**TEAM PROJECT – Vehicle Finder System**

COMP 246 - 001

**Professor: Mohamed Khan**

June 11th, 2023

**GROUP 2**

Gursimran - 301273109

Hong Wang - 301243509

Leonardo dos Santos Nunes - 301263098

Ling Lin - 301292283

Sing Cheung Tin - 301246562

Vivian Enwerem - 301318591

Augusto Schmitt - 301245327

Yuwen Hung - 301253941

Contents

[Part A: Project Scope and Requirements 3](#_Toc613864316)

[Section 1: Problem Statement 4](#_Toc1467873052)

[1.1.a. Problem & Need 4](#_Toc27719366)

[1.1.b. List of Capabilities and Benefits 4](#_Toc1365537712)

[1.2 Identify the stakeholders and their roles 4](#_Toc1169947768)

[1.3 Identify the sub-systems of your application (What are its functional components) 5](#_Toc492244892)

[1.4 Who are the intended users of the SRS documentation. 5](#_Toc1608507567)

[Section 2: General Overview Modelling 5](#_Toc1030485794)

[2.1 Context Flow Diagram (CFD)--- WHAT 6](#_Toc1524937944)

[Section 3: Requirements - functional and non-functional 6](#_Toc736534994)

[3.1 Non- functional Requirements 6](#_Toc2115971685)

[3.2 Functional Requirements 7](#_Toc1106080954)

[3.2.1 User Account Subsystem 7](#_Toc697864222)

[3.2.2 Inventory subsystem 8](#_Toc1244248973)

[3.2.3 Search engine management subsystem 8](#_Toc1033928151)

[3.2.4 Communication subsystem (Live Chat, Mail Communication such as sending notification, order confirmations, or responses to user inquiries) 9](#_Toc1228483609)

[3.2.5 Reporting subsystem (Reporting Subsystem such as pre-defined report templates, or custom report generation capabilities) 10](#_Toc2055238499)

[Section 4: Domain Class Diagram 11](#_Toc508670247)

[4.1 List of classes 11](#_Toc620028065)

[4.2 Diagram 11](#_Toc1168162860)

[Section 5: ERD 12](#_Toc1362174268)

[Section 6: Systems Sequence Diagrams 13](#_Toc1755065163)

[6.1 Inventory Subsystem (use case: create vehicle inventory) 14](#_Toc164197721)

[6.2 Search (Use case: enter information and retrieve information) 14](#_Toc473997538)

[Section 7: Object State Machine Diagrams 15](#_Toc879938516)

[7.1 Potential Product Object State Machine Diagram 15](#_Toc1217844634)

[7.2 Communication Object State Machine Diagram 15](#_Toc1698043115)

[Section 8: Technologies 16](#_Toc1949017466)

[8.1 Web Application: HTML, CSS, JavaScript, C#, ASP.NET MVC Core 16](#_Toc2116075358)

[8.2 Database: Oracle SQL 16](#_Toc232653206)

[Section 9: Project Management (Gantt Chart) 16](#_Toc36281218)

[Part B: Software Design Architecture 17](#_Toc1759001940)

[Section 1: Requirements Edits to Part A 18](#_Toc416246221)

[1.1 Corrections on Domain Class Diagram (section 4.2) 18](#_Toc829463157)

[1.2 ERD (section 5) 18](#_Toc1356803764)

[Section 2: Overview Model 18](#_Toc668106139)

[2.1 Intended users of the SDD document 18](#_Toc1043942614)

[2.2.1 SRS Context Flow Diagram --CFD (What diagram) 19](#_Toc186153090)

[2.2.2 Architectural Context Diagram—ACD (HOW diagram) 20](#_Toc2147052379)

[Section 3: Modularization 20](#_Toc1305473271)

[3.1 Partition the analysis model 21](#_Toc41781809)

[3.1.1 User Account Subsystem 21](#_Toc683111149)

[3.1.2 Inventory subsystem 21](#_Toc1932976861)

[3.1.3 Search engine management subsystem 22](#_Toc1577584158)

[3.1.4 Communication subsystem 24](#_Toc1941833799)

[3.2 Class Responsibility Collaboration (CRC) 24](#_Toc1801216669)

[3.2.1 User Account subsystem 25](#_Toc34936047)

[3.2.2 Inventory subsystem 25](#_Toc965266020)

[3.3.3 Search Engine management subsystem 26](#_Toc793394383)

[3.2.4 Communication subsystem (corrected) 26](#_Toc922825265)

[Section 4: Framework M(odel) V(iew) C(ontroller) 32](#_Toc1551655011)

[4.1 MVC pattern diagram to include 32](#_Toc503587829)

[4.1.1 Login Subsystem 33](#_Toc1317264142)

[4.1.2 Search Subsystem 33](#_Toc95839112)

