CS6400 - Database Systems Concepts & Design

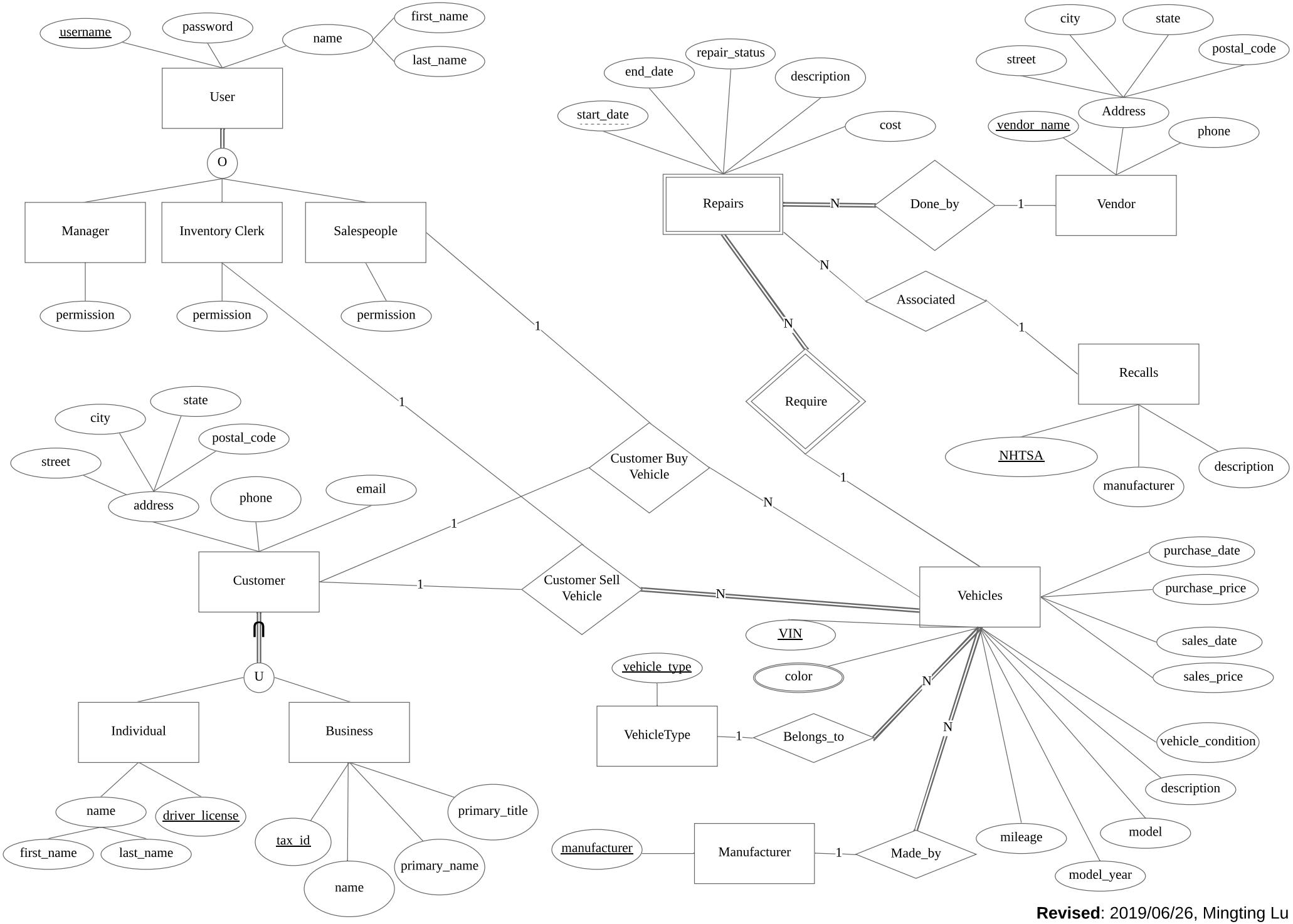
Phase 2 – EER Diagram and Relational Mapping

Team 50

Jirong Huang, Mingting Lu, Qiaojue Wang, Quanfei Zhang

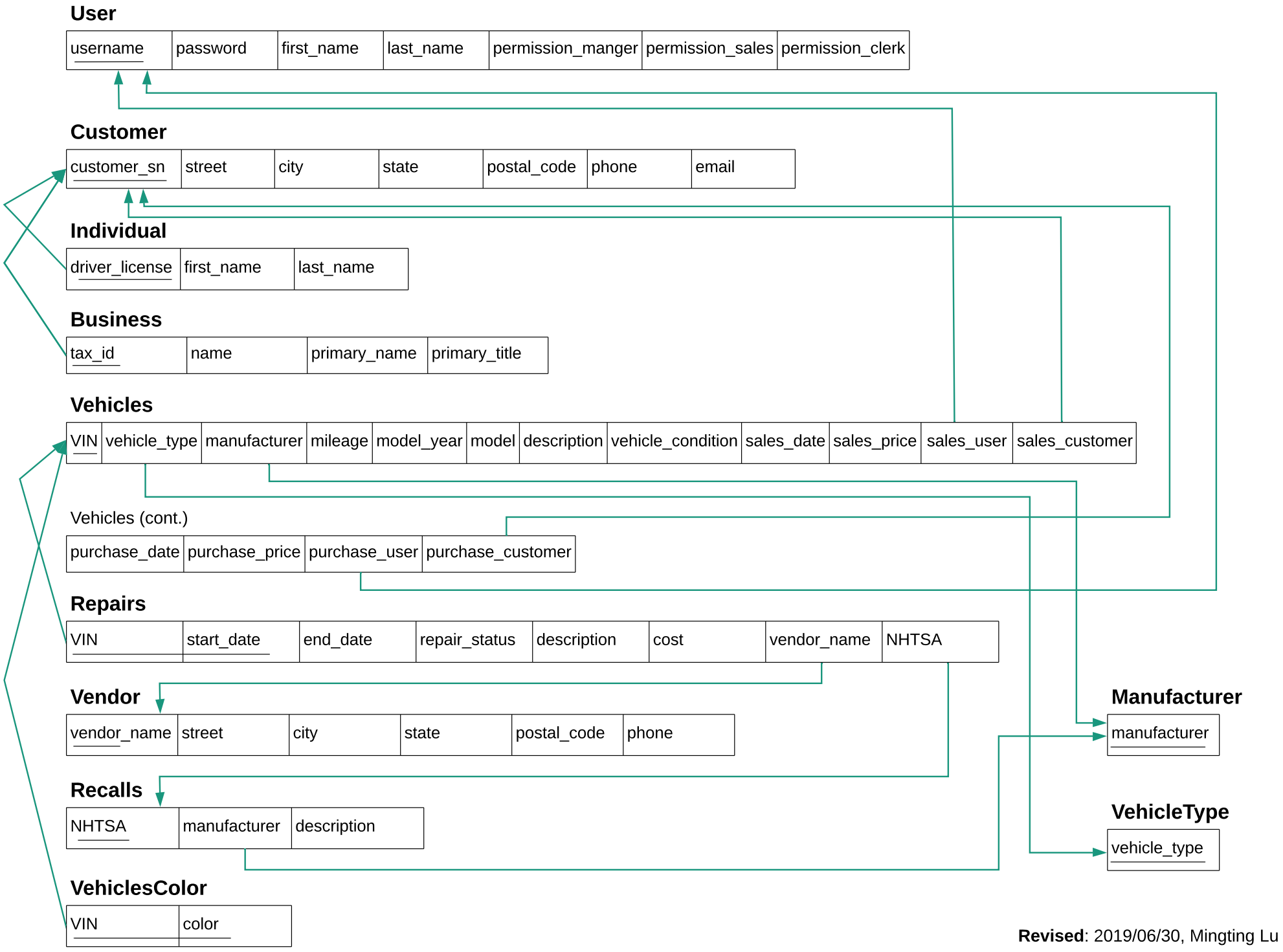
{jhuang433, mlu305, qwang377, qzhang392}@gatech.edu

Phase 2 Updated EER | CS 6400 - Summer 2019 | **Team 050**



Revised: 2019/06/26, Mingting Lu

Phase 2 EER to Relational Mapping | CS 6400 - Summer 2019 | **Team 050**



Revised: 2019/06/30, Mingting Lu

CS6400 - Database Systems Concepts & Design

Phase 2 – Abstract Code + SQL

Team 50

Jirong Huang, Mingting Lu, Qiaojue Wang, Quanfei Zhang

{jhuang433, mlu305, qwang377, qzhang392}@gatech.edu

Table of Content

(The following titles are clickable)

Task Decomposition & Abstract Code

In this report, we use the following naming conventions for abstract code.

- **Documents** are listed as bolded underlined
- ***Buttons*** are listed as bold and italics
- **Tasks** are bolded
- Tables are denoted in blue font
- Aliased tables created within SQL query are denoted in purple
- Attribute values are in lower case wrapped with rounded brackets and ($ ") e.g. ('$username')

To execute SQL in line queries,

- You may first create a database and insert data into it by executing the .sql script in ./Phase_2/team50_p3_seed_data.sql
- Next you may execute the sql scripts specific to the respective section in folder ./Phase_2/sql_for_abstract_code

1.1. Log In

Task Decomposition

- **Lock Types**: Read-only on User.
- **Number of Locks**: Single
- **Enabling Conditions**: None
- **Frequency**: Around 50 logins per day.
- **Consistency (ACID)**: not critical, order is not critical
- **Subtasks**: Mother Task is not needed. No decomposition needed.

Abstract Code

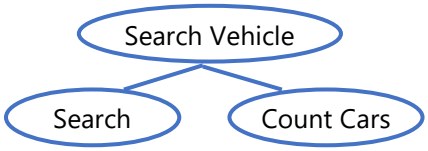
- User click ***User Login*** button from **Search Page**
- User enters username ('$username'), password ('$password') input fields.
- If data validation is successful for the both username and password input fields, then:
    - When **Login** button is clicked:

```
SELECT password FROM User WHERE username = '$username';
```

        - If User record is found but User.password != ('$password'):
            - Go back to **User Login Page**, with error message - 'Wrong password'
        - Else If User record is found & User.password == ('$password'):
            - Go to **Search Page**
            - Fetch User.permissions, User.name (first_name and last_name) into HTTP session, so that we can reduce the times of looking up User
- Else email and password input fields are invalid, display Login form, with error message – 'Invalid username or password'

1.2. Search Vehicle

Task Decomposition



- **Lock Types**: Read-only on Vehicles table and Repairs table
- **Number of Locks**: Two, different schema constructs are needed
- **Enabling Conditions**: None
- **Frequency**: High - Two have same frequencies
- **Consistency (ACID)**: Consistency is not critical; order is not critical.
- **Subtasks**: All tasks must be done, but can be done in parallel. Mother task is needed to coordinate subtasks.

