

# **Searching for vehicles/Display Main Menu**

Abstract Code

* (Initial state of the application and upon ***Return to Main Menu*** button clicks.)
* Clear all input fields
* Disable additional features unless the user is already login(‘$UserType’ is not null or anonymous ).
* Show ***search*** and ***login*** button on the **Main Menu screen**
* Query for information of counts of **Vehicle** using Vehicle.VIN, show this information as the total number of vehicles available for purchase in the system in the **Main Menu screen**
* Find available choices of searching input fields (Vehicle type, using **Vehicle**.type, Manufacturer, using **Vehicle**.Manufacturer, Model year, using **Vehicle**.Model and **Vehicle**.Year, Color, using **Vehicle**.Color) and fill available choices in the dropdowns for input fields in the **Main Menu screen**
* All users are able to fill in **List price** and **Keyword** input fields. Users also are able to select a **vehicle type**/**Manufacturer**/**Model**/**Year**/**Color/VIN(if enabled)** from the dropdown list in each input field in the **Main Menu screen**
* Upon:
  + Click the ***search*** button:
    - Read **Type**, **manufacturer**, **model**, **year**, **color,** **list price**, **keyword, VIN(if enabled)** input fields from the **Main Menu screen**
    - if data is valid and if there are **Vehicle** that matches the search criteria:
      * Display the matching **Vehicle** as a list sorted by **Vehicle**.VIN in ascending order on **Main Menu screen**
      * Session variable ‘$vehiclevin’ = Vehicle.VIN
      * Users are allowed to select an individual result from the list. If the user selects a **Vehicle**:
        + Jump to the **View Vehicle Details** task.
    - Otherwise, display the message “Sorry, it looks like we don’t have that in stock!” on the **Main Menu screen**
  + Click the ***login*** button: Jump to the **Login** task:
    - If ‘$UserType’ returned:
      * ‘$UserType’==’Inventory clerks’, enable ***Add Vehicle*** button and **new vehicle form** on **Main Menu screen**
      * ‘$UserType’==’Salespeople’, enable input field **VIN** in searching criteria on **Main Menu screen**
      * ‘$UserType’==’Service Writer’, enable ***repair form*** button on **Main Menu screen**
      * ‘$UserType’==’Managers’, enable ***report*** button, input field **VIN** in searching criteria, option to filter by soldvehicles, unsold vehicles, or all vehicles, Dropdown on **Main Menu screen**
      * ‘$UserType’==’Roland Around’, enable all the features on **Main Menu screen**
  + Click the ***Add Vehicle*** button:
    - User will fill the **VIN**, **vehicle type**, **invoice price**, etc., along with the **date** it was added to inventory in **new vehicle form** on **Main Menu screen**.
    - Read those values from input fields and if Data is valid and **VIN** does not already exist as a Vehicle.VIN:
      * Inert new **Vehicle** instance with those values, then clear any success/error message, display a success message, ‘$vehiclevin’ = **VIN** and call the **View Vehicle Details** task.
  + Click the ***repair form*** button: Jump to the **Repair** task
  + Click the ***report*** button: Read the choice from the report dropdown menu on **Main Menu screen** then call the corresponding report task

# **View Vehicle Details**

Abstract Code

* User selected on Vehicle from the list on the **Main Menu screen**
* Display **Detail Page** Screen
* Enable link to ***sell the vehicle*** on **Detail Page** Screen if ‘$UserType’==’Salespoeple’
* Query for information about the **Vehicle** and it’s details using ‘$vehiclevin’ from the HTTP Session/Cookie.:
  + Display ‘$vehiclevin’ on the **Detail Page**
  + Find and display the **Vehicle**.Type on the **Detail Page**
  + Find and display the attributes of the **Vehicle**.Type on the **Detail Page**
  + Find and display the **Vehicle**.Model and Vehicle.Year on the **Detail Page**
  + Find and display the **Vehicle**.Manufacturer on the **Detail Page**
  + Find and display the **Vehicle**.Color on the **Detail Page**
  + Find **Vehicle**.InvoicePrice and display invoice price times 125% as list price on the **Detail Page**
  + Find and display the **Vehicle**.VDescription on the **Detail Page**
  + If ‘$UserType’==‘Inventory clerks’:
    - Find and display the **Vehicle**.invoiceprice on the **Detail Page**
* Upon:
  + Click ***sell the vehicle*** link: Call the **sales order** task.
  + Click ***Return to Main Menu*** button: Call the **Display Main Menu** task.

