Alicia Sortwell - 12155625

Fathi Aiyyoub – 12151675

COIT12200 – Assignment 3

Contents

[Functional & Non-Functional Requirements 2](#_Toc105177484)

[Functional 2](#_Toc105177485)

[Non-Functional 2](#_Toc105177486)

[Architectural Design 3](#_Toc105177487)

[User Interface Design 3](#_Toc105177488)

[UML Class Diagram 0](#_Toc105177489)

[Database Schemas & Design 0](#_Toc105177490)

[E-R Diagram 0](#_Toc105177491)

[Tables & Attributes 0](#_Toc105177492)

[SQL Script 1](#_Toc105177493)

[Tables with Sample Data 2](#_Toc105177494)

[Testing Scenarios 3](#_Toc105177495)

[Testing Figures 5](#_Toc105177496)

[Figure 1 5](#_Toc105177497)

[Figure 2 6](#_Toc105177498)

[Figure 3 6](#_Toc105177499)

[Figure 4 7](#_Toc105177500)

[Figure 5 7](#_Toc105177501)

[Figure 6 7](#_Toc105177502)

[Figure 7 8](#_Toc105177503)

[Figure 8 8](#_Toc105177504)

[Figure 9 9](#_Toc105177505)

[Figure 10 9](#_Toc105177506)

[Figure 11 10](#_Toc105177507)

[Figure 12 10](#_Toc105177508)

[Figure 13 11](#_Toc105177509)

[Figure 14 11](#_Toc105177510)

[Figure 15 12](#_Toc105177511)

[Figure 16 12](#_Toc105177512)

[Figure 17 13](#_Toc105177513)

[Figure 18 13](#_Toc105177514)

# Functional & Non-Functional Requirements

### Functional

* A user shall be able to add a new customer in the system that captures and stores the customer’s first name, last name, address and phone number. Once entered, the system will automatically assign a customer an ID number.
* A user shall be able to update a customer’s address and phone number
* A user shall be able to search a customer in the database by name
* A user shall be able to search a customer in the database by phone number
* A user shall be able to browse customers existing in the database.
* A user shall be able to add a vehicle to a new or existing customer that captures the vehicle’s rego number, brand, model, year of manufacturing and kms on odometer. Once entered, the system will automatically assign the vehicle an ID number.
* A user shall be able to add a service session into the system against a customer’s rego number, capturing service description, date and price. Once entered, the system will automatically assign the service an ID number.
* A user shall be able to browse all vehicles existing in the database
* A user shall be able to search a service booking by a rego number
* A user shall be able to cancel a service booking
* The system shall generate a list of all service details when specified by the user, which shall be sorted by price from lowest to highest
* A manager user shall be able to view service statistical data that includes the minimum, maximum and average price of all services offered, the number of vehicles serviced by make and a simple bar chart to show the top 3 brands by number of services

### Non-Functional

*Reliability & Availability*

* The system shall implement error handling to ensure the correctness of the data implemented
* The system shall be available during normal working hours (9am – 6pm)
* Response time will be consistent with all user requests

*Security*

* The database will be secured by a username and password for authentication. In order to access the database and the platform to be used, correct URL, username and password need to be used.

*Compatibility*

* The system will run on Netbeans 12+, using JavaFX 11+, utilising XAMPP or MySQL 8+.

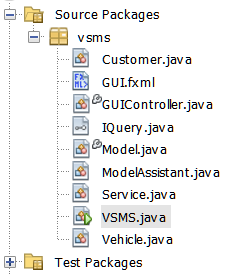
*Usability*

* The system should have a GUI that is in line with the design principles and promoting ease-of-use for all staff. This includes having clear and appropriate labels and sectioned areas that are easy to follow. Buttons are also clearly labelled.

*Capacity*

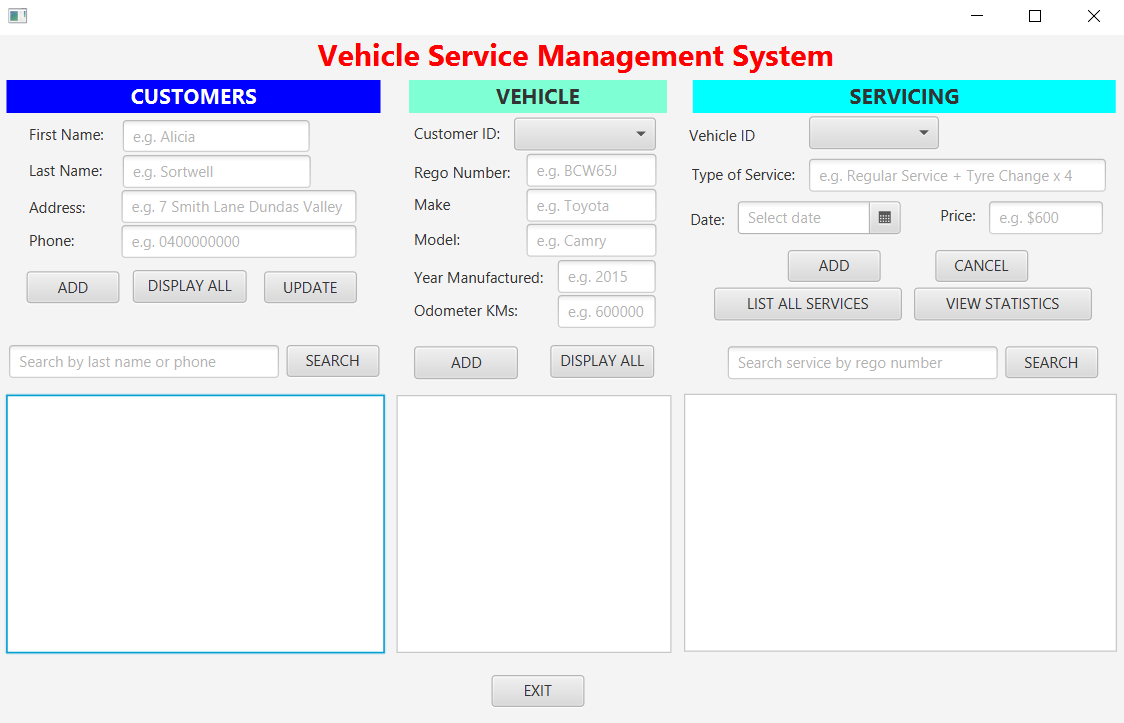
* The maximum capacity, as per the maximum capacity for SQL server, is 16 TBs of data, which will be able to 32,767 files.