Abstract Code

- Show buttons/ dropdown lists:
  - ***User Login*** if the user is not logged in
  - ***Search*** button
  - ***Add Vehicle*** for logged in user with User.permissions == 'Clerk'
  - ***Report*** dropdown for logged in user with User.permissions == 'Manager'
  - ***Sold Filter*** dropdown for logged in user with User.permissions == 'Manager'.
- Show search criteria:
  - Vehicle Type, Manufacturer, Model Year, Color, and Keyword.
  - VIN field for all logged in user
- Count for cars in database:

```
SELECT COUNT(Vehicles.VIN) AS COUNT

FROM Vehicles;
```

- Number of cars available (Count for vehicles that all the Repairs.repair_status are 'complete' or that have no repair record, and Vehicle.sales_date IS NULL i.e not sold yet)

```
SELECT ((SELECT COUNT(Vehicles.VIN) FROM Vehicles WHERE Vehicles.sales_date IS NULL)  -
COUNT(DISTINCT Vehicles.VIN)) AS COUNT

FROM Vehicles

INNER JOIN Repairs on Vehicles.VIN = Repairs.VIN

WHERE (Repairs.repair_status = "Pending" OR Vehicles.repair_status = "In Progress");
```

- Number of cars under repair (Repairs. vehicle_status has any in {'Pending', 'In Progress'}) for logged in user with User.permissions in {'Clerk', 'Manager'}

```
SELECT COUNT(DISTINCT Vehicles.VIN) AS COUNT

FROM Vehicles

INNER JOIN Repairs on Vehicles.VIN = Repairs.VIN

WHERE (Repairs.repair_status = "Pending" OR Repairs.repair_status = "In Progress");
```

- Upon
  - Click ***User Login*** button – Jump to the **Log In** task.

- Click **Search** button – Query for information about the vehicles and their repair status where criteria match the corresponding attributes.
    - Search with the following criteria:
        - Vehicles.VIN == ('$vin')
        - AND Vehicles.vehicle_type == ('$vehicle_type')
        - AND Vehicles.manufacturer == ('$manufacturer')
        - AND Vehicles.model_year == ('$model_year')
        - AND ('$color') in any Vehicles.color
        - AND ('$keyword') in any substring of {Vehicles.manufacturer, Vehicles.model_year, Vehicles.model_name, Vehicles.description}
        - AND Repairs.status.all() == 'complete' if User.permissions == 'Salespeople' or not logged in user
    - Display the query result as a table at the bottom of the search page: Vehicles.VIN, Vehicles.vehicle_type, Vehicles.model_year, Vehicles.manufacturer, Vehicles.model_name, concatenating all Vehicles.color into one sting, Vehicles.mileage, Vehicles.sales_price, and Vehicles.sales_date (not displaying Vehicles.sales_date but use it for manager's filter.)

*Alias tables in purple

```
SELECT Vehicles_available.VIN , Vehicles_available.vehicle_type, Vehicles_available.model_year,

Vehicles_available.manufacturer, Vehicles_available.model, Vehicles_available.Color,
Vehicles_available.mileage,  Vehicles_available.sales_price, Vehicles_available.sales_date


FROM


############Cars that are available for sale############

   (SELECT a.not_available_indicator, v.VIN ,v.vehicle_type, v.model_year, v.manufacturer, v.mileage,

   v.sales_price, v.description, v.model, GROUP_CONCAT(DISTINCT VehiclesColor.color SEPARATOR
', ') as Color,

   CONCAT(v.manufacturer, " ",

        v.model_year, " ",

        v.model, " ",

        v.description, " ",

        GROUP_CONCAT(DISTINCT VehiclesColor.color SEPARATOR ', ')

       ) AS keyword


   FROM


   (SELECT DISTINCT Vehicles.VIN, 1 as not_available_indicator

   FROM Vehicles
```

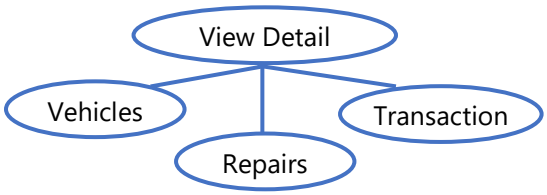
```
    INNER JOIN Repairs on Vehicles.VIN = Repairs.VIN
    WHERE (Repairs.repair_status = "Pending" OR Repairs.repair_status = "In Progress" OR
Vehicles.sales_date IS NULL)
    ) as a

    RIGHT JOIN Vehicles as v on a.VIN = v.VIN
    LEFT JOIN VehiclesColor on v.VIN = VehiclesColor.VIN
    WHERE a.not_available_indicator IS NULL
    GROUP BY v.VIN
    ) as Vehicles_available
##########################################################

WHERE
Vehicles_available.VIN LIKE '$VIN' AND
Vehicles_available.vehicle_type LIKE '$vehicle_type' AND
Vehicles_available.manufacturer LIKE '$manufacturer' AND
Vehicles_available.model_year = '$model_year' AND
Vehicles_available.Color LIKE '$Color' AND     #Created in alias tables
Vehicles_available.keyword LIKE '$keyword'     #Created in alias tables
;
```

- The values in the search result table are clickable and will jump to **View Detail** task.
- If no vehicle matches the search criteria – Display ‘Sorry, it looks like we don’t have that in stock!’.

- Click ***Add Vehicle*** button will switch to the **Add Vehicle** task.
- Click the element in ***Report*** dropdown will switch to **View Report** task with (‘$selected_report’)
- Click the element in ***Sold Filter*** dropdown – Update the search result table with following rules:
  - ‘Sold Vehicle’ is selected: Vehicles.sales_date != NULL
  - ‘Unsold Vehicle’ is selected: Vehicles.sales_date == NULL
  - ‘All Vehicle’ is selected: VIN != NULL

1.3. View Detail

Task Decomposition



- **Lock Types**: 7 Read-only look-ups of Vehicles, Repairs, Recalls, User and Customer (including Individual and Business) information for a searched vehicle.

- **Number of Locks**: Depending on which subtask(s) is going to execute. Range from 1 to 7.
- **Enabling Conditions**: When the hyperlinked value in search result table from **Search Page** is clicked.
- **Frequency**: Middle - All four have the same frequency.
- **Consistency (ACID)**: Consistency is not critical, even if the information is being edited by inventory clerk or salespeople or the owner, while a user or public member is looking at it.
- **Subtasks**: Subtasks should be decided by User.permissions. For multiple subtasks are needed, they can be done in parallel. Mother task is required to decide what subtasks should be executed and further coordinate subtasks. Order is not necessary.

Abstract Code

- Show buttons:
  - ***Back Arrow*** button
  - ***Sell This Car*** button for logged in user with User.permissions == 'Salespeople
  - ***Add Repair*** button for logged in user with User.permissions == 'Clerk'
- Upon
  - Click ***Back Arrow*** button switches to the **Search Vehicle** task.
  - Click ***Sell This Car*** button switches to the **Add Sales Order** task with ('$sales_price') (fetch from Vehicles.sales_price).
  - Click ***Add Repair*** button switches to the **Add Repair** task.
- Run the **View Detail** task: query for information about the vehicle, their repair history and related recall information. The User.permissions of current user should be cached from the HTTP session. The ('$vin') is the vehicle ID of selected record in the search result table from **Search Page**.
  - Run **Vehicles** subtask
    - Find the current Vehicles using Vehicles.VIN == ('$VIN').
    - Display Vehicles VIN, vehicle_type, model_year, manufacturer, mileage, sales_price, descrption.
    - For each color for the Vehicles.color:
      - Concatenate(('$color'), " ", color)
      - Display ('$color')

```
SELECT Vehicles.VIN, Vehicles.vehicle_type, Vehicles.model_year, Vehicles.manufacturer,
Vehicles.mileage, Vehicles.sales_price, Vehicles.description, GROUP_CONCAT(DISTINCT
VehiclesColor.color SEPARATOR ', ') AS Color

FROM Vehicles

LEFT JOIN VehiclesColor ON Vehicles.VIN = VehiclesColor.VIN

WHERE Vehicles.VIN = '$VIN'

GROUP BY Vehicles.VIN;
```

  - Run **Repairs** subtask
    - Check User.permissions in {'Clerk', 'Manger'}
    - Find the current Repairs using ('$vin'). Display Repairs vendor_name, start_date, end_date, status, cost, NHTSA in tabulate form. Fetch Repairs NHTSA, description.
    - Map the current Recalls using Repairs.NHTSA == Recalls.NHTSA. Fetch Recalls description.

```
SELECT Repairs.VIN, Repairs.start_date, Repairs.end_date, Repairs.repair_status,  Repairs.cost,
Repairs.vendor_name, Repairs.NHTSA
```

```
FROM Repairs
WHERE Repairs.VIN = '$VIN';
```

- If the repair record in the table is clicked, pop-up a small window display Repairs.description and Recalls.description.

```
SELECT Repairs.description, Recalls.description
FROM Repairs
LEFT JOIN Recalls ON Repairs.NHTSA = Recalls.NHTSA
WHERE Repairs.VIN = '$VIN' AND Recalls.NHTSA = '$NHTSA';
```

- For User.permissions in {‘Clerk’, ‘Owner’}, the Repairs.status that are not ‘complete’ are clickable, which will jump to **Update Repair Status** task.

- Run **Transaction** subtask for inventory clerks
  - Check User.permissions == ‘Clerk’
  - Find the current Vehicles using (‘$VIN). Display Vehicles.purchase_price.

```
SELECT purchase_price FROM `Vehicles` WHERE Vehicles.VIN = '$VIN';
```

- Find the current Repairs using (‘$VIN’). Calculate and display (‘$total_repair_cost’) by adding up all Repairs.cost

```
SELECT sum(cost) FROM Repairs WHERE Repairs.VIN = '$VIN';
```

- Run **Transaction** subtask for manager and owner
  - Check User.permissions in {‘Manger’, ‘Owner’}
  - Find the current Vehicles using (‘$vin’). Display Vehicles purchase_date, purchase_price, sales_date. Fetch Vehicles purchaser_username, sales_username, seller_id, buyer_id.
  - Find the current Repairs using (‘$vin’). Calculate and display (‘$total_repair_cost’) by adding up all Repairs.cost.