###### **Login**

Abstract Code

* User click ***login*** button on the **Main Menu screen**
* Display **Login form** Screen
* User enters username, password input fields.
* If data validation is successful for both username and password, then:
  + When ***Enter*** button is clicked:
    - If User record is found but password not match **User**.password:
      * Go back to **Login form**, with error message.
    - else:
      * Find the User using combination of User.username and User.password
      * Find the User.Type of the User and Store login information as session variable ‘$UserType’
      * Call the Display Main Menu task with ‘$UserType’.
    - Else username and password input fields are invalid, display Login form, with error message.

### **Sales order**



Abstract Code:

* user click ***sell the vehicle*** on **Detail Page**
* **Sales order Form** is Displayed.
* user will fill the customer profile inputs field showed on **Sales order Form**:
  + if customer is an individual: fill their **first** and **last names**, along with their **driver’s license number**
  + if customer is a business: fill the business’ **tax identification number** and **business name**, along with the **name of a primary contact** and their **title**
* user will fill the transaction detail field including: **Vehicle’s VIN**, **sold price**, **sold date**
* upon:
  + click the ***lookup*** button on **Sales order Form**: run the **lookup customer** task by query customer with **driver’s license number or tax identification number**
    - if a customer is not found:
      * Read customer profile input fields, and call **Add customer** task.
  + click the confirm sale button on **Sales order Form**: run the **confirm sale** task by query vehicle’s invoice price by reading and using **Vehicle’s VIN**
    - if **sold price** is less than or equal to 95% of **Vehicle**.invoiceprice:
      * Display error massage of rejecting sale.
    - Otherwise, insert new **sale** instance with those values including **sold price**, **sold date**,customer **driver’s license number** or **tax identification number**, **Vehicle’s VIN**, **SalesPerson’s Name.** Display a success message.
  + Click ***Return to Main Menu*** button: Call the **Display Main Menu** task.

###### **Repair**

Abstract Code

* User click ***repair form***  button on the **Main Menu screen**
* Partial **repair form** is displayed.
* User will fill the **VIN** input field.
* If data validation is successful for **VIN**, then:
  + When ***Enter*** button is clicked: run **search vin** task.
    - If the **VIN** does not match a **Vehicle**.VIN in the database:
      * Display an error message
    - Otherwise, the rest of the **repair form** will be displayed:
      * Run **View Vehicle Details** task, the results will be displayed on **repair form screen**
      * Check if the **Vehicle** is associated with a **repair** order.
        + If no repairs are open for the **Vehicle**:

display **add repair button**

run **add repair** task after user click ***add repair***

button

Insert new **repair** order instance with **odometer reading** input field filled by user

click the ***lookup*** button on **Sales order Form**: run the **lookup customer** task by query customer with **driver’s license number or tax identification number**

if a customer is not found:

Read customer profile input fields, and call **Add customer** task. add the customer to the system for repair order

otherwise, add the customer to the system for repair order

User are allowed to fill the inputs field of **labor charge** and **parts**:

input field **quantity**, **vendor**, **part number**, **price** can be filled by user.

If Data is valid and user click ***add parts*** button, insert new parts instance with those values with the service writer’s name, display a success message, Otherwise, display an appropriate error message. run **add part task to** add the part to current **repair** order

if ***add labor charge*** button is clicked, add the labor charge to current **repair** order

* + - * + Otherwise, upon:

click ***updating labor charges*** button: read input field labor charges and run **update repair** task

click ***adding parts*** button: read input field for parts and run **add part** task

click ***completing*** button: add **repair**.completiondate to current **repair** as current date

* Click ***Return to Main Menu*** button: Call the **Display Main Menu** task.

### **Sales by Color**

Abstract Code

* Called from the **Main Menu screen**
* Query for information about each **Sale**, find the **Sale**.PurchaseDate and corrsponding sold **Vehicle**.color using **Vehicle**.VIN. **Sale** and **Vehicle** are related by **transfers ownership of** relationship. Count the number of **Sale** based on the different periods (**Sale**.PurchaseDate) and group the count by different colors. Put count and color data in a table:
  + Each color is one row.
  + Columns are the count of sales.
  + Columns including sales in the previous 30 days, sales in the previous year, sales overall time.
  + If a color does not have any sales, it is shown with a value of “0”.
* Display the table in the **Sales by Color report**
* Click ***Return to Main Menu*** button: Call the **Display Main Menu** task.