# Architectural Design



# User Interface Design

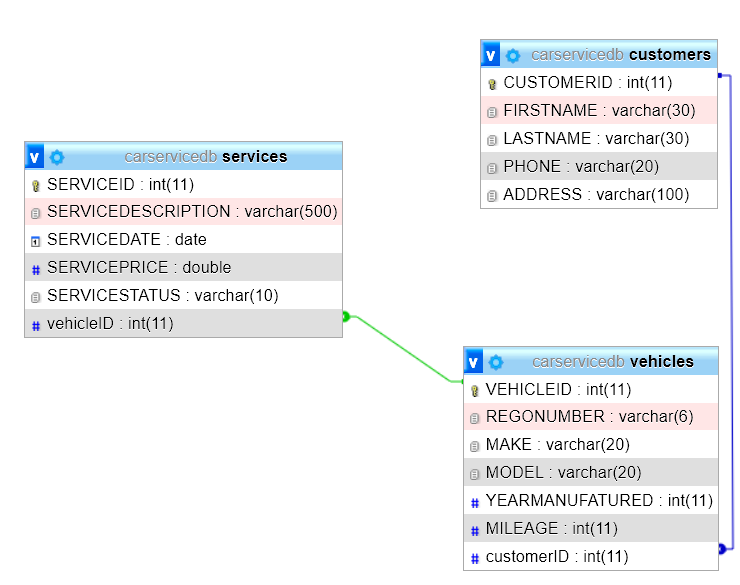
Our user interface is designed to be easy to use. It is split into three areas, from left to right, in order of how the information­­­­ needs to be entered into the system, which makes sense for staff. The areas are contained amongst themselves to make the GUI easy to follow. The customer section allows for entry of customer details and additional functions, the vehicle section allows for vehicle details and functions, and the services section allows for service details and functions. This clear manner should allow for ease of use and all buttons are self-explanatory in their sections, with prompt-text in each of the fillable fields to, again, allow for easy use of the system.



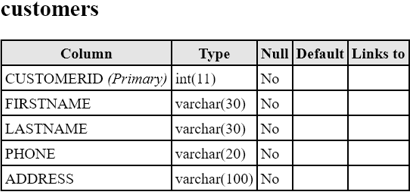
# UML Class Diagram

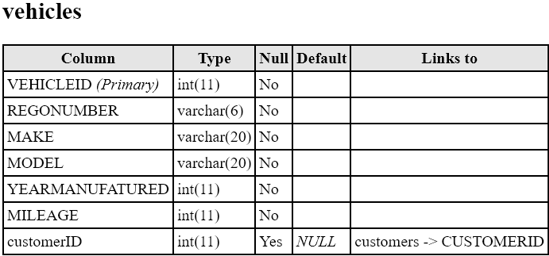
# Database Schemas & Design

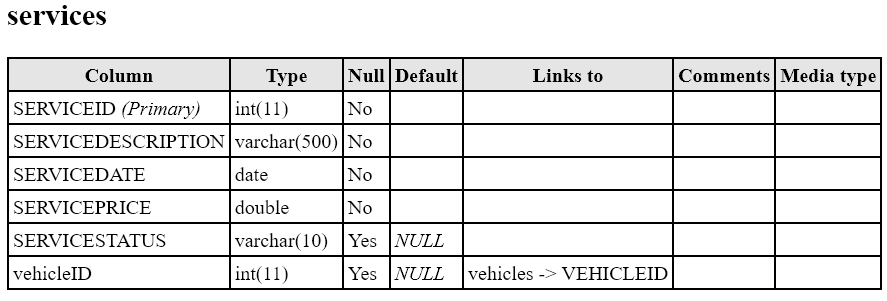
### E-R Diagram



### Tables & Attributes







### SQL Script

drop database if exists CarServiceDB;

----------------------------------------------------

CREATE DATABASE CarServiceDB;

USE CarServiceDB;

CREATE TABLE CUSTOMERS

(

CUSTOMERID INT NOT NULL,

FIRSTNAME VARCHAR (30) NOT NULL,

LASTNAME VARCHAR (30) NOT NULL,

PHONE VARCHAR (20) NOT NULL,

ADDRESS VARCHAR(100) NOT NULL,

PRIMARY KEY(CUSTOMERID)

);

INSERT INTO CUSTOMERS (CUSTOMERID, FIRSTNAME, LASTNAME, PHONE, ADDRESS) VALUES

(1,'John','Smith','0422731711','Parramatta'),

(2,'Fathi','AIYYOUB','0455019234','Greenacre');

CREATE TABLE VEHICLES

(

VEHICLEID INT NOT NULL,

REGONUMBER VARCHAR (6) NOT NULL,

MAKE varchar (20) NOT NULL,

MODEL VARCHAR (20) NOT NULL,

YEARMANUFATURED INT NOT NULL,

MILEAGE INT NOT NULL,

customerID int,

PRIMARY KEY(VEHICLEID),

Foreign Key (CUSTOMERID) REFERENCES CUSTOMERS (CUSTOMERID)

);

INSERT INTO VEHICLES (VEHICLEID, REGONUMBER, MAKE, MODEL, YEARMANUFATURED, MILEAGE, CUSTOMERID) VALUES

(1,'DER432','TOYOTA','AURION','2015', '85000', 1),

(2,'DEW444','MERCEDES','G63','2020', '2697', 2);

CREATE TABLE SERVICES

(

SERVICEID INT NOT NULL,

SERVICEDESCRIPTION VARCHAR (500) NOT NULL,

SERVICEDATE DATE NOT NULL,

SERVICEPRICE DOUBLE NOT NULL,

SERVICESTATUS VARCHAR(10),

vehicleID int,

PRIMARY KEY(SERVICEID),

Foreign Key (VEHICLEID) REFERENCES VEHICLES(VEHICLEID)