[4.1.3 Products Design Subsystem 34](#_Toc1047725373)

[4.2 Full Sequence diagrams 36](#_Toc2144132111)

[4.2.1 Inventory Subsystem (use case: create vehicle inventory) 36](#_Toc1010312570)

[4.2.2. Use cases: Enter information and Retrieve information (Search Subsystem) 37](#_Toc1597904008)

[4.3 State Machine Diagrams 37](#_Toc186707748)

[4.3.1 Objects: use case: Create vehicle inventory 37](#_Toc1811928911)

[4.3.2 Objects: use case: Send Mail Communication – Notifications 38](#_Toc1757642813)

[Section 5: Data Layer 38](#_Toc194030356)

[5.1. Database schema. 38](#_Toc379266961)

[5.2 Technology List Update 38](#_Toc1404999566)

[Section 6: Gantt chart update 39](#_Toc1661723117)

[Part C: System Design Documents 39](#_Toc2090086326)

[Section 1: Corrections to Design Specifications Part B 39](#_Toc1898789902)

[Section 3: UI/UX design 42](#_Toc1220606379)

[3.1 Sign In (login) 43](#_Toc1639575423)

[Section 4: High Level Component / Deployment Diagram 44](#_Toc83921027)

[Section 5: Updated Gantt Chart for part C 45](#_Toc416106668)

# Part A: Project Scope and Requirements

## Section 1: Problem Statement

### 1.1.a. Problem & Need

The process of buying a car can be challenging due to the research process of finding an available model and the communication aspect with multiple dealerships. Finding a car is not the end of the process, as closing a deal is also a lengthy process of exchanging information with multiple dealerships. The workload and waiting times put stress on the potential buyers, who might give up because of how inefficient this process is.

The solution proposed is to consolidate a good number of Canadian dealerships that have stores in big Canadian metropolitan cities. These dealerships should also cover a good range of vehicle brands, and the system would provide a bridge between the potential buyer (user) and the dealerships. The system will provide a centralized platform to research models, their availability, and critical payment information based on user’s profile information. The platform will also serve to keep track of your deals, scheduling meetings with dealerships and keeping track of important information for the process. As each dealership can inform the application of how they calculate downpayment and weekly fees, the user would have an easier time setting their expectations and setting up a meeting with the dealership that is best for them.  
  
Ultimately the system serves as a bridge between potential buyers and sellers. The system is a facilitator through its functionalities.

### 1.1.b. List of Capabilities and Benefits

(i) Capabilities

1. User account registration
2. User account self-management (password reset, user account information update)
3. Display new and used vehicles in stock.
4. Display new vehicle estimated arrival date.
5. Notification which includes notifying user vehicle stock availability, vehicle estimated time of arrival, and vehicle to collect.

(ii) Benefits

1. Save time. Users can compare vehicles in brand, model, price, function, and feature at one system.
2. Vehicle inventory status
3. Vehicle arrival tracking
4. Vehicle arriving notifications.

### 1.2 Identify the stakeholders and their roles

|  |  |
| --- | --- |
| Stakeholder | Role |
| Vehicle seeker | A person, an organization, or a company who is looking for a new or used vehicle. |
| Dealership | A company that provides product information, quantity of inventory, and payment plan (instalment or full payment) |
| System administrator | Setup web server, database server, and maintain the servers. |
| Customer services | Tier 1 user support. E.g. user account, vehicle finder, and billing issues. |
| Project manager | Project management which includes kickoff meeting, project agenda, project timeline, and project budget. |
| QA test engineer | Test the system on the staging and production servers when there are any changes to the system. |