### 

### **Sales by Type**

Abstract Code

* Called from the **Main Menu screen**
* Query for information about each **Sale**, find the **Sale**.PurchaseDate and corrsponding sold **Vehicle**.type using **Vehicle**.VIN. **Sale** and **Vehicle** are related by **transfers ownership of** relationship. Count the number of **Sale** based on the different periods (**Sale**.PurchaseDate) and group the count by different types. Put count and type data in a table:
  + Each type is one row.
  + Columns are the count of sales.
  + Columns including sales in the previous 30 days, sales in the previous year, sales overall time.
  + If a type does not have any sales, it is shown with a value of “0”.
* Display the table in the **Sales by Type report**
* Click ***Return to Main Menu*** button: Call the **Display Main Menu** task.

### **Sales by Manufacturer**

Abstract Code

* Called from the **Main Menu screen**
* Query for information about each **Sale**, find the **Sale**.PurchaseDate and corrsponding sold **Vehicle**.Manufacturer using **Vehicle**.VIN. **Sale** and **Vehicle** are related by **transfers ownership of** relationship. Count the number of **Sale** based on the different periods (**Sale**.PurchaseDate) and group the count by different Manufacturers. Put count and Manufacturer data in a table:
  + Each Manufacturer is one row.
  + Columns are the count of sales.
  + Columns including sales in the previous 30 days, sales in the previous year, sales overall time.
  + If a type does not have any sales, it will not be put on the table.
* Display the table in the **Sales by Type report**
* Click ***Return to Main Menu*** button: Call the **Display Main Menu** task.

### **Gross Customer Income**

Abstract Code

* Called from the **Main Menu screen**
* Query for information about all **Sale** and **Repair**, group **Sale** and **Repair** the by **Customer**’s driver’s license number or tax identification number. And sum **Sale**.soldprice and **Repair** total cost as gross income for each **Customer** ID. Both **Sale** and **Repair** has relationship with **Customer**. Sort Customer ID by gross income and keep the largest 15 one. Find and Place all following data for each one of 15 Customer in a list:
  + - **Customer**’s name
    - The date of the first sale or repair start date
    - the date of the most recent sale or repair start date
    - The number of sales
    - The number of repairs
    - the Gross income
  + The list of **customer**s will be by gross income descending and by last sale/repair start date descending.
  + Display the list in the **Gross Customer Income report**
  + User are able to select one ***customer’s name*** in the list. If User click one of the ***customer’s name***: Jump to **View Drill-Down** task with selected **Customer.** and their **sale** and **repair** which can be get from the table that group **sale** and **repair** by customer.
* Click ***Return to Main Menu*** button: Call the **Display Main Menu** task.

# **View Drill-Down Customers**

Abstract Code

* User selected a customers from the list on the **Gross Customer Income report**
* Retreve the Customer ID, Sales, repairs from the **Gross Customer Income** task.
* Find and place all following data in a list, each row is for one **Sale**:
  + **Sale**.SoldDate
  + **Sale**.SoldPrice
  + Find the **Vehicle** associated with **Sale**, and get the **Vehicle**.VIN
  + **Vehicle**.manufactuer
  + **Vehicle**.model
  + **Sale**.salespersonname
* The listing should be sorted by sale date descending and VIN ascending
* Display the list in the **section for vehicle sales** on the **Drill-Down Screen**
* Find and place all following data in a list, each row is for one **Repair**:
  + **repair**.startdate
  + **repair**.completedate if available
  + Find the **Vehicle** associated with **Repair**, and get the **Vehicle**.VIN
  + **repair.**odometer
  + **repair**.labor cost
  + parts cost
  + total cost
  + the service writer who opened the repair
* This listing should be sorted by start date descending, end date descending, and VIN ascending; however, any incomplete repairs should be listed before completed ones with the same sorting criteria.
* Display the list in the **section for repairs** on the **Drill-Down Screen**
* Click ***Return to Main Menu*** button: Call the **Display Main Menu** task.