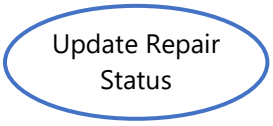
```
SELECT sum(cost) FROM Repairs WHERE Repairs.VIN = '$VIN';
```

- Find the current User using Vehicles.purchase_username and Vehicles.sales_username. Display User name (first_name and last_name) as (‘$purchaser’) and (‘$salesperson’) respectively.
- Find the Customer using Vehicles.seller_id and Vehicles.buyer_id. Displaying seller and buyer information respectively by the following steps.
  - For individual customer: Display Individual name (first_name and last_name). Display Customer address (street, city, state, postal_code), phone, email.
  - For business customer: Display Business name, primary_contact_name, primary_contact_title. Display Customer address (street, city, state, postal_code), phone, email.

```
SELECT Vehicles.VIN, Vehicles.purchase_date, Vehicles.purchase_price,
Vehicles.sales_date, Vehicles.purchase_user, Vehicles.sales_user,
CONCAT( i1.first_name, " ", i1.last_name) AS 'Purchase customer individual name',
CONCAT( i2.first_name, " ", i2.last_name) AS 'Sales customer individual name',
```

```
b1.name AS 'Purchase customer business name',
b2.name AS 'Sales customer business name'
FROM Vehicles
LEFT JOIN Customer as c1 ON Vehicles.purchase_customer = c1.customer_sn
LEFT JOIN Individual as i1 ON c1.customer_sn = i1.driver_license
LEFT JOIN Business as b1 ON c1.customer_sn = b1.tax_ID
LEFT JOIN Customer as c2 ON Vehicles.sales_customer = c2.customer_sn
LEFT JOIN Individual as i2 ON c2.customer_sn = i2.driver_license
LEFT JOIN Business as b2 ON c2.customer_sn = b2.tax_ID
WHERE Vehicles.VIN = '$VIN';
```

1.4. Update Repair Status



Task Decomposition

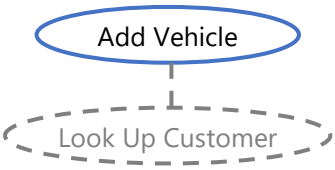
- **Lock Types**: Update of Repairs table.
- **Number of Locks**: Single.
- **Enabling Conditions**: It is enabled when an inventory clerk or an owner is logged in and click the clickable repair status in Repair History section of **Detail Page.**
- **Frequency**: Low frequency.
- **Consistency (ACID)**: Consistency is not critical.
- **Subtasks**: No subtasks.

Abstract Code

- Under Repairs.repair_status column of the table, each of the Repairs.repair_status!= ‘Complete’ record comes with hyperlink. When user clicks on hyperlink to update Repairs.repair_status
    - A box will come up for User to update repair status to {‘Pending’, ‘In Progress’, ‘Complete’}
    - User cannot select the current status.
    - User cannot change if the Repairs.repair_status == ‘Complete’

```
UPDATE Repairs
SET repair_status = '$repair_status'
WHERE VIN = '$VIN';
```

1.5. Add Vehicle



Task Decomposition

- **Lock Types**: Write-only on Vehicles.
- **Number of Locks**: Single.
- **Enabling Conditions**: Enabled after an inventory clerk or the owner log in and click the "Add Vehicle" button from **Search Page**.
- **Frequency**: Middle.
- **Consistency (ACID)**: Consistency is critical. Every required fields must be written into database at the same time.
- **Subtasks**: **Look Up Customer** subtask may be required (the user may directly key in the whole customer ID without the supporting function of searching customer). Mother task is not needed.

Abstract Code

- Show buttons/ dropdown lists:
  - ***Save and Add*** button for adding vehicle
  - ***Cancel*** button if user wishes to exit the page
- Show fields to fill:
  - VIN
  - Vehicle Type, Manufacturer, Model Year, Color, Mileage, Description
  - Purchase Date and Condition
- Following fields will be displayed as default on the screen
  - Customer (purchase): (Individual.first_name & Individual.last_name) or Business.name

```
SELECT Individual.last_name as 'Last Name', Individual.first_name as 'First Name', Business.name as 'Business Name'

FROM Vehicles

LEFT JOIN Individual on Vehicles.purchase_customer = Individual.driver_license

LEFT JOIN Business on Vehicles.purchase_customer = Business.tax_id

WHERE Vehicles.VIN= '$VIN';
```

  - Purchase Price from kbb.com, Vehicles.purchase_price

```
SELECT purchase_price

FROM Vehicles

WHERE Vehicles.VIN= '$VIN';
```

- User enters the following into input fields
  - Vehicles.VIN ('$VIN')
  - Vehicles.name ('$model_name')
  - Vehicles. model_year ('$model_year')
  - Vehicles. mileage ('$mileage')
  - Vehicles. description ('$description')
- User selects from dropdown/ selection list for the following fields
  - Vehicles. purchase_date ('$purchase_date')
  - Vehicles. Vehicle_condition ('$vehicle_condition')
  - Vehicles. vehicle_type ('$vehicle_type')
  - Vehicles. manufacturer ('$manufacturer')

- Under seller field, user will carry out **Look up/ Add Customer task** i.e. User will first try to find the customer. Or the user can add Customer by clicking ***Add Ind*** or ***Add Bus*** to add individual or business customer into the database.

```
SELECT customer_sn FROM Customer
WHERE Customer.customer_sn = '$customer_sn';
```

```
INSERT INTO Customer(customer_sn, street, city, state, postal_code, phone, email)
VALUES ('$customer_sn', '$street', '$city', '$state', '$postal_code', '$phone', '$email');
```

- For Vehicles.color field in the User Interface,
  - User first selects a color from a dropdown list
  - If user chooses to add 1 more color, he/she will click on the "+" button and select additional color from dropdown list
  - In the case which there're more than 1 color on the form; if user wishes to remove a color, he/she will click on the "-" button
- Once user is ready to enter the vehicle:
  - ***Save and Add*** button is clicked:
    - If all fields are filled and business logic constraints for every field is satisfied
      - Following message will be displayed - 'Vehicle added successfully'
    - Else if any of the fields is missing
      - Following message will be displayed - '<Field> is missing'
    - Else if any of business logic constraints is not satisfied
      - Following message will be displayed - '<Field> has failed to satisfy the following requirements: <Business Logic Constraints>'

```
INSERT INTO Vehicles (VIN, vehicle_type, manufacturer, mileage, model_year, model, description,
vehicle_condition, sales_date, sales_price, sales_user, sales_customer, purchase_date, purchase_price,
purchase_user, purchase_customer)
VALUES ('$VIN', '$vehicle_type', '$manufacturer', '$mileage', '$model_year', '$model_name',
'$description', '$vehicle_condition', NULL, (1.25 * '$purchase_price'), NULL, NULL, '$purchase_date',
'purchase_price', $purchase_user', '$purchase_customer');
```

- If any point of time, user wish to exit the form back to **Search Page**, he/ she can click on the **Cancel** button.

1.6. Add Sales Order

Task Decomposition

- **Lock Types**: Update on Vehicles.
- **Number of Locks**: Single.
- **Enabling Conditions**: It is enabled when a salesperson or an owner is logged in and click the ***Sell This Car*** button on **Detail Page**.
- **Frequency**: Low.

- **Consistency (ACID)**: Consistency is critical. Every required fields must be written into database at the same time.
- **Subtasks**: **Look Up Customer** subtask may be required (the user may directly key in the whole customer ID without the supporting function of searching customer). Mother task is not needed.

Abstract Code

- Show buttons/ dropdown lists:
  - ***Save and Add*** button for adding sales order
  - ***Cancel*** button if user wishes to exit the page
- Show fields to fill:
  - Sales date
  - Sales price
  - Buyer
- Following fields will be displayed as default on the screen
  - Vehicles.VIN ('$VIN')

```
SELECT VIN FROM Vehicles WHERE Vehicles.VIN = '$VIN';
```

  - Salesperson name - User. Name ('$name' - composite attribute of '$first_name' and '$last_name')

```
SELECT first_name, last_name FROM User WHERE User.username = '$username';
```

  - Vehicles. sales_price ('$sales_price')

```
SELECT sales_price FROM Vehicles WHERE Vehicles.VIN = '$VIN';
```

- To add a customer – buyer to sales order form
  - User will carry out **Look up/ Add Customer task** i.e. User will first try to find the customer. If he/she is not found, user will proceed to add customer

```
SELECT customer_sn FROM Customer WHERE Customer.customer_sn = '$customer_sn';
```

```
INSERT INTO Customer(customer_sn, street, city, state, postal_code, phone, email)
VALUES ('$customer_sn', '$street', '$city', '$state', '$postal_code', '$phone', '$email');
```

- User enters the following sales order form information into input fields
  - Vehicles.sales_date ('$sales_date')
  - Vehicles.sales_price ('$sales_price')

```
UPDATE Vehicles SET Vehicles.sales_date = '$sales_date' WHERE Vehicles.VIN = '$VIN';
```

- Once user is ready to enter the sales order form:
  - ***Save and Add*** button is clicked:
    - If all fields are filled and business logic constraints for every field is satisfied
      - Following message will be displayed - 'Sales order form added successfully'
    - Else if any of the fields is missing
      - Following message will be displayed - '<Field> is missing'
    - Else if any of business logic constraints is not satisfied
      - Following message will be displayed - '<Field> has failed to satisfy the following requirements: <Business Logic Constraints>'
- If any point of time, user wish to exit the form back to **Detail Page**, he/ she can click on the ***Cancel*** button

1.7. Look Up/ Add Customer

Look Up Customer Add Customer

Task Decomposition

- **Lock Types**: Read-only lookup or write-only on Customer.
- **Number of Locks**: Single.
- **Enabling Conditions**: It is enabled by an inventory clerk or the owner adding a vehicle through **Add Vehicle** task, or the salespeople or the owner selling a car through **Add Sales Order** task.
- **Frequency**: Middle frequency for customer lookup. Low frequency for adding customer.
- **Consistency (ACID)**: Consistency is critical when adding customer. Every required fields must be written into database at the same time.
- **Subtasks**: No subtasks.