);

INSERT INTO SERVICES (SERVICEID, SERVICESTATUS, SERVICEDESCRIPTION, SERVICEDATE, SERVICEPRICE, vehicleID) VALUES

(1, 'Completed', 'oil change', '2022-03-25', 550.30 , 1),

(2, 'Completed', 'pink slip, oil change, filter change', '2022-05-25', 800.30 , 2);

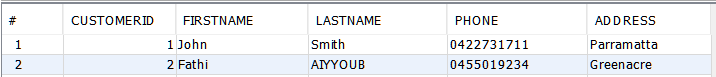
select \* from VEHICLES;

select \* from customers;

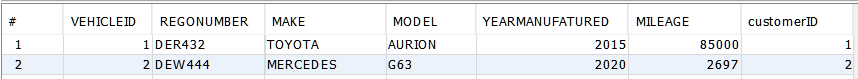
select \* from SERVICES;

### Tables with Sample Data

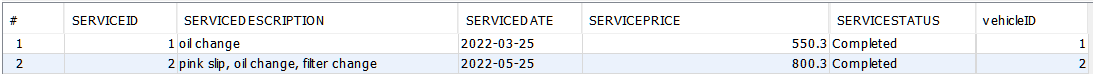
**CUSTOMER**



**VEHICLE**



**SERVICES**

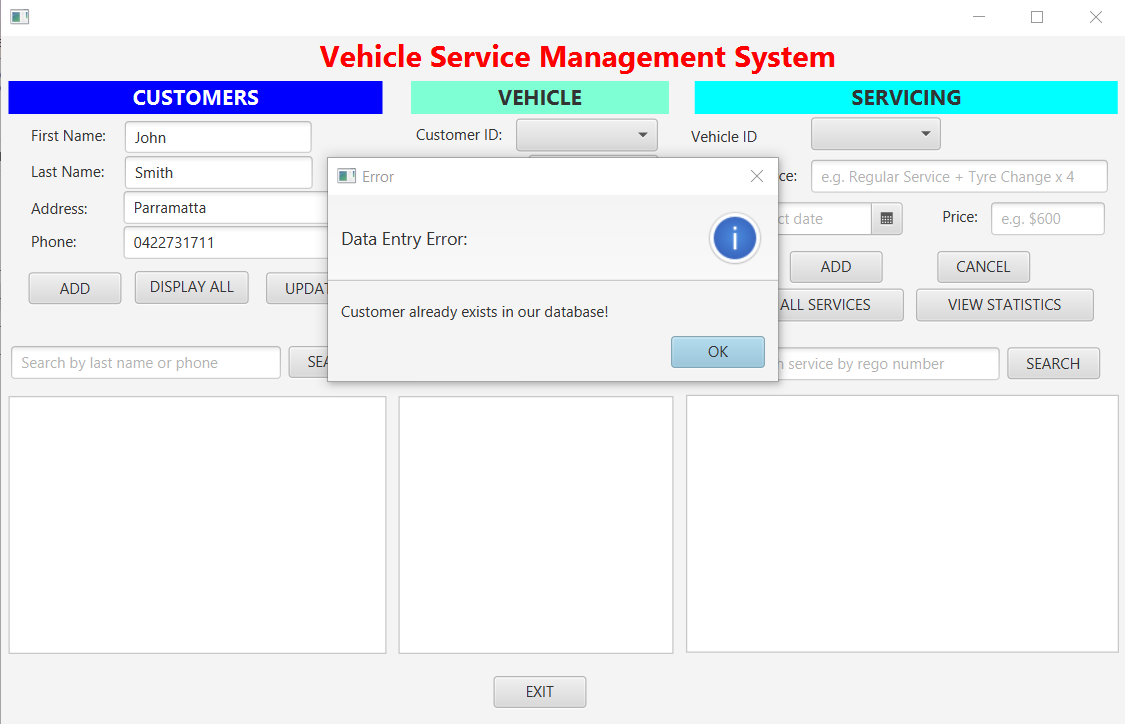


# Testing Scenarios

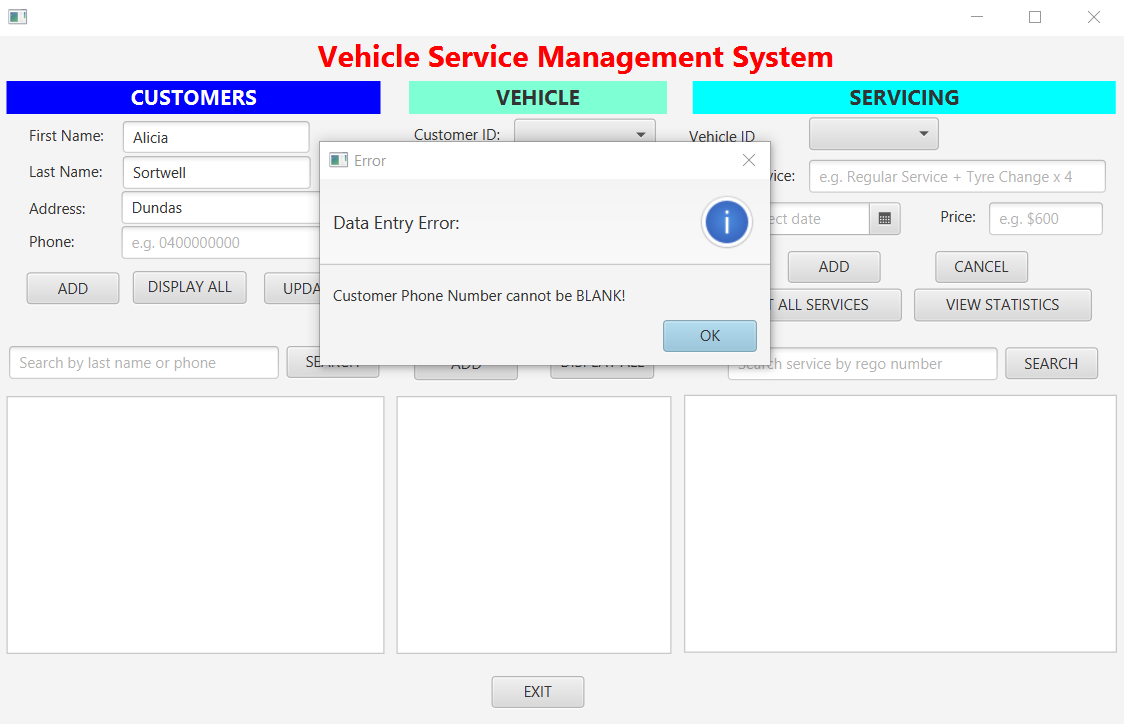
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Reference | Functionality | Input List | Expected Result | Actual Result | Evidence |
| 1 | Adding new customer with same details as already existing customer | John, Smith, 0422731711, Parramatta | On press of button ‘ADD’ error message appears saying customer already exists in database | Pop-up displayed correctly | [Figure 1](#_Figure_1_1) |
| 2 | Adding new customer without filling out all requirements | Alicia, Sortwell, Dundas | On press of button ‘ADD’ error message appears saying blank element cannot be blank | Pop-up displayed correctly | [Figure 2](#_Figure_2) |
| 3 | Adding new customer | Alicia, Sortwell, Dundas, 0400000000 | On press of button ‘ADD’, customer is added to the database | Customer added to database successfully | [Figure 3](#_Figure_3) |
| 4 | Displaying all customers in database | Click ‘Display All’ button | Displays all rows from database onto GUI | All customers displayed correctly | [Figure 4](#_Figure_4) |
| 5 | Updating customer address | Click customer ID 1 in from list of customers in display, and change Parramatta to Manly | On press of button ‘UPDATE’, customer is updated in database | Customer updated successfully | [Figure 5](#_Figure_5) |
| 6 | Updating customer phone | Click customer ID 3 in from list of customers in display, and change 0400000000 to 0410000000 | On press of button ‘UPDATE’, customer is updated in database | Customer updated successfully | [Figure 6](#_Figure_6) |
| 7 | Search by customer last name | Search ‘Smith’ and click SEARCH button | On click of ‘Search’ button, list below should display the rows from database that match that name | Search results displayed correctly | [Figure 7](#_Figure_7) |
| 8 | Search by customer phone | Search ‘0410000000’ and click SEARCH button | On click of ‘Search’ button, list below should display the rows from database that match that phone | Search results displayed correctly | [Figure 8](#_Figure_8) |