### Identify the sub-systems of your application (What are its functional components)

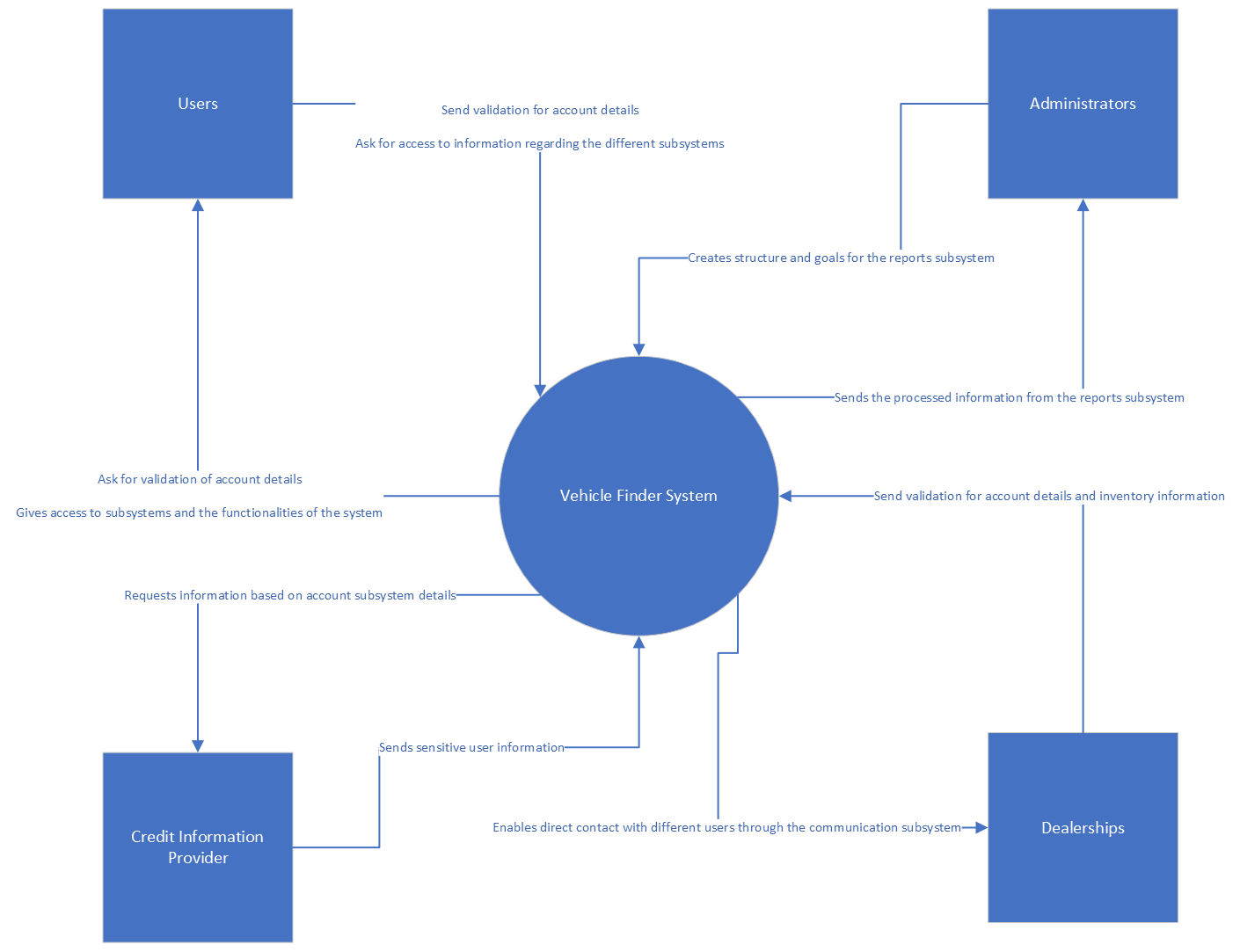
1. User account subsystem
2. Inventory subsystem
3. Search engine management subsystem
4. Communication subsystem (Live Chat, Mail Communication such as sending notification, order confirmations, or responses to user inquiries)
5. Reporting subsystem (Reporting Subsystem such as pre-defined report templates, or custom report generation capabilities)

### Who are the intended users of the SRS documentation.

1. Software developer
2. QA test engineer

## Section 2: General Overview Modelling

### 2.1 Context Flow Diagram (CFD)--- WHAT



**Picture 1 –** *Context Flow Diagram of the Vehicle Finder System*

## Section 3: Requirements - functional and non-functional

### 3.1 Non- functional Requirements

|  |  |  |
| --- | --- | --- |
| NFR# | Name | Description |
| NFR001 | Performance | Requirements of the software system can manage heavy traffic and large volumes of data efficiently, with a focus on speed and accuracy. |
| NFR002 | Security | Ensure all user and dealership data have strong security and prevent unwanted access. |
| NFR003 | Compatibility | The application should work with all operating systems and seamlessly connect with any device. |
| NFR004 | Scalability | When a campaign or promotion is launched that influx of users to the site at the same time, the system must be able to accommodate this growth. As a result, the function of scalability must be designed from the software system design beginning. |
| NFR005 | Usability | The application must be easy to learn and use for the target user to increase user efficiency and satisfaction. |
| NFR006 | Maintainability | The system is designed to be as simple and easy to maintain as possible. The repair and restore time are as short as possible so as not to delay the use of users. |
| NFR007 | Localization | The application must have at least two languages, English and French, for Canadian localization. Additional languages can be added to make it accessible to global audiences. |

### 3.2 Functional Requirements

#### 3.2.1 User Account Subsystem

|  |  |  |  |
| --- | --- | --- | --- |
| FR# | Name (Goal Use case) | Role Player | Description |
| FR01 | Sign up the system | All users | For user registration, user needs to input user name, email address, phone number, organization/individual, and user ID. Before that, user needs to be verified not robot by reCAPTCHA system. |
| FR02 | Login the system | All users | Allow registered users to access the system by entering user ID, password, and one-time password. |
| FR03 | Logout the system | All users | User logs out the system properly. |
| FR04 | Change password | All users | Allow user to update password after login successfully. |
| FR05 | Update user profile | All users | Allow user to update profile after login successfully. |