Abstract Code

- Show buttons/ dropdown lists:
  - ***Add Ind*** radio button to add an individual customer or ***Add Bus*** radio button to add a business customer. Based on selection, an individual or business customer form will be displayed.
  - ***Add Customer*** button to write customer information to database.
- Show fields to fill when ***Add Ind*** or ***Add Bus*** radio button is selected
  - For individual
    - Driver's license number
    - Name, first name and last name
    - Address, Street, City, State, Postal code
    - Phone number
    - Email (optional)
  - For business:
    - Tax Identification Number
    - Business name
    - Address, Street, City, State, Postal code
    - Phone number
    - Email (optional)
    - Primary contact name
    - Primary contact Title
- Under **Add Vehicle** task & **Add Sales Order** task, user is required to include the Customer identifier (Individual.driver_license or Business.tax_id)
- User will first **Look Up Customer** by entering the following into input fields
  - Driver license number, Individual.driver_license ('$driver_license') or Tax identification number, Business.tax_id ('$tax_id')

```
SELECT customer_sn FROM Customer WHERE Customer.customer_sn = '$customer_sn';
```

  - And click on ***Search*** button
  - If Individual.driver_license == ('$driver_license') or Business.tax_id == ('$tax_id') i.e. in the database
    - Return identifier ('$driver_license' or '$tax_id') and name
  - Else if NOT (Individual.driver_license == ('$driver_license') or Business.tax_id == ('$tax_id')) i.e. not in the database

- Return the following message: “Customer is not found in database”
- User will then proceed to **Add Customer** by entering the following into input fields that are applicable to Individual or Business)
    - Address (‘$address’ with multi-attributes into separate fields)
        - Customer.street (‘$street’)
        - Customer.city (‘$city’)
        - Customer.state (‘$state’)
        - Customer.postal code (‘postal_code’)
    - Customer.phone (‘$phone’)
    - Customer.email (‘$email’)

```
INSERT INTO Customer(customer_sn, street, city, state, postal_code, phone, email)
VALUES ('$customer_sn’, '$street', '$city', '$state', '$postal_code', '$phone', '$email');
```

- If user wants to add an individual
    - User selects radio button, ***Add Ind*** under individual section
    - User enters the following into input fields
        - Individual.driver_license (‘$driver_license’)
        - Name (‘$name’ with multi-attributes into separate fields)
            - Individual.first_name (‘$first_name’)
            - Individual.last_name (‘$ last_name’)

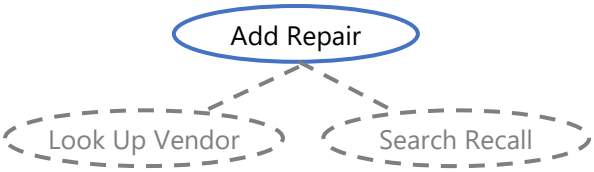
```
INSERT INTO Individual (driver_license, first_name, last_name)
VALUES('$driver_license’, ‘$first_name’,  ’$last_name');
```

- Else if user wants to add a Business instead
    - User selects radio button, ***Add Bus*** under individual section
    - User enters the following into input fields
        - Business.tax_id (‘$tax_id’)
        - Business.name (‘$name’)
        - Business.primary_name (‘$primary_name’)
        - Business.primary_title (‘$primary_title’)

```
INSERT INTO Business (tax_id, name, primary_name, primary_title)
VALUES('$tax_id', '$Name', '$primary_name', '$primary_title');
```

- Once user is ready to enter the customer:
    - ***Add Customer*** button is clicked:
        - If all required fields are filled and business logic constraints for every field is satisfied
            - Following message will be displayed - Customer is added successfully’
        - Else if any of the fields is missing
            - Following message will be displayed - ‘<Field> is missing’
        - Else if any of business logic constraints is not satisfied
            - Following message will be displayed - ‘<Field> has failed to satisfy the following requirements: <Business Logic Constraints>’
- If any point of time, user wishes to exit the form back to **Search Page** or **Detail Page** depending on the previous page, he/ she can click on the ***Cancel*** button

1.8. Add Repair



Task Decomposition

- **Lock Types**: Write-only of Repairs and Vehicle table.
- **Number of Locks**: Single.
- **Enabling Conditions**: It is enabled when an inventory clerk or an owner is logged in and click the ***Add Repair*** button on **Detail Page**.
- **Frequency**: Middle frequency.
- **Consistency (ACID)**: Consistency is critical. Every required fields must be written into database at the same time.
- **Subtasks**: **Look Up Vendor** and **Search Recall** subtasks may not be required (the user may directly fill in the corresponding fields without the supporting function of searching customer). Mother task is not needed.

Abstract Code

- ***Save and Add*** button for adding repair to database
- ***Cancel*** button if user wishes to exit the page
- Show fields to fill
    - Vendor name
    - Start date
    - End date
    - Repair cost
    - Repair and description
- Following fields will be displayed as default on the screen
    - Vehicles.VIN ('$VIN')
    - Repairs.status ('$status')
- User will then proceed to add the following input fields,
    - Vendor.name ('$name')
        - carrying out **Look up Vendor** task to search the existing Vendor and choose the expected Vendor name
        - if Vendor is not exist after search, then manually press the ***Add Vendor*** button and conduct the **Add Vendor** task stated under 1.10 section.
    - Repairs.start_date ('$start_date')
    - Repairs.end_date ('$end_date')
    - Repairs.cost ('$cost')
    - Repairs.description ('$description')
    - Recalls.NHTSA('$NHTSA')
        - carrying out **Search Recall** task to search the existing Recall and choose the expected Recall number
        - if Recall is not exist after search, then manually press the ***Add Recall*** button and conduct the **Add Recall** task stated under 1.9 section.
- Once user is ready to enter the fields:
    - ***Save and Add*** button is clicked:
        - If all fields are filled and business logic constraints for every field is satisfied

```
INSERT INTO Repairs (VIN, start_date, end_date, repair_status, description, cost, vendor_name, NHTSA)
VALUES('$VIN', '$start_date', '$end_date', '$repair_status', '$description', '$cost', '$vendor_name', '$NHTSA');
```

- Following message will be displayed - ‘Add Repair form added successfully’
- Else if any of the fields is missing
  - Following message will be displayed - ‘<Field> is missing’
- Else if any of business logic constraints is not satisfied
  - Following message will be displayed - ‘<Field> has failed to satisfy the following requirements: <Business Logic Constraints>’
- At the same time the button is clicked, Vehicle.sales_price will be updated

```
UPDATE Vehicles
SET Vehicles.sales_price = Vehicles.sales_price + 1.1 * '$cost'
WHERE Vehicles.VIN = '$VIN';
```

- If any point of time, user wishes to exit the form back to **Detail Page**, he/ she can click on the ***Cancel*** button

1.9. Search/ Add Recall

Task Decomposition

- **Lock Types**: Read-only lookup or write-only on Recalls.
- **Number of Locks**: Single.
- **Enabling Conditions**: It is enabled by an inventory clerk or the owner adding a record for repair through **Add Repair** task.
- **Frequency**: Middle frequency for recall lookup. Low frequency for adding new recall.
- **Consistency (ACID)**: Consistency is critical when adding new recall. Every required fields must be written into database at the same time.
- **Subtasks**: No subtasks.

Abstract Code

- ***Search*** button for searching recall
- ***Add Recall*** button to activate form for filling up recall information
- ***Save and Add*** button to write recall information to database
- Under **Add Repair** task & **Add Sales Order** task, user is required to include the Recall number, Recalls.NHTSA (‘$NHTSA’)
- User will first **Search Recall** by entering the following into input fields
  - Recall number, Recalls.NHTSA (‘$NHTSA’)
  - And click on ***Search*** button

```
SELECT NHTSA FROM Recalls WHERE Recalls.NHTSA = '$NHTSA';
```

- If Recalls.NHTSA == ‘$NHTSA’ i.e. in the database
  - Return Recalls.NHTSA (‘$NHTSA’)
- Else if Recalls.NHTSA != ‘$NHTSA’ i.e. not in the database
  - Return the following message: “NHTSA number is not found in database”

- User will then proceed to click ***Add Recall*** button to activate a form and entering the following into input fields
  - Recalls.NHTSA ('$NHTSA')
  - Recalls.manufacturer ('$manufacturer')
  - Recalls.description ('$description')

- Once user is ready to enter the recall number, the user should click ***Save and Add*** button:
  - If all required fields are filled and business logic constraints for every field is satisfied

```
INSERT INTO Recalls (NHTSA, manufacturer, description)
VALUES('$NHTSA', '$manufacturer ', '$description');
```

    - Following message will be displayed – Recall number, NHTSA is added successfully'
  - Else if any of the fields is missing
    - Following message will be displayed - '<Field> is missing'
  - Else if any of business logic constraints is not satisfied
    - Following message will be displayed - '<Field> has failed to satisfy the following requirements: <Business Logic Constraints>'

1.10. Look up/ Add Vendor

Task Decomposition

- **Lock Types**: Read-only lookup or write-only on Vendor.
- **Number of Locks**: Single.
- **Enabling Conditions**: It is enabled by an inventory clerk or the owner adding a record for repair through **Add Repair** task.
- **Frequency**: Middle frequency for vendor lookup. Low frequency for adding new vendor.
- **Consistency (ACID)**: Consistency is critical when adding new vendor. Every required fields must be written into database at the same time.
- **Subtasks**: Mother task is needed. No subtasks.

Abstract Code

- ***Search*** button for searching vendor
- ***Add Vendor*** button to activate form for filling up vendor information
- ***Save and Add*** button to write vendor information to database
- Under **Add Vendor** task, user is required to include Vendor.name
- User will first **Look Up Vendor** by entering the following into input fields
  - Vendor.name ('name')
  - And click on ***Search*** button

```
SELECT vendor_name FROM Vendor WHERE Vendor.vendor_name LIKE '%$vendor_name%';
```

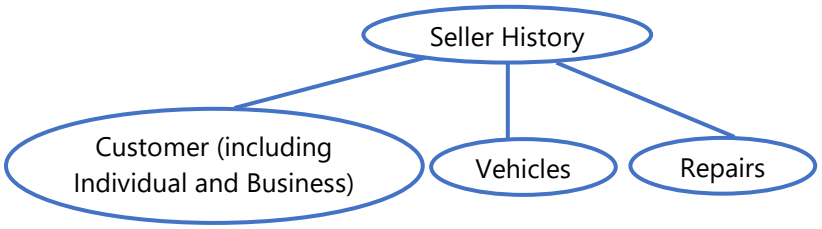
    - If Vendor.name == '$name' i.e. in the database
      - Return Vendor.name
    - Else if Vendor.name != '$name' i.e. not in the database
      - Return the following message: "Vendor is not found in database"
      - User will then proceed to click ***Add Vendor*** button to activate a form and entering the following into input fields

        - Vendor.name ('$name')
        - Address ('$address' with multi-attributes into separate fields)
            - Vendor.street ('$street')
            - Vendor.city ('$city')
            - Vendor.state ('$state')
            - Vendor.postal code ('postal_code')
        - Vendor.phone ('$phone')
        - Vendor.email ('$email')
- Once user is ready to enter the vendor, the user should click ***Save and Add*** button:
    - If all required fields are filled and business logic constraints for every field is satisfied

```
INSERT INTO Vendor (vendor_name, street, city, state, postal_code, phone)
VALUES('$vendor_name', '$street', '$city', '$state', '$postal_code', '$phone');
```

        - Following message will be displayed - Vendor is added successfully'
    - Else if any of the fields is missing
        - Following message will be displayed - '<Field> is missing'
    - Else if any of business logic constraints is not satisfied
        - Following message will be displayed - '<Field> has failed to satisfy the following requirements: <Business Logic Constraints>'

1.11. Seller History



Task Decomposition

- **Lock Types**: Read-only of table Customer (including Individual and Business), Vehicles, and Repairs.
- **Number of Locks**: Three.
- **Enabling Conditions**: It is enabled by a manager or the owner by selecting "Seller History" from the ***Report*** dropdown on **Search Page**.
- **Frequency**: Middle Frequency.
- **Consistency (ACID)**: Consistency is not critical.
- **Subtasks**: Subtasks are needed, they can be done in parallel. Mother task is required to coordinate subtasks. Order is not necessary.