| 9 | Add new vehicle into database against customer without filling in all fields | Select: 1, Enter: BCW65S, Toyota, Camry, \*leave blank\*, 600000 | On click of ‘ADD’ button, error message should display saying element cannot be blank | Error messaged appeared successfully | [Figure 9](#_Figure_9) |
| 10 | Add new vehicle into database against customer | Select: 1, Enter: BCW65S, Toyota, Camry, 2015, 600000 | On click of ‘ADD’ button, vehicle added to database | Vehicle successfully added to database | [Figure 10](#_Figure_10) |
| 11 | Displaying all vehicles in database | Click ‘DISPLAY ALL’ button in vehicle section | Displays all rows from database for vehicles in GUI | All vehicles displayed correctly | [Figure 11](#_Figure_11) |
| 12 | Add new service to database without filling all fields | Select Vehicle ID: 2, Enter: \*leave blank\*, 3/6/2022, $600 | On click of ‘ADD’ button, error message should display saying element cannot be blank | Error message appeared successfully | [Figure 12](#_Figure_12) |
| 13 | Add new service to database and having $ sign in price field | Select Vehicle ID: 2, Enter: Oil Change, 3/6/2022, $600 | On click of ‘ADD’ button, error message should display saying element can only be a number | Error message appeared successfully | [Figure 13](#_Figure_13) |
| 14 | Adding new service to database | Select Vehicle ID: 2, Enter: Oil Change, 3/6/2022, 600 | On click of ‘ADD’ button, service added to database | Service successfully added to database | [Figure 14](#_Figure_14) |
| 15 | Display all services in database in price ascending | Click ‘LIST ALL SERVICES’ button in servicing section | Displays all rows from database for services in GUI by price ascending | All services displayed correctly | [Figure 15](#_Figure_15) |
| 16 | Cancel a service | Click on Service ID 3 in list, then click ‘CANCEL’ button | On button press, updates Service Status to Cancelled and changes price to 0 in database | Service Status and price updated successfully | [Figure 16](#_Figure_16) |
| 17 | Search service by rego number | Enter DEW444 and click SEARCH button | On click of ‘Search’ button, list display shows the services associated with that rego number | Search results display correctly | [Figure 17](#_Figure_17) |
| 18 | View statistics about services | Click VIEW STATISTICS button | On click of button, dialog pop-up displays showing statistics regarding to services | Statistics display correctly | [Figure 18](#_Figure_18) |
| 19 | Exit program | Click EXIT button at bottom of screen | On click of button, pop-up appears confirming you want to exit and you click OK to exit | Pop-up displays and program exists | NA |