#### 3.2.2 Inventory subsystem

|  |  |  |  |
| --- | --- | --- | --- |
| FR# | Name (Goal Use case) | Role Player | Description |
| FR01 | Create vehicle inventory | System, inventory clerk | Create a new vehicle inventory that includes the vehicle identification number, make, model, year, mileage, color, body type, price, availability, and images. |
| FR02 | Delete vehicle inventory | System, inventory clerk | Delete existing vehicle inventory that includes the vehicle identification number, make, model, year, mileage, color, body type, price, availability, and images. |
| FR03 | Update vehicle inventory | System, inventory clerk | Update existing vehicle inventory that includes the vehicle identification number, make, model, year, mileage, color, body type, price, availability, and images. |

#### 3.2.3 Search engine management subsystem

|  |  |  |  |
| --- | --- | --- | --- |
| FR# | Name (Goal Use case) | Role Player | Description |
| FR01 | Search Engine Integration | System, Users | Integrate the search engine management subsystem with a third-party system to retrieve and display accurate stock availability, price, and delivery date information to users. |
| FR02 | Connect to Third-Party System | System | Establish a secure and reliable connection with the third-party system to fetch real-time stock, price, and delivery date data. |
| FR03 | Stock Availability Search | Users | Enable users to search for products based on their stock availability, ensuring that the search results reflect the current inventory status. |
| FR04 | Price Information Retrieval | Users | Retrieve accurate pricing information from the third-party system and display it to users during their search or product browsing. |
| FR05 | Delivery Date Display | Users | Fetch delivery date details from the third-party system and present the estimated delivery timeframe to users, allowing them to make informed purchase decisions. |
| FR06 | Error Handling and Notifications | System | Implement error handling mechanisms to handle connectivity issues or data retrieval failures from the third-party system. Provide appropriate notifications to users if there are any errors or delays in retrieving stock, price, or delivery information. |

#### 3.2.4 Communication subsystem (Live Chat, Mail Communication such as sending notification, order confirmations, or responses to user inquiries)

|  |  |  |  |
| --- | --- | --- | --- |
| FR# | Name (Goal Use case) | Role Player | Description |
| FR01 | Live Chat | All users, Support Staff | Enable real-time chat communication between users and support staff to address issues, provide assistance, and offer immediate support. |
| FR02 | Mail Communication - Sending Notifications | System, All users | The system sends notifications to users via email, such as important updates, account changes, or service reminders. |
| FR03 | Mail Communication - Order Confirmations | System, All users | The system sends order confirmation emails to users to confirm order details, payment information, and estimated delivery dates. |
| FR04 | Mail Communication - Response to User Inquiries | Support Staff, All users | Support staff respond to user inquiries, questions, or requests via email, providing detailed answers, suggestions, or assistance. |

#### 3.2.5 Reporting subsystem (Reporting Subsystem such as pre-defined report templates, or custom report generation capabilities)