Abstract Code

- Run the **Seller History** task when user select "Seller History" in ***View Report*** dropdown list in **Search Page**.
- Query for information about the seller and their profile using the system from the HTTP Session/Cookie.
- Identify all selling records of customers (sold vehicles to Burdell) using the Customer.customer_sn; Display individual name Concatenate(Individual.first_name, " ",Individual.last_name) OR Business.name
- GROUP BY (Customer.customer_sn) i.e. For each customer:

- COUNT(Vehicles.VIN) - Count total purchased vehicle number by Burdell and display the information.
- AVG(Vehicles.purchase_price) - Calculate average purchase price by Burdell and display the information.
- AVG(Identifier (Vehicles.VIN, Repairs.start_date)) - Calculate average repairs sold by the specific customer and display the information.

- Highlight the tabulated records in red if the average repairs >= 5

```
SELECT
    vehicles_count_price_name.purchase_customer_name AS 'Purchase customer individual name',
    vehicles_count_price_name.purchase_business_name AS 'Purchase customer business name',
    vehicles_count_price_name.count_vehicles AS 'Total Vehicles Purchased',
    vehicles_count_price_name.avg_purchase_price AS 'AVG Purchase Price',
    avg_repairs.AVERAGE_REPAIRS AS 'AVG Repairs per Vehicle',
    IF(avg_repairs.AVERAGE_REPAIRS >=5, 1, 0) AS 'AVG Repairs over 5'

FROM

   (SELECT vehicles_count_price.purchase_customer, vehicles_count_price.count_vehicles,
   vehicles_count_price.avg_purchase_price,
   CONCAT( i1.first_name, " ", i1.last_name) AS 'purchase_customer_name',
   b1.name AS 'purchase_business_name'

   FROM
      (SELECT Vehicles.purchase_customer, COUNT(Vehicles.VIN) as count_vehicles,
AVG(Vehicles.purchase_price) as avg_purchase_price
      FROM Vehicles
      GROUP BY Vehicles.purchase_customer) as vehicles_count_price

   LEFT JOIN Customer as c1 ON vehicles_count_price.purchase_customer = c1.customer_sn
   LEFT JOIN Individual as i1 ON c1.customer_sn = i1.driver_license
   LEFT JOIN Business as b1 ON c1.customer_sn = b1.tax_id
   ) as vehicles_count_price_name

   LEFT JOIN (
      SELECT Vehicles.purchase_customer, COUNT(Repairs.repair_status)/ COUNT(DISTINCT
Vehicles.VIN) AS AVERAGE_REPAIRS
      FROM Vehicles
```

```
        LEFT JOIN Repairs on Vehicles.VIN = Repairs.VIN
        GROUP BY Vehicles.purchase_customer
    ) AS avg_repairs ON vehicles_count_price_name.purchase_customer = avg_repairs.purchase_customer
    ORDER BY vehicles_count_price_name.count_vehicles DESC,
vehicles_count_price_name.avg_purchase_price
;
```

1.12. Inventory Age

Inventory Age

Task Decomposition

- **Lock Types**: Read-only of table Vehicles.
- **Number of Locks**: Single.
- **Enabling Conditions**: It is enabled by a manager or the owner by selecting "Inventory Age" from the ***Report*** dropdown on **Search Page**.
- **Frequency**: Middle Frequency.
- **Consistency (ACID)**: Consistency is not critical.
- **Subtasks**: Mother Task is not needed. No decomposition needed.

Abstract Code

- Run the **Inventory Age** task when user select "Inventory Age" in ***View Report*** dropdown list in **Search Page**.
- Query for information about the vehicle type and their age information using the system from the HTTP Session/Cookie
- Find all the vehicle types by Vehicles.vehicle_ type and display the information
- GROUP BY (Vehicles.vehicle_type) i.e. For each vehicle type:
    - Find unsold vehicles using Vehicles.sales_date = NULL
    - Calculate the days in inventory of all unsold vehicles by using Days_in_inventory = Vehicles.current_ date - Vehicles.purchase_date;
    - Calculate the minimum, average, and maximum of Days_in_inventory and display the information. If there is no unsold vehicle for the vehicle type, display N/A

```
SELECT vehicle_type AS 'Vehicle Type',
MIN(IF(sales_date IS NULL, timestampdiff(day,purchase_date,SYSDATE()), 'N/A')) AS 'MIN Age(days)',
AVG(IF(sales_date IS NULL, timestampdiff(day,purchase_date,SYSDATE()), 'N/A')) AS 'AVG Age(days)',
MAX(IF(sales_date IS NULL, timestampdiff(day,purchase_date,SYSDATE()), 'N/A')) AS 'MAX Age(days)'
FROM Vehicles
WHERE sales_date IS NULL
GROUP BY vehicle_type;
```

1.13. Average Time in Inventory

Average Time in Inventory

Task Decomposition

- **Lock Types**: Read-only of table Vehicles.
- **Number of Locks**: Single.
- **Enabling Conditions**: It is enabled by a manager or the owner by selecting "Average Time in Inventory" from the ***Report*** dropdown on **Search Page**.
- **Frequency**: Middle Frequency.
- **Consistency (ACID)**: Consistency is not critical.
- **Subtasks**: Mother Task is not needed. No decomposition needed.

Abstract Code

- Run the **Average Time in Inventory** task when user select "Average Time in Inventory" in ***View Report*** dropdown list in **Search Page**.
- Query for information about the vehicle type and their age information using the system from the HTTP Session/Cookie.
- Find all the vehicle types using Vehicles.vehicle_ type and display the information
    - Find all the sold vehicles (the Vehicles.sales_date != Null)
    - Calculate the days in inventory of all sold vehicles by using Days_in_inventory_sold = Vehicles.sales_date - Vehicles.purchase_date
    - GROUP BY (Vehicles.vehicle_type) i.e. For each vehicle type:
        - AVERAGE (Days_in_inventory_sold) and display the information.If there is no sold vehicle in the vehicle type, display N/A

```
SELECT
vehicles_days_in_inventory.vehicle_type as 'Vehicle Type',
avg(vehicles_days_in_inventory.days_in_inventory) as 'Average Amount of Time in Inventory'
FROM

(SELECT VIN, vehicle_type, timestampdiff(day, purchase_date, sales_date) as days_in_inventory
FROM Vehicles
GROUP BY vehicle_type, VIN
) AS vehicles_days_in_inventory

GROUP BY vehicles_days_in_inventory.vehicle_type
;
```

1.14. Price Per Condition

Price Per Condition

Task Decomposition

- **Lock Types**: Read-only of table Vehicles.
- **Number of Locks**: Single.
- **Enabling Conditions**: It is enabled by a manager or the owner by selecting "Price Per Condition" from the ***Report*** dropdown on **Search Page**.

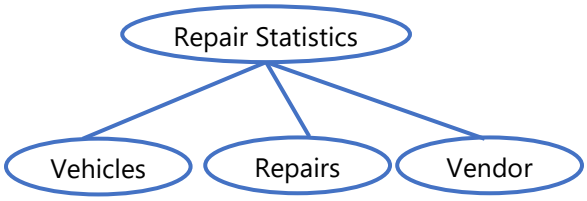
- **Frequency**: Middle Frequency.
- **Consistency (ACID)**: Consistency is not critical.
- **Subtasks**: Mother Task is not needed. No decomposition needed.

Abstract Code

- Run the **Price Per Condition** task when user select "Price Per Condition" in ***View Report*** dropdown list in **Search Page**.
- Query for information about the vehicle type and their price information per condition using the system from the HTTP Session/Cookie.
    - Find all the vehicle types using Vehicles.vehicle_type and display the information
    - GROUP BY (Vehicles.vehicle_type, Vehicles.vehicle_condition) i.e. For each vehicle type and for each condition:
        - Calculate the AVERAGE (Vehicles. purchase_price) and display the information; If there is no vehicle in the specific vehicle type and condition, display "$0";

```
SELECT vehicle_type,

SUM(IF(vehicle_condition = 'Excellent', purchase_price, 0))/SUM(IF(vehicle_condition = 'Excellent', 1, 0)) AS 'Excellent',

SUM(IF(vehicle_condition = 'Very Good', purchase_price, 0))/SUM(IF(vehicle_condition = 'Very Good', 1, 0)) AS 'Very Good',

SUM(IF(vehicle_condition = 'Good', purchase_price, 0))/SUM(IF(vehicle_condition = 'Good', 1, 0)) AS 'Good',

SUM(IF(vehicle_condition = 'Fair', purchase_price, 0))/SUM(IF(vehicle_condition = 'Fair', 1, 0)) AS 'Fair'

FROM Vehicles

GROUP BY vehicle_type
```

1.15. Repair Statistics



Task Decomposition

- **Lock Types**: Read-only of table Vehicles, Repairs, and Vendor.
- **Number of Locks**: Three.
- **Enabling Conditions**: It is enabled by a manager or the owner by selecting "Repair Statistics" from the ***Report*** dropdown on **Search Page**.
- **Frequency**: Middle frequency.
- **Consistency (ACID)**: Consistency is not critical.
- **Subtasks**: Subtasks are needed, they can be done in parallel. Mother task is required to coordinate subtasks. Order is not necessary.

Abstract Code

- Run the **Repair Statistics** task when user select "Repair Statistics" in ***View Report*** dropdown list in **Search Page**.
- Query for information about the repair vendors and their repair information using the system from the HTTP Session/Cookie.
    - Find all the vendors using the Vendor.name and display the information
    - Filter by Repairs.status == "Complete"
    - GROUP BY (Vendor.name) i.e. For each vendor:
        - COUNT(Repairs.status) i.e. Count the number of repairs completed by each vendor and display the information
        - SUM(Repairs.cost) i.e. Calculate the total repair costs and display the information
        - AVERAGE(Repairs.end_date - Repairs.start_date) i.e. Calculate the average length of time to complete the repairs for each vendor and display the information
        - Calculate the average number of repairs per vehicle for each vendor and display the information

```
SELECT vendor_name,
COUNT(Repairs.VIN) as Repair_completed,
SUM(Repairs.cost) as Total_cost_on_completed_Repair ,
(COUNT(Repairs.VIN))/(COUNT(DISTINCT Repairs.VIN)) as Average_repair_per_vehicle,
AVG(timestampdiff(day,Repairs.start_date,Repairs.end_date)) as Average_days_to_completed_repair
FROM Repairs
WHERE Repairs.repair_status ='Complete'
GROUP BY vendor_name;
```

1.16. Monthly Sales

Task Decomposition

- **Lock Types**: Read-only of table User and Vehicles.
- **Number of Locks**: Two.
- **Enabling Conditions**: It is enabled by a manager or the owner by selecting "Monthly Sales" from the ***Report*** dropdown on **Search Page**.
- **Frequency**: High frequency.
- **Consistency (ACID)**: Consistency is not critical.
- **Subtasks**: Subtasks are needed, they can be done in parallel. Mother task is required to coordinate subtasks. Order is not necessary.