# Testing Figures

## Figure 1



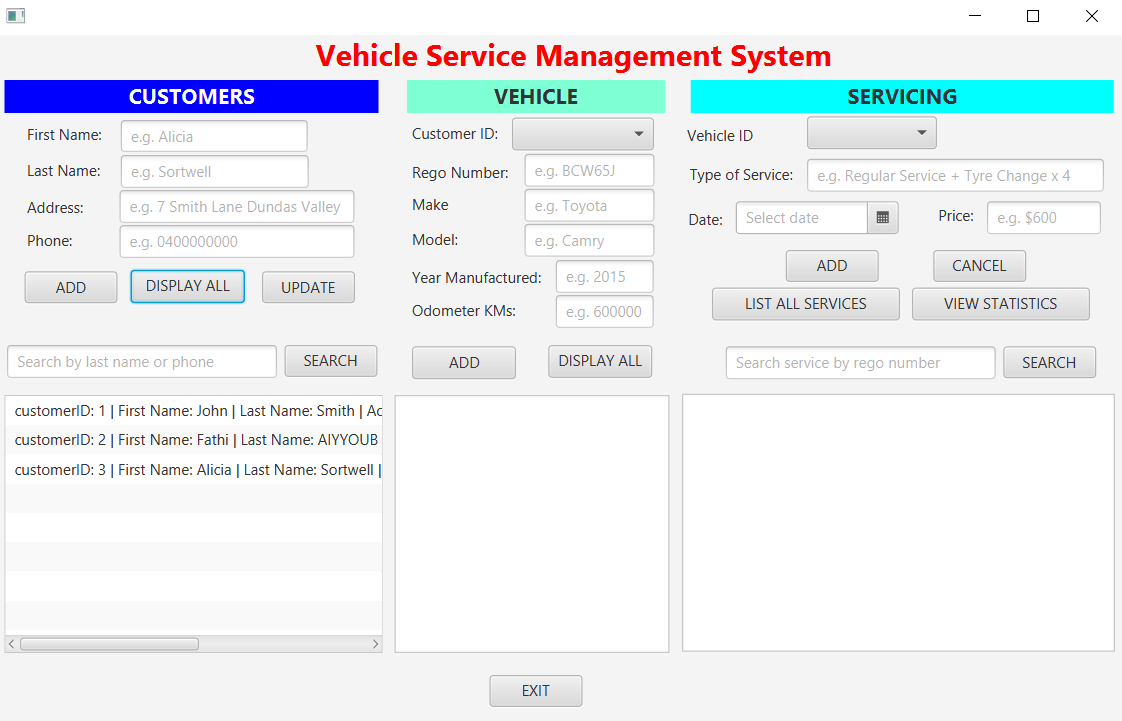
## Figure 2



## Figure 3



## Figure 4



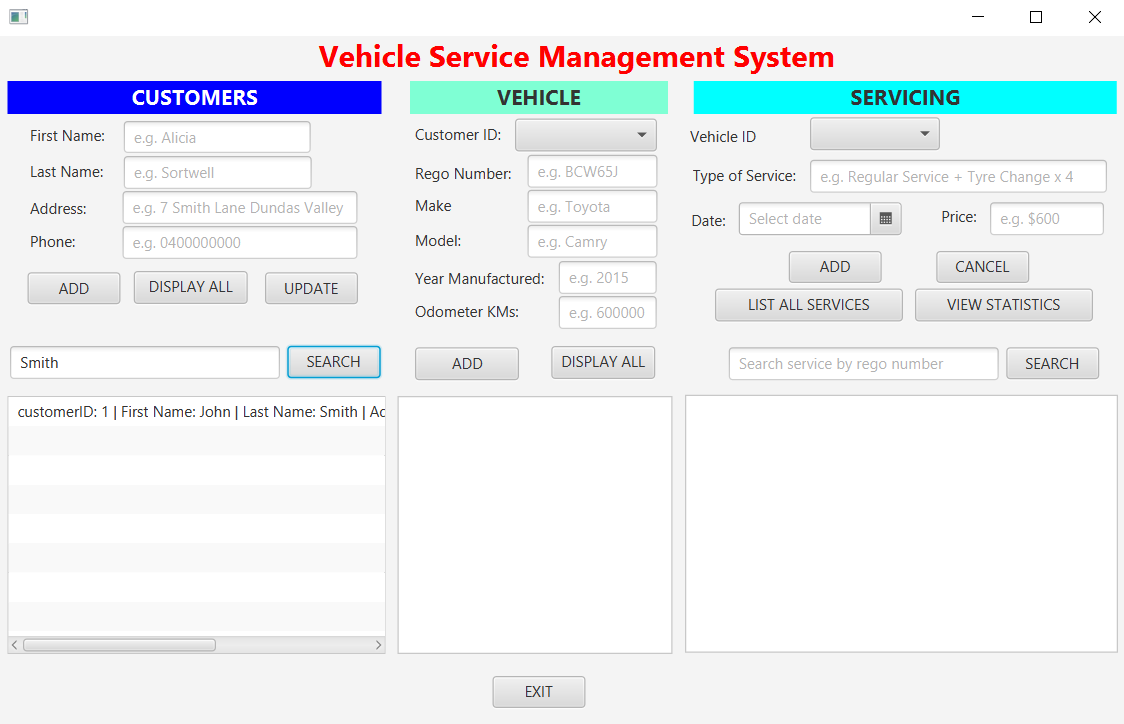
## Figure 5



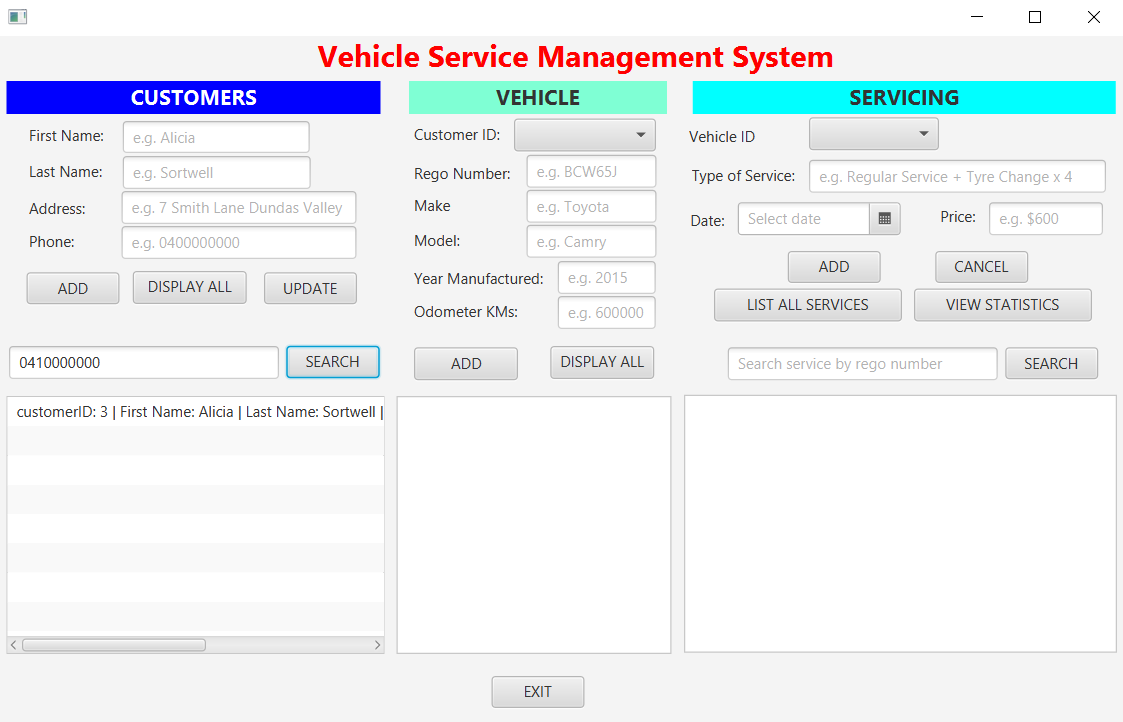