### **Repairs by Manufacturer/Type/Model**

Abstract Code

* Called from the **Main Menu screen**
* Query for information about each **Repair**, Count the number of **Repair**, the sum of all parts cost, the sum of all labor cost, and the sum of total repair costs, including any repairs in progress for each **Vehicle**.Manufacturer. The **Vehicle**.Manufacturer is found by using **Vehicle**.VIN associate with each **repair**.
* Populate these data in a list where each row is for one **Vehicle**.Manufacturer, Manufacturers whose vehicles do not have any repairs should be listed on this list, and the list should be sorted by manufacturer name ascending.
* Display the list on **Repairs by Manufacturer/Type/Model screen**
* Users are able to select one manufacturer’s name from the list
  + Run **Drill-down** task with the manufacturer’s name

# **View Drill-Down Manufacturer**

Abstract Code

* User select a manufacturer from the list on the **Repairs by Manufacturer/Type/Model screen**
* Retrieve the manufacturer, **Vehicle**, **repairs** from the **Repairs by Manufacturer/Type/Model** task.
* Find and place all following data in a list, each row is for one **Vehicle**.Type:
  + repair count
  + parts costs
  + labor costs
  + total costs
* Find and place all following data in a list, each row is for one **Vehicle**.model:
  + repair count
  + parts costs
  + labor costs
  + total costs
* Lists are sorted by repair count descending(by vehicle type sorted first, and then detail rows sorted).
* Display the list on the **Drill-Down Screen**
* Click ***Return to Main Menu*** button: Call the **Display Main Menu** task.

### **Below Cost Sales**

Abstract Code

* Called from the **Main Menu screen**
* Query for information about each **Sale**, find the **Sale** that **Sale**.invoicepprice > **Sale**.soldprice. Find and place all following data in a list, each row is for one **Sale**:
  + **Sale**.completedate
  + **Sale**.invoice price
  + **Sale**.sold price
  + sold price/invoice price ratio as a percentage
  + **customer**.name. Retrieve customer’s name by **customer** ID associate with **Sale**.
  + **Sale**.salesperson’s name
* For a sale whose ratio is less than or equal to 95%, the background of that row should be highlighted red. Sales should be listed by sales date descending and ratio descending.
* Display the table in the **Below Cost Sales report**
* Click ***Return to Main Menu*** button: Call the **Display Main Menu** task.

### **Average Time in Inventory**

Abstract Code

* Called from the **Main Menu screen**
* Query for information about each **Sale**, find the **Sale**.complete date. Find the Vehicle.Date using VehicleVIN associated with Sale. Calculate the difference between **Sale**.compolete date and **Vehicle**.Date as the amount of time a vehicle remains in inventory group by Vehicle.Type. Calculate and Put the average amount of time a vehicle remains in inventory in a list, each row is for one Vehicle.Type
* If a **vehicle**.type has no sales history, the report should display “N/A” for that **vehicle**.type.
* Display the table in the **Average Time in Inventory report**
* Click ***Return to Main Menu*** button: Call the **Display Main Menu** task.

### **Parts Statistics**

Abstract Code

* Called from the **Main Menu screen**
* Query for information about each **Part**, find the **Part**.price and **Part**.quantity group by **Part**.vendorname. Calculate the total cost and total quantity of part for each **Part**.vendorname. Then put the vendor’s name, the number of parts supplied by that vendor, and the total dollar amount spent on parts in a list.
* Display the list in the **Parts Statistics report**
* Click ***Return to Main Menu*** button: Call the **Display Main Menu** task.

### **Monthly Sales**

Abstract Code

* Called from the **Main Menu screen**
* Query for information about **Sale**.
  + Group the Sale by year and month based on **Sale**.Date then calculate the count and sum for each group.
  + create a list which has:
    - the total number of vehicles sold, the total sales income, the total net income (calculate by using soldprice - invoice price), and the sold price/invoice price ratio as a percentage (such as 125%) for each year and month based on **Sale.**Date.
  + If a year or month does not have sales data, it can be excluded from this report.
  + When the ratio for a month is greater than or equal to 125%, its row should be highlighted with a green background. If the ratio is less than or equal to 110%, it should be highlighted with a yellow background.
  + The results will be ordered by year and month descending, with the most recent year and month as the first result.
  + Display the list on **Monthly Sales screen**
* Users are able to select one manufacturer’s name from the list:
  + Run **Drill-down** task with the manufacturer’s name

### **Monthly Sales Drill-Down**

Abstract Code

* User select a month/year from the list on the **Monthly Sales screen**
* Retrieve the **sale**s in select month/year group from the **Monthly Sales** task.
* Group the **sale** by **sale**.salesperson then calculates the total vehicles and total sales for each **sale**.salesperson. sort the Sale.salesperson by total vehicles descending followed by total sales descending. The first **sale**.salesperson is the top salesperson
* Display the top salesperson on the **Monthly Sales Drill-Down Screen**
* Click ***Return to Main Menu*** button: Call the **Display Main Menu** task.