Abstract Code

- Run the **Monthly Sales** task when user select "Monthly Sales" in ***View Report*** dropdown list in **Search Page**.
- Query for information about the sales information using the system from the HTTP Session/Cookie.

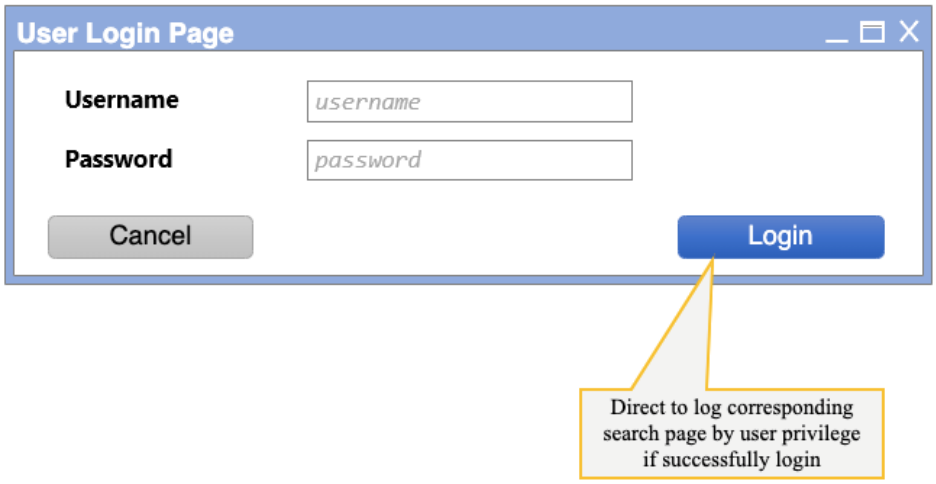
- GROUP BY (Year (Vehicle.sales_date), Year (Vehicle.sales_date)) i.e. For each Year/Month, find all the sales record using the Vehicle.sales_date
    - COUNT (Vehicle.sales_date) Count the number of sold vehicles using Vehicle.VIN
    - SUM(Vehicle.sales_price) Calculate the sales income by add up all the Vehicle.sales_price
    - SUM(Vehicle.sale_price - Vehicle.purchase_price - Repairs.cost) i.e. Calculate the sales income by add up all the net income
    - GROUP BY (User. name) i.e. for each sales people,
        - COUNT (Vehicles.sales_date) Count the number of sold vehicles using Vehicles.sales_date
        - SUM(Vehicle.sales_price) Calculate the sales income by add up all the Vehicle.sales_price
        - SUM(Vehicle.sale_price - Vehicle.purchase_price - Repairs.cost) i.e. Calculate the sales income by add up all the net income

```
SELECT extract(year_month from sales_date) AS 'Year_Month', sales_user,
COUNT(VIN) AS 'Vehicle Sold',
SUM(sales_price) AS 'Sales Income',
SUM(sales_price-purchase_price) AS 'Net Income'
FROM Vehicles
WHERE sales_date IS NOT NULL
GROUP BY extract(year_month from sales_date),sales_user order by extract(year_month from sales_date) DESC;
```

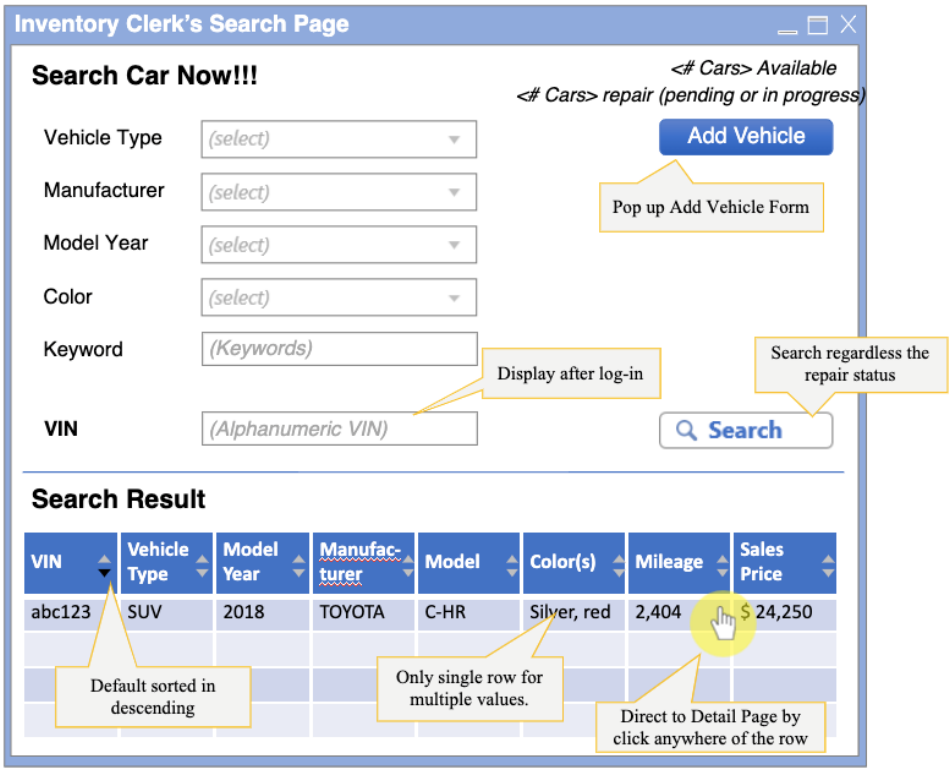
2. Appendix

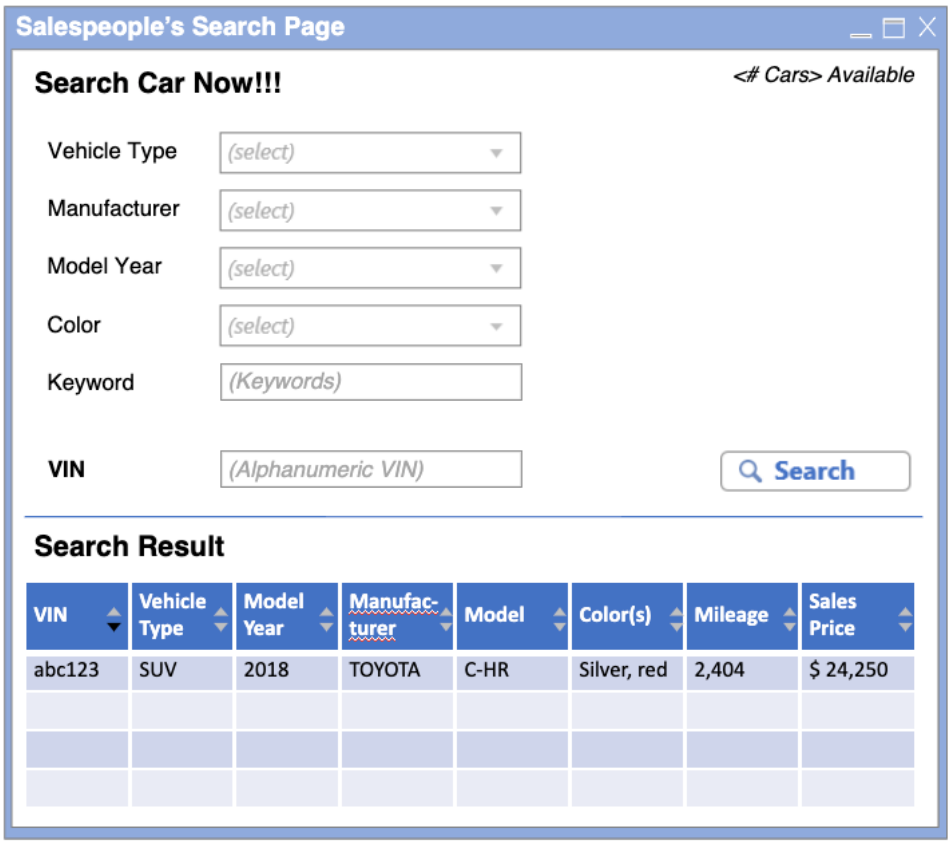
Below are UI for each task in section 1.