## Figure 6



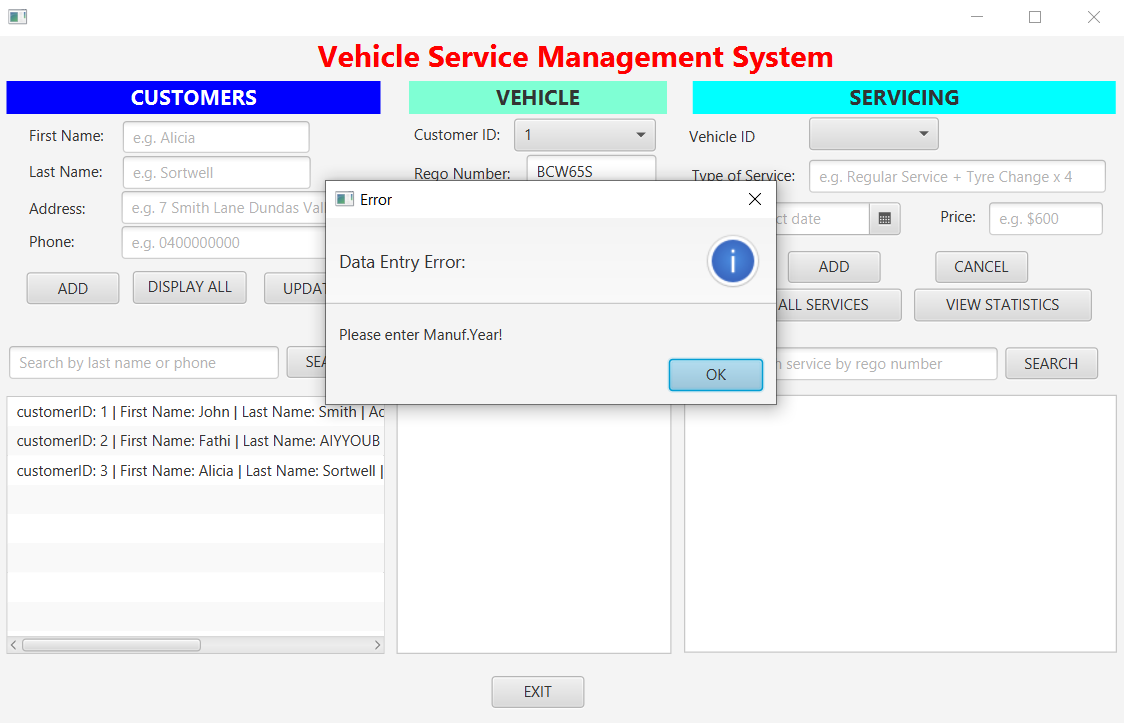
## Figure 7



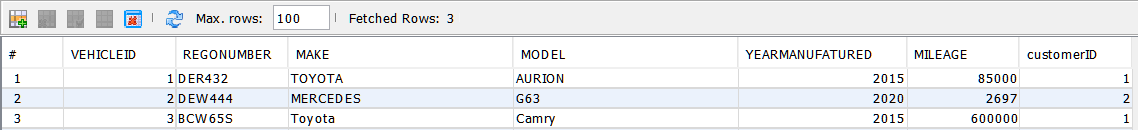
## Figure 8



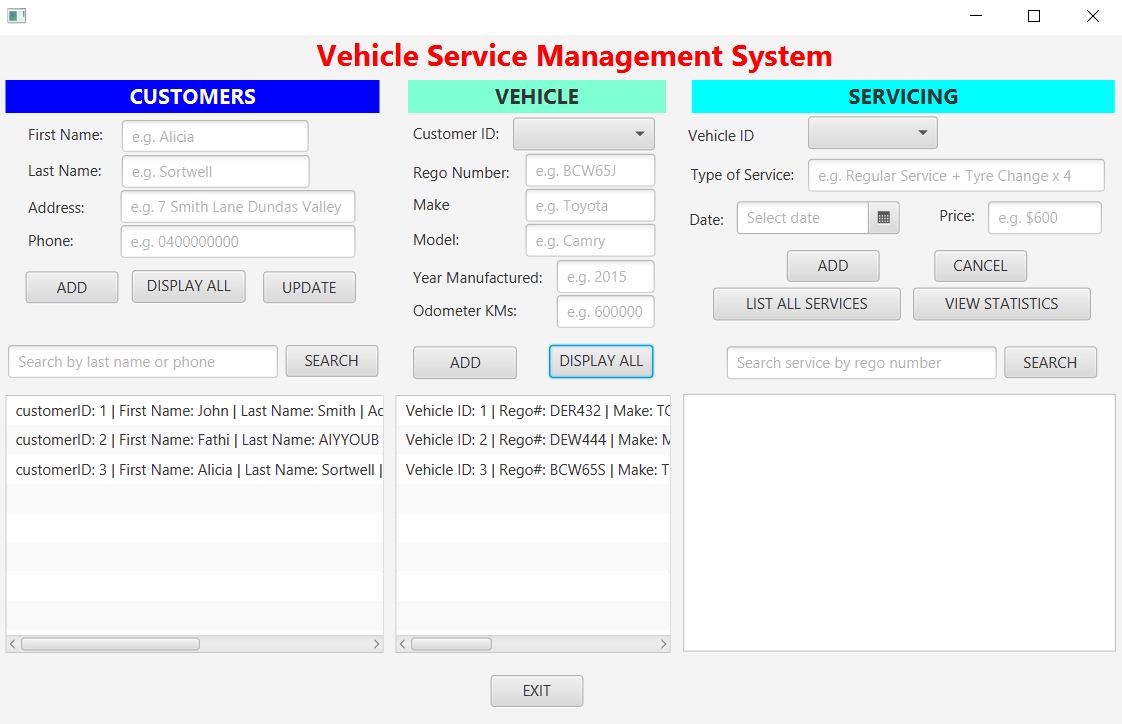