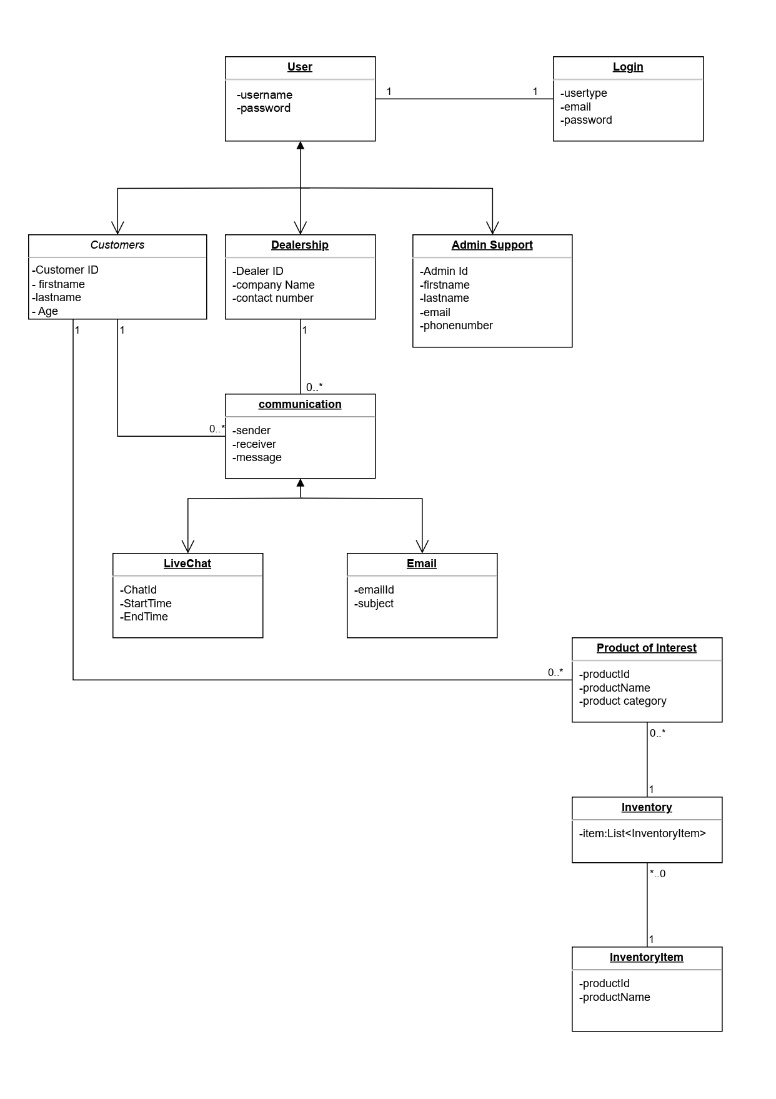
|  |  |  |  |
| --- | --- | --- | --- |
| FR# | Name (Goal Use case) | Role Player | Description |
| FR01 | Generation of Pre-defined Report Templates | System Administrator | Allow the system administrator to create and manage pre-defined report templates, including defining the structure, layout, and data sources of the reports. |
| FR02 | Custom Report Generation | Users | Enable users to generate custom reports by selecting specific report parameters, filtering conditions, and data sources to meet their specific needs. |
| FR03 | Report Export | Users | Allow users to export the generated reports in common file formats such as PDF, Excel, or CSV for further analysis, sharing, or printing. |
| FR04 | Report Scheduling and Automation | System Administrator | Provide report scheduling and automation capabilities to generate and distribute reports on a regular basis, such as generating specific reports daily, weekly, or monthly and sending them to designated recipients. |
| FR05 | Report Access Control | System Administrator, Users | Implement report access control to ensure that only authorized users can access and view specific reports. |

## Section 4: Domain Class Diagram

### 4.1 List of classes

1. Communication
   * 1. Email communication
     2. Live chat
2. Search Engine Management
   * 1. ProductOfInterest
3. Inventory
   * 1. Promotion
4. InventoryItems
5. User Account
   * 1. Customer
     2. Dealership
     3. Admin
6. LoginAccount
7. Report