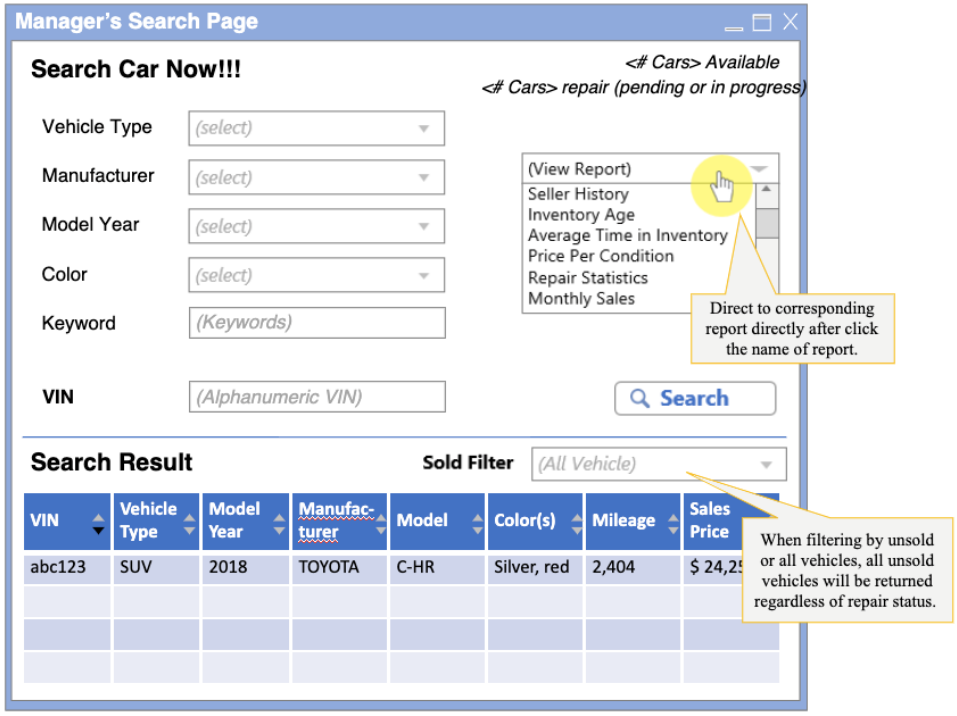
For **Section 1.1:** Log in

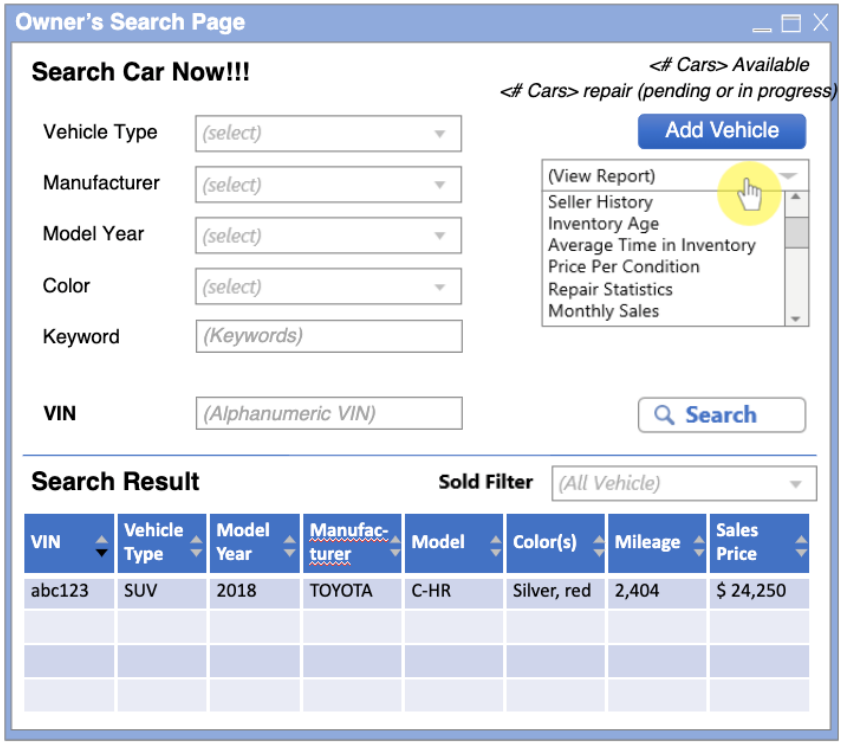


For **Section 1.2:** Search Vehicle

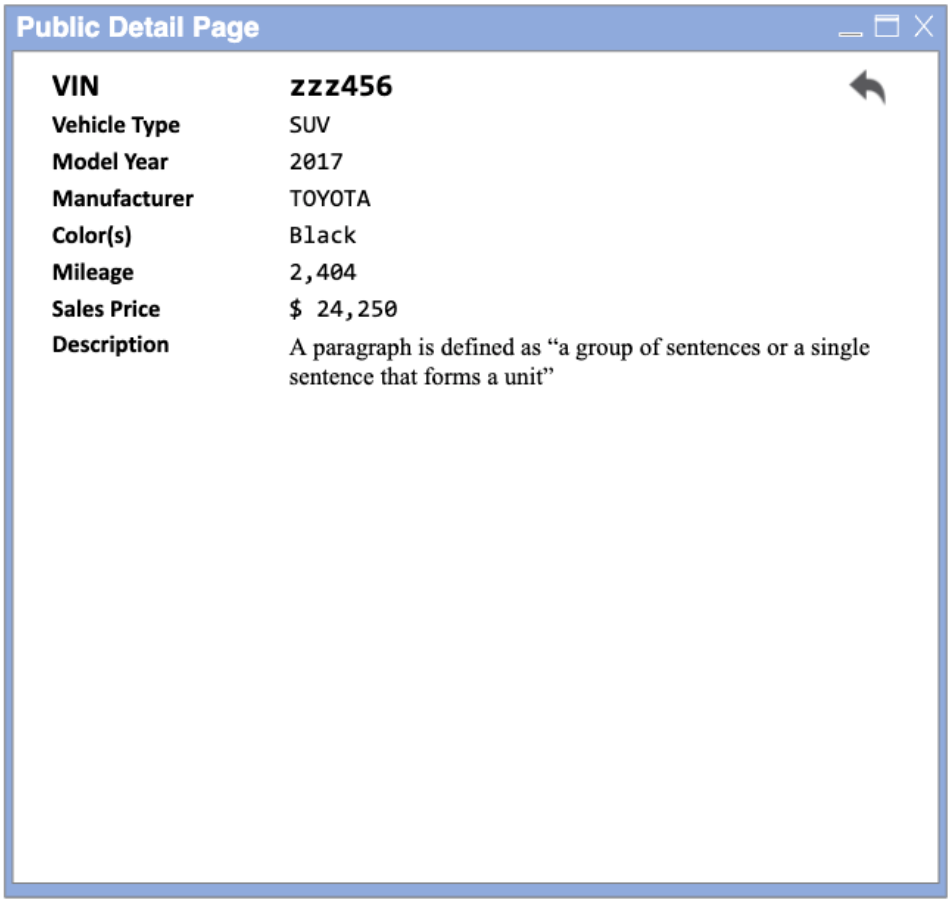

Inventory Clerk's Search Page
Search Car Now!!!
<# Cars> Available
<# Cars> repair (pending or in progress)
Add Vehicle
Pop up Add Vehicle Form
Vehicle Type
(select)
Manufacturer
(select)
Model Year
(select)
Color
(select)
Keyword
(Keywords)
Display after log-in
Search regardless the repair status
VIN
(Alphanumeric VIN)
Search
Search Result
VIN
Vehicle Type
Model Year
Manufac-turer
Model
Color(s)
Mileage
Sales Price
abc123
SUV
2018
TOYOTA
C-HR
Silver, red
2,404
$ 24,250
Default sorted in descending
Only single row for multiple values.
Direct to Detail Page by click anywhere of the row

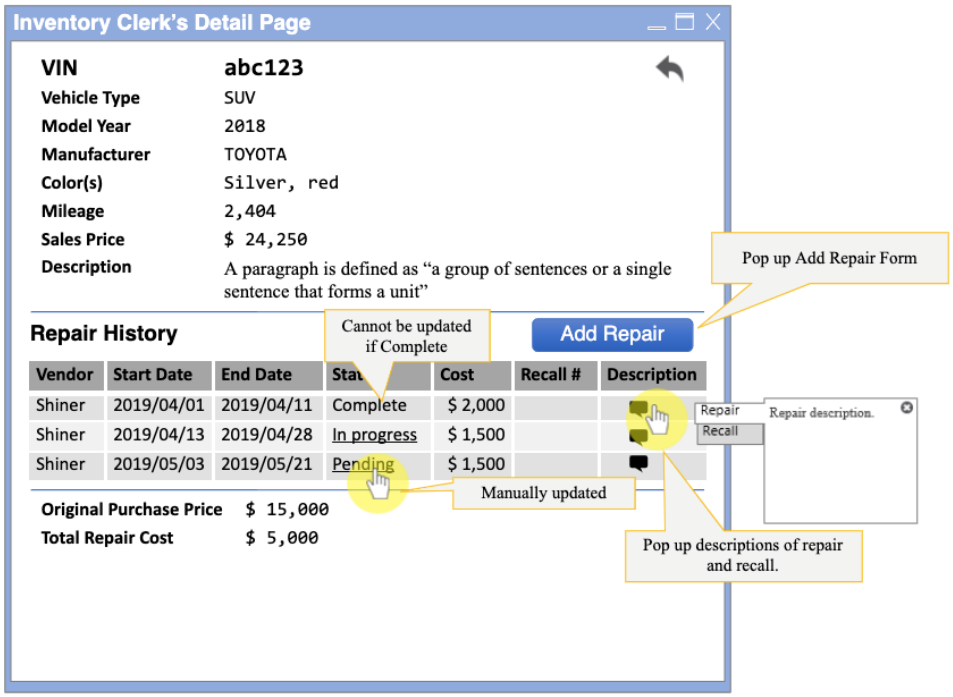

Salespeople's Search Page
Search Car Now!!!
<# Cars> Available
Vehicle Type
(select)
Manufacturer
(select)
Model Year
(select)
Color
(select)
Keyword
(Keywords)
VIN
(Alphanumeric VIN)
Search
Search Result
VIN
Vehicle Type
Model Year
Manufac-turer
Model
Color(s)
Mileage
Sales Price
abc123
SUV
2018
TOYOTA
C-HR
Silver, red
2,404
$ 24,250


Manager's Search Page
Search Car Now!!!
<# Cars> Available
<# Cars> repair (pending or in progress)
Vehicle Type
(select)
Manufacturer
(select)
Model Year
(select)
Color
(select)
Keyword
(Keywords)
VIN
(Alphanumeric VIN)
(View Report)
Seller History
Inventory Age
Average Time in Inventory
Price Per Condition
Repair Statistics
Monthly Sales
Direct to corresponding report directly after click the name of report.
Search
Search Result
Sold Filter
(All Vehicle)
VIN
Vehicle Type
Model Year
Manufac-turer
Model
Color(s)
Mileage
Sales Price
abc123
SUV
2018
TOYOTA
C-HR
Silver, red
2,404
When filtering by unsold or all vehicles, all unsold vehicles will be returned regardless of repair status.


Owner's Search Page
Search Car Now!!!
<# Cars> Available
<# Cars> repair (pending or in progress)
Add Vehicle
Vehicle Type
(select)
Manufacturer
(select)
Model Year
(select)
Color
(select)
Keyword
(Keywords)
VIN
(Alphanumeric VIN)
(View Report)
Seller History
Inventory Age
Average Time in Inventory
Price Per Condition
Repair Statistics
Monthly Sales
Search
Search Result
Sold Filter
(All Vehicle)
VIN
Vehicle Type
Model Year
Manufac-turer
Model
Color(s)
Mileage
Sales Price
abc123
SUV
2018
TOYOTA
C-HR
Silver, red
2,404
$ 24,250

For **Section 1.3:** View Detail





Salespeople Detail Page

VIN	zzz456
Vehicle Type	SUV
Model Year	2017
Manufacturer	TOYOTA
Color(s)	Black
Mileage	2,404
Sales Price	$ 24,250
Description	A paragraph is defined as "a group of sentences or a single sentence that forms a unit"

Sell This Car

Pop up Sales Order Form.