## Figure 9



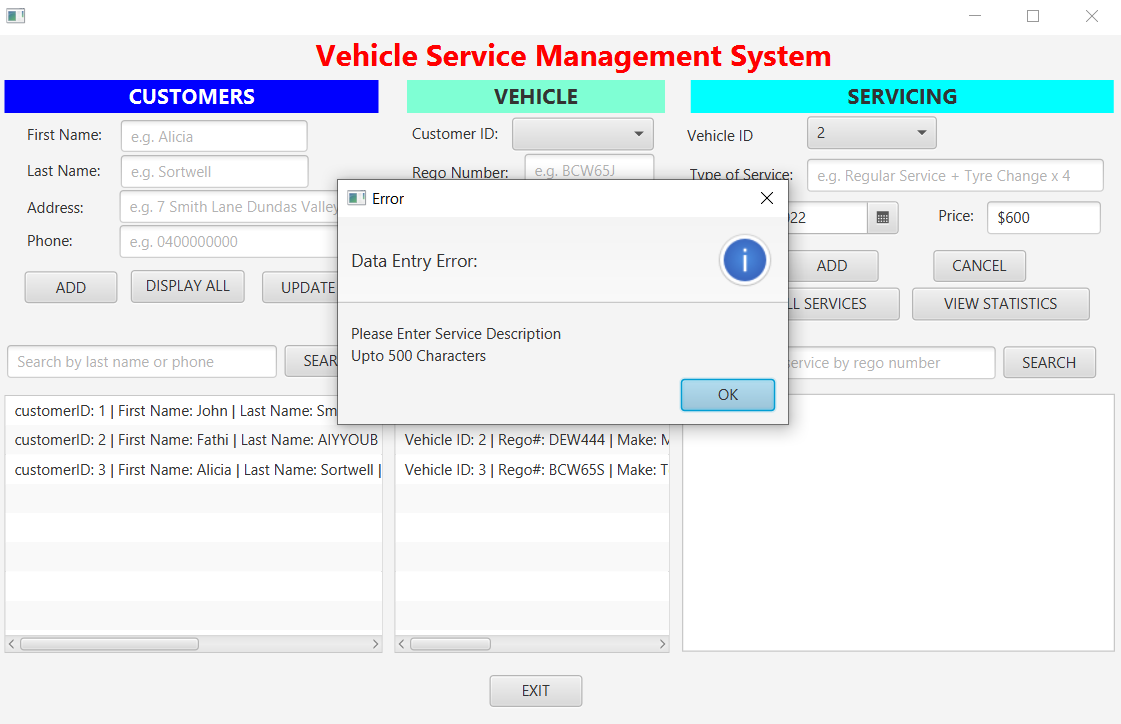
## Figure 10



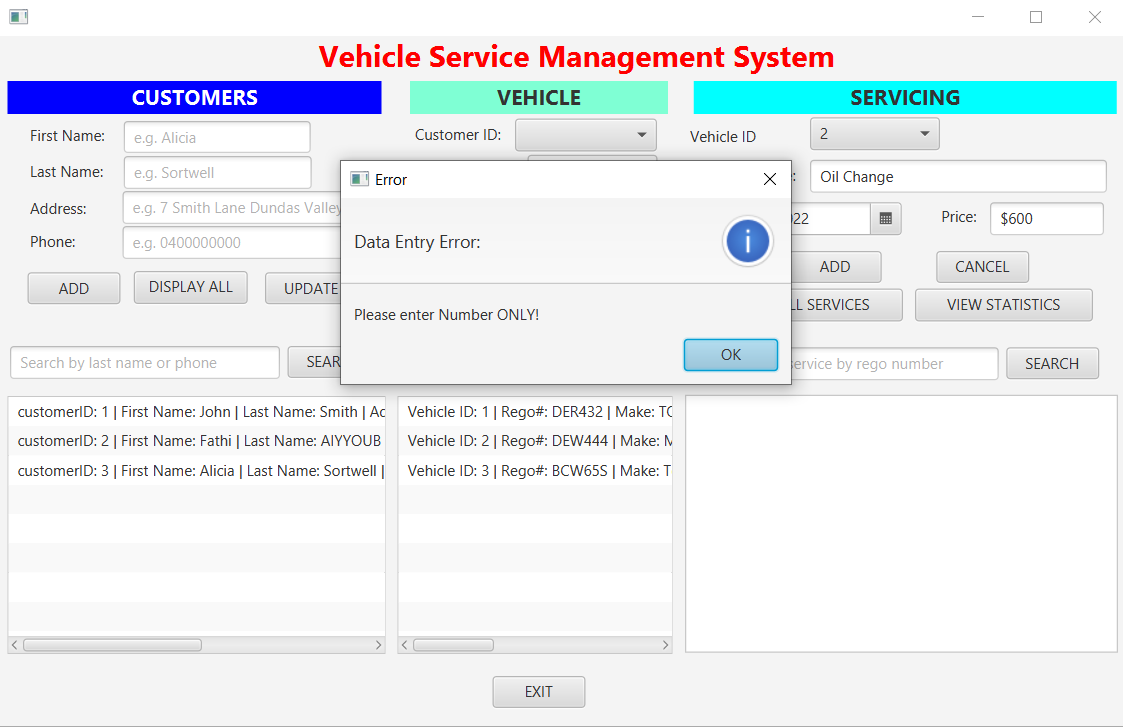
## Figure 11



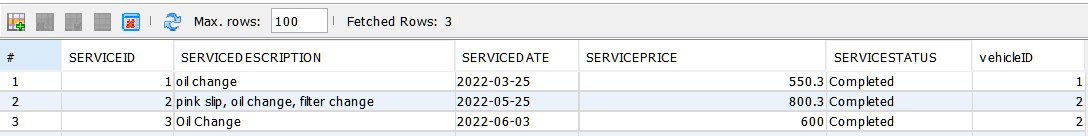
## Figure 12



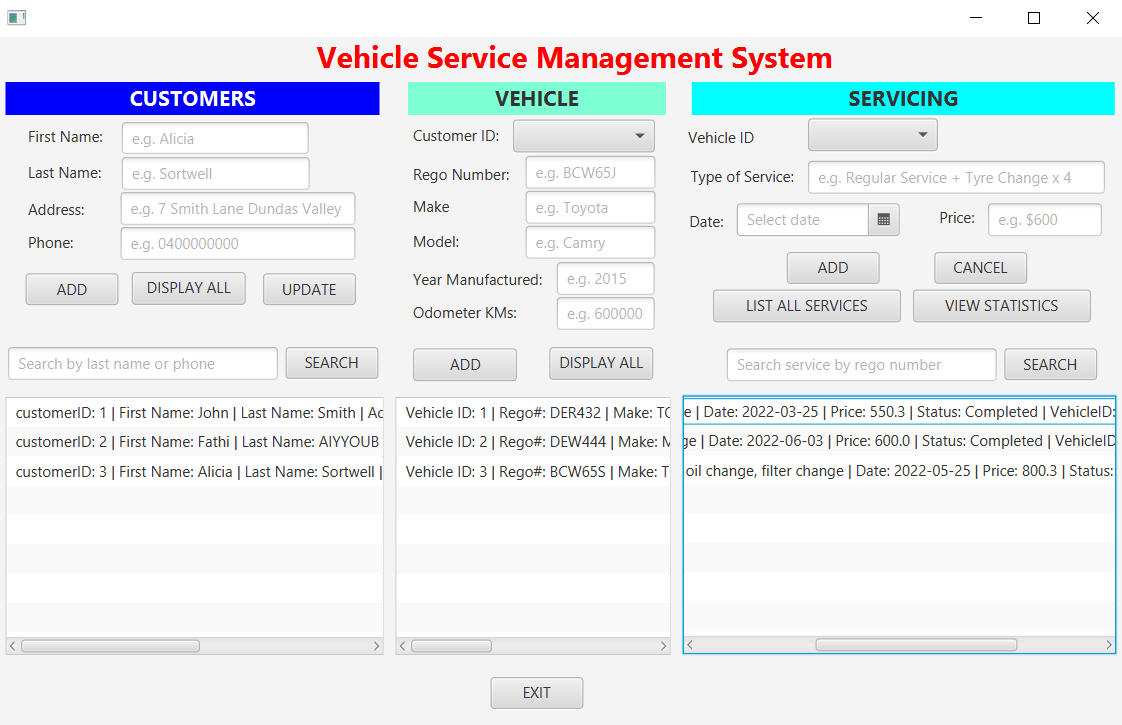