### 4.2 Diagram



**Picture 2 –** *Domain Class Diagram*

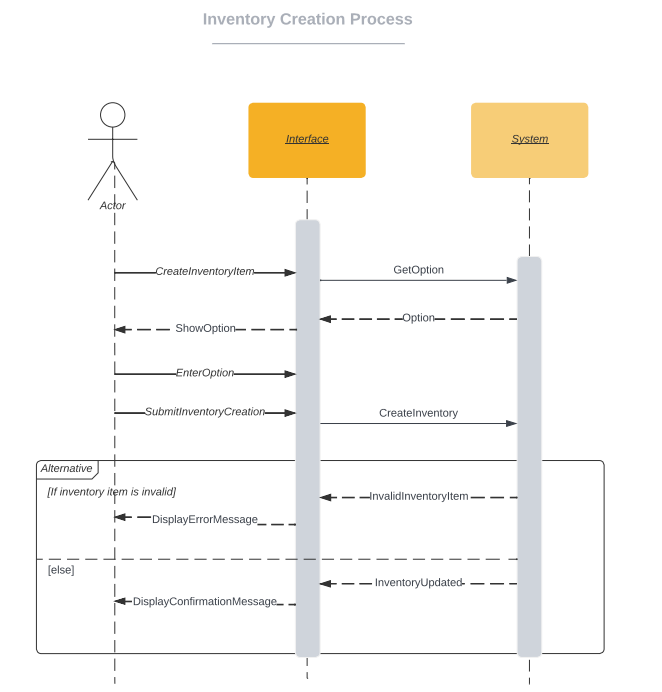
## Section 5: ERD



**Picture 3 –** *Vehicle Finder System ERD*

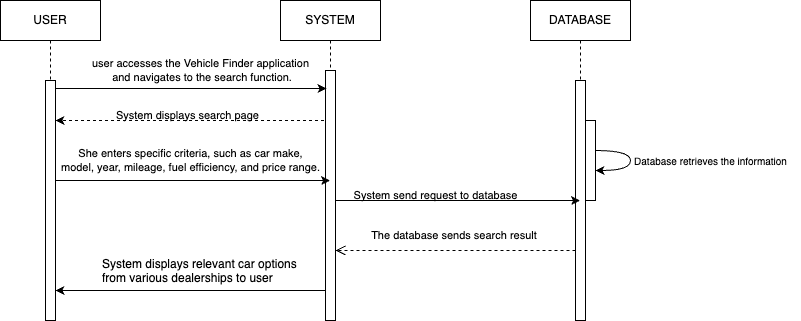
## Section 6: Systems Sequence Diagrams

### 6.1 Inventory Subsystem (use case: create vehicle inventory)



**Picture 4 –** *Create Vehicle Inventory Sequence Diagram*

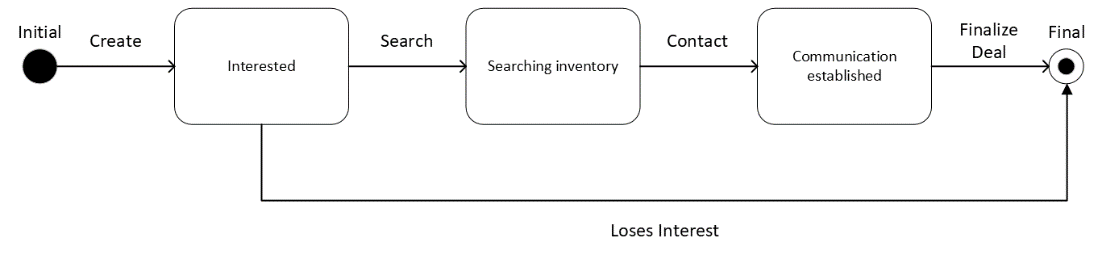
### 6.2 Search (Use case: enter information and retrieve information)



**Picture 5 –** *Enter and Retrieve Information Sequence Diagram*

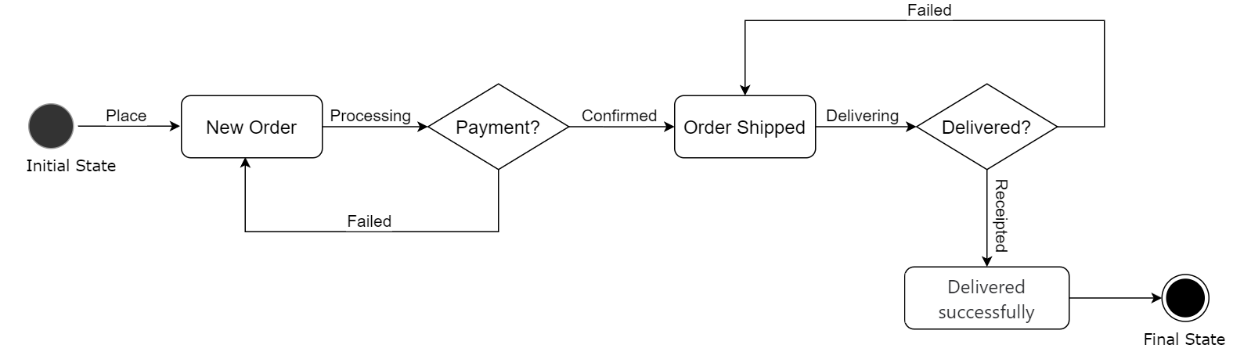
## Section 7: Object State Machine Diagrams