Manager's Detail Page

VIN	abc123
Vehicle Type	SUV
Model Year	2018
Manufacturer	TOYOTA
Color(s)	Silver, red
Mileage	2,404
Sales Price	$ 24,250
Description	A paragraph is defined as "a group of sentences or a single sentence that forms a unit"

Repair History

Vendor	Start Date	End Date	Status	Cost	Recall #	Description
Shiner	2019/04/01	2019/04/11	Complete	$ 2,000		
Shiner	2019/04/13	2019/04/28	In progress	$ 1,500		
Shiner	2019/05/03	2019/05/21	Pending	$ 1,500		

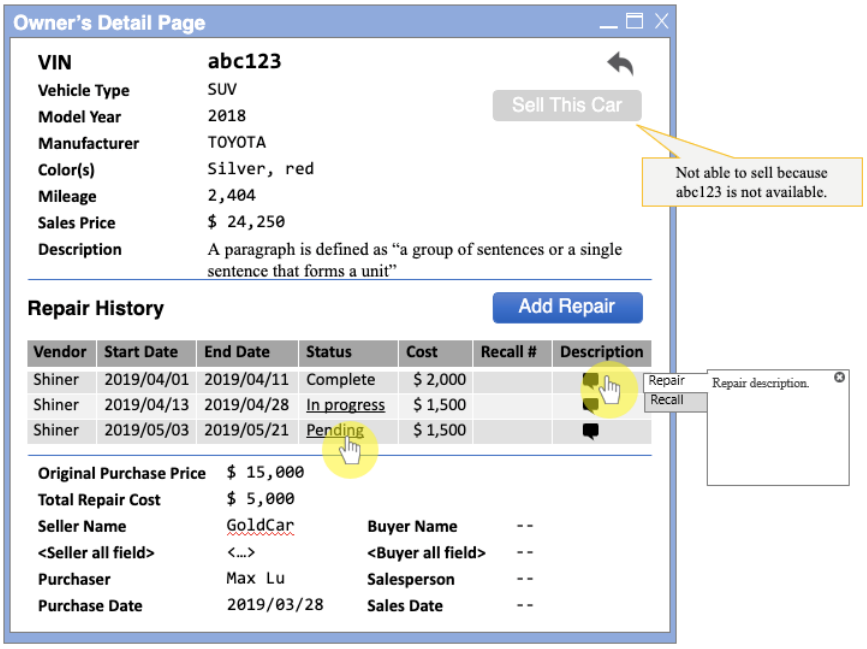
Repair
Recall

Repair description.

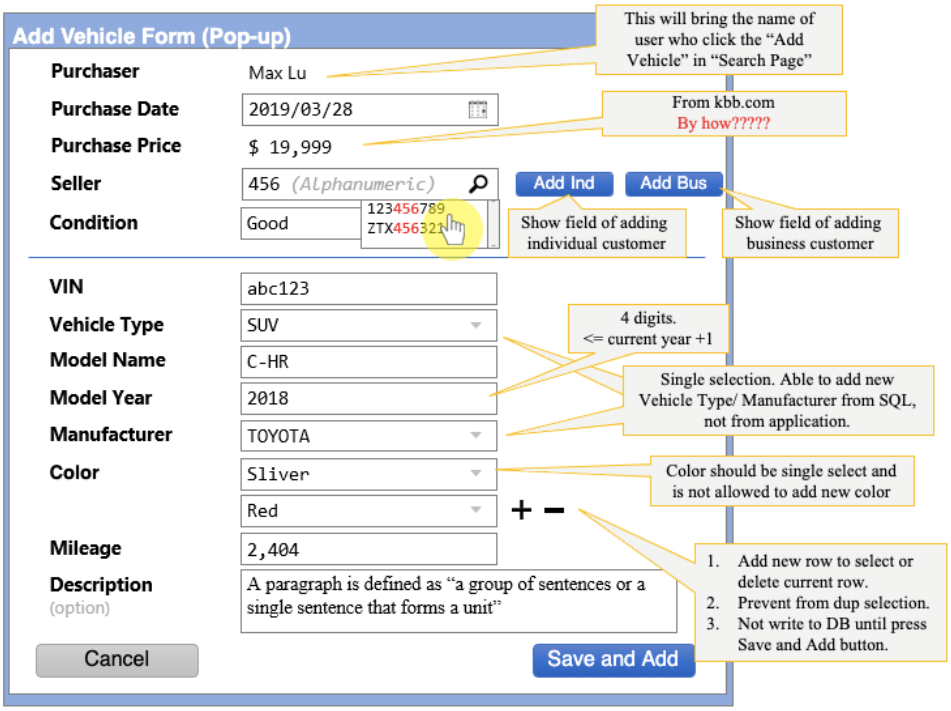
Original Purchase Price	$ 15,000
Total Repair Cost	$ 5,000
Seller Name	
<Seller all field>	
Purchaser	Max Lu
Purchase Date	2019/03/28

Buyer Name	--
<Buyer all field>	--
Salesperson	--
Sales Date	--

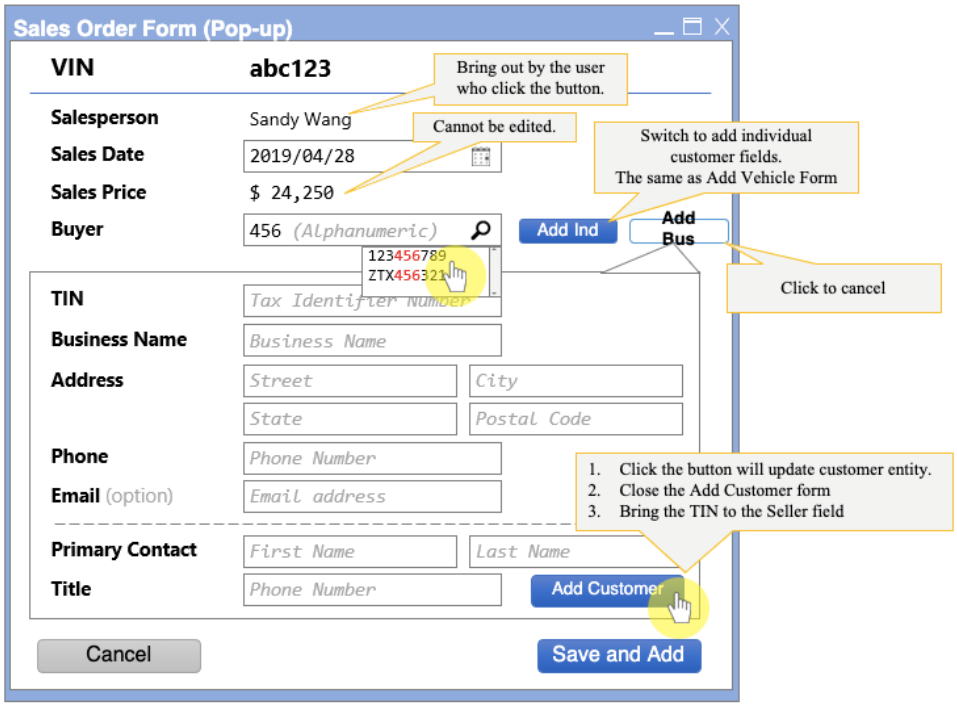
<Discussion>
Always display these fields or only after sold?



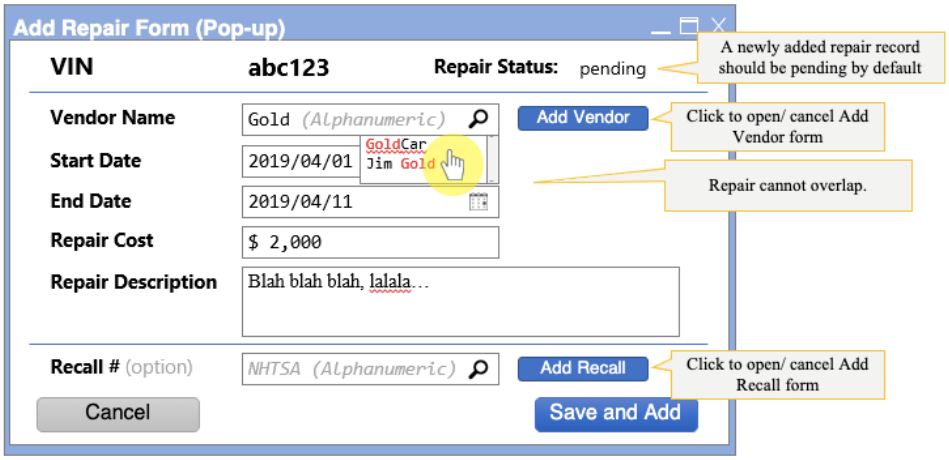
For **Section 1.5:** Add Vehicle



For **Section 1.6:** Add Sales Order



For **Section 1.8:** Add Repair



For **Section 1.11:** View Reports

Primary sort, descending

Secondary sort, descending

Seller History Report

Seller Name	Total # Purchased	AVG Purchase Price	AVG Repairs/ Vehicle
Stark Industries	30	$ 25,000	3.4
GoldCar	30	$ 23,000	6.1
Shiner	14	$ 15,000	5.0
Max Lu	2	$ 17,000	4.1
...			

Highlight with red background if >= 5

Age = TODAY – Purchase Date
Only for unsold units

Inventory Age Report

Vehicle Type	MIN Age (days)	AVG Age (days)	MAX Age (days)
Sedan	14	22	39
Coupe	45	64	94
Convertible	16	26	34
Truck	N/A	N/A	N/A
...			

N/A if that type has no unsold units

Inventory Time = Sales Date – Purchase Date

Average Time in Inventory Report

Vehicle Type	Average Amount of Time in Inventory
Sedan	20
Coupe	40
Convertible	22
Truck	N/A
...	

N/A for no sales records

AVG(Original Purchase Price)

Price Per Condition Report

Vehicle Type	Excellent	Very Good	Good	Fair
Sedan	$ 14,000	$ 12,000	$ 11,000	$ 9,500
Coupe	$ 22,000	$ 0	$ 16,000	$ 14,000
Convertible	$ 0	$ 11,000	$ 10,000	$ 0
Truck	$ 0	$ 0	$ 0	$ 0
...				

$0 if that cell has no purchase record

Repair Statistics Report

Vendor Name	# Repairs Completed	Total $ on Completed Repair	AVG Repairs/ Vehicle	AVG Days to Complete Repair
Shiner	30	$ 50,000	5.0	15
XYZ	14	$ 20,000	3.2	16
ABC	3	$ 5,000	3	26
Bill Fix	1	$ 2,000	1	5
...				

Monthly Sales Report

Sort in descending

Year/Month	# Vehicle Sold	Sales Income	Net Income
2019/05	14	$ 280,000	$ 50,000
Salesperson Name	**# Vehicle Sold**	**Total Sales**	
Sandy Wang	4	$ 120,000	
Andy Lee	4	$ 100,000	
Leo Chen	3	$ 35,000	
...			
2019/04	20	$ 430,000	$ 80,000
2019/03	17	$ 350,000	$ 72,000
...			

Net Income = Sales Price – Purchase Price – Total Repair Cost

All salespeople who have sold cars that month

Primary sort, descending

Secondary sort, descending

Drilldown report