## Figure 13



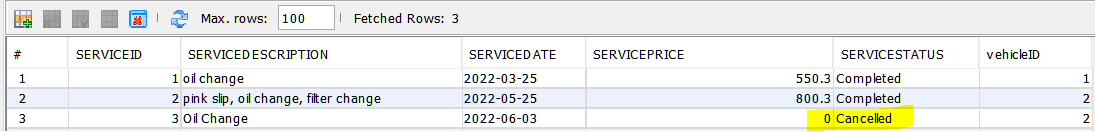
## Figure 14



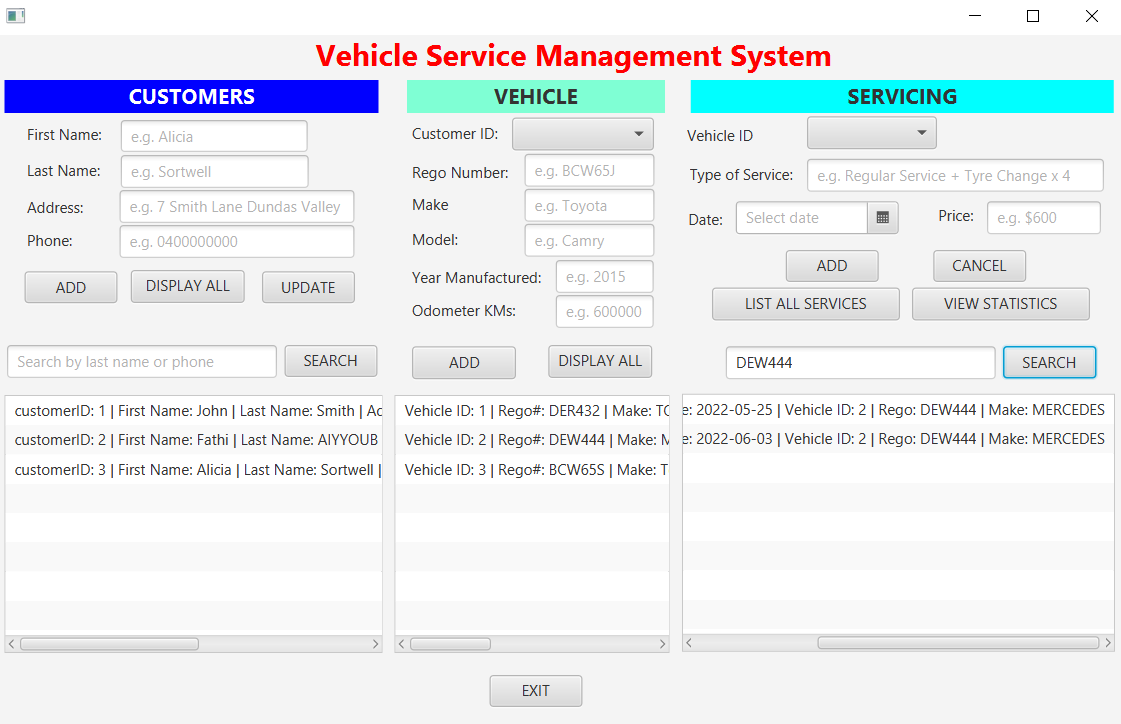
## Figure 15



## Figure 16



## Figure 17



## Figure 18