### 7.1 Potential Product Object State Machine Diagram



**Picture 6 –** Object State Machine Diagram of the Potential Product class

### 7.2 Communication Object State Machine Diagram



**Picture 7 –** CommunicationObject State Machine Diagram

## Section 8: Technologies

## 8.1 Web Application: HTML, CSS, JavaScript, C#, ASP.NET MVC Core

## 8.2 Database: Oracle SQL

## Section 9: Project Management (Gantt Chart)

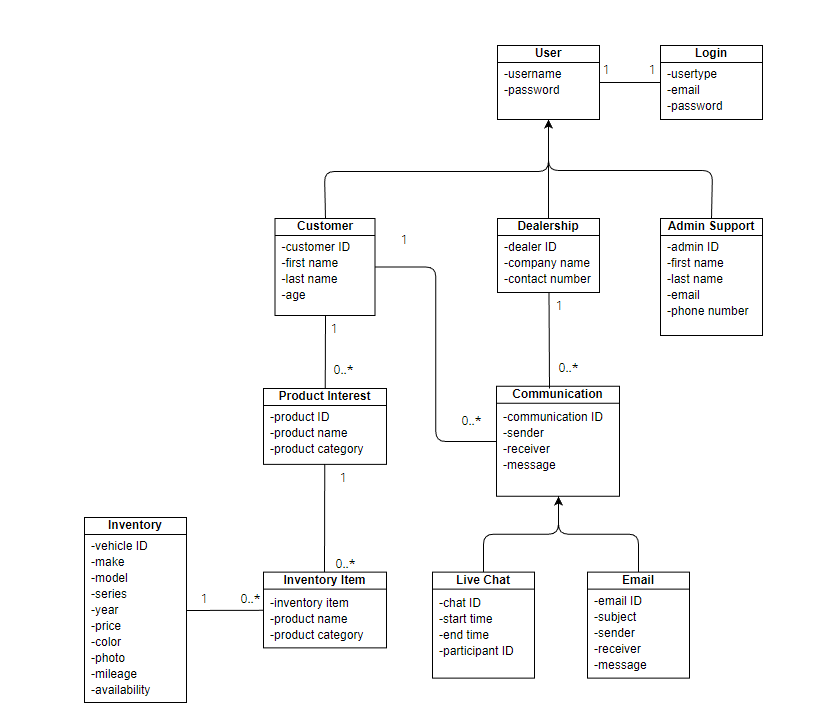
## 

**Picture 8 –** *Gantt Chart*

# Part B: Software Design Architecture

## Section 1: Requirements Edits to Part A

### 1.1 Corrections on Domain Class Diagram (section 4.2)



*Correction of Domain Class Diagram (Section 4.2 from Part A).*

### 1.2 ERD (section 5)

## Section 2: Overview Model

### 2.1 Intended users of the SDD document

The intended users of the SDD (System Design Document) are the project manager, project team and future development team. The document serves as a guideline and orientation for the builders of this application, including management, technology and internal consulting as well.

### 2.2.1 SRS Context Flow Diagram --CFD (What diagram)

A diagram of a vehicle finder system

Description automatically generated

#### 2.2.2 Architectural Context Diagram—ACD (HOW diagram)

A diagram of a vehicle finder system

Description automatically generated

## Section 3: Modularization

### 3.1 Partition the analysis model

#### User Account Subsystem

A diagram of a customer service

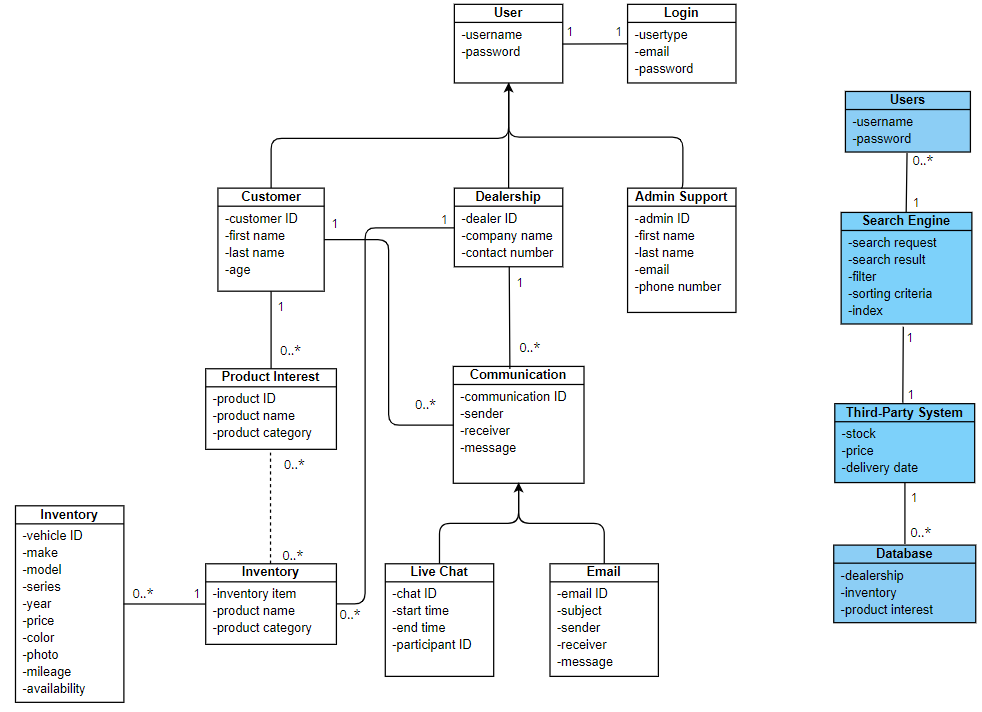
Description automatically generated

#### Inventory subsystem

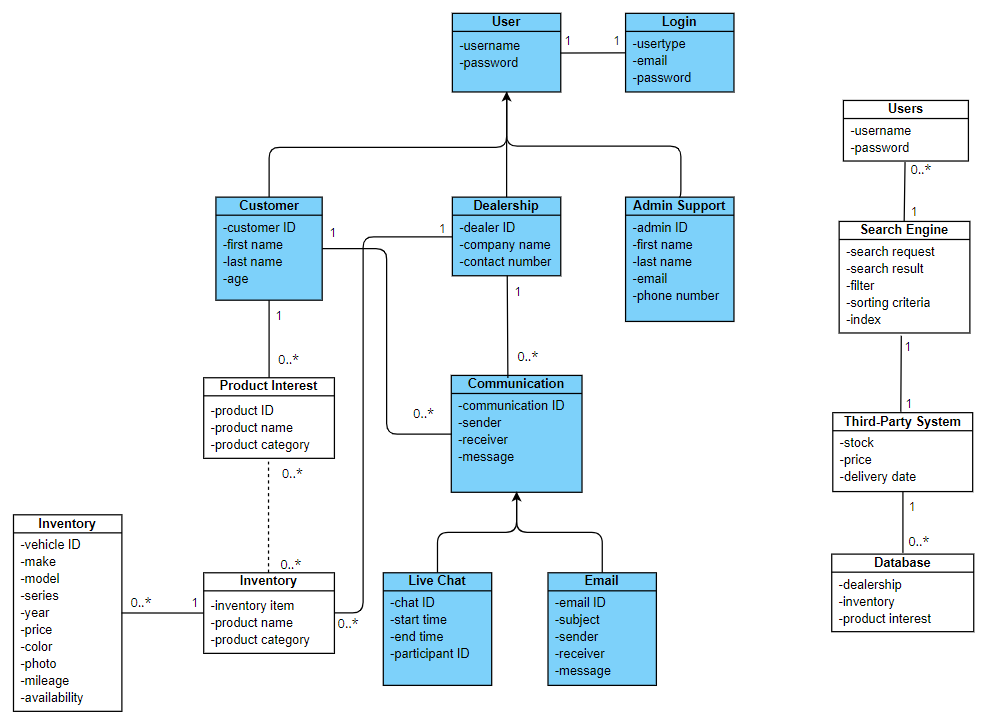
A diagram of a company

Description automatically generated

#### Search engine management subsystem



#### Communication subsystem



### 3.2 Class Responsibility Collaboration (CRC)

#### 3.2.1 User Account subsystem

A picture containing text, screenshot, parallel, number

Description automatically generated

*CRC Cards – User Account subsystem.*

#### 3.2.2 Inventory subsystem

A screenshot of a computer

Description automatically generated

*CRC Cards – Inventory subsystem*

#### 3.3.3 Search Engine management subsystem

A screenshot of a computer

Description automatically generated

*CRC Cards – Search Engine subsystem*

#### 3.2.4 Communication subsystem (corrected)

A screenshot of a computer

Description automatically generated

*CRC Cards – Communication subsystem*

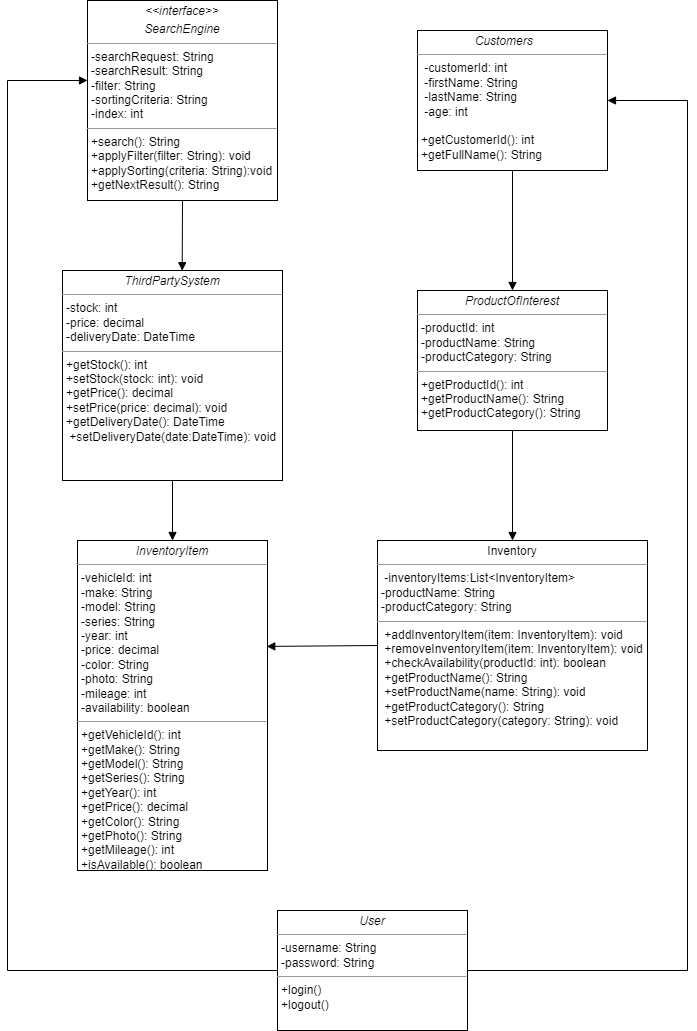
3.3 Design classes diagram

3.3.1 Login subsystem diagram:

A diagram of a computer

Description automatically generated

3.3.2: Search subsystem diagram:



3.3.3:Inventory subsystem diagram:

A diagram of a company

Description automatically generated

3.3.3:communication subsystem diagram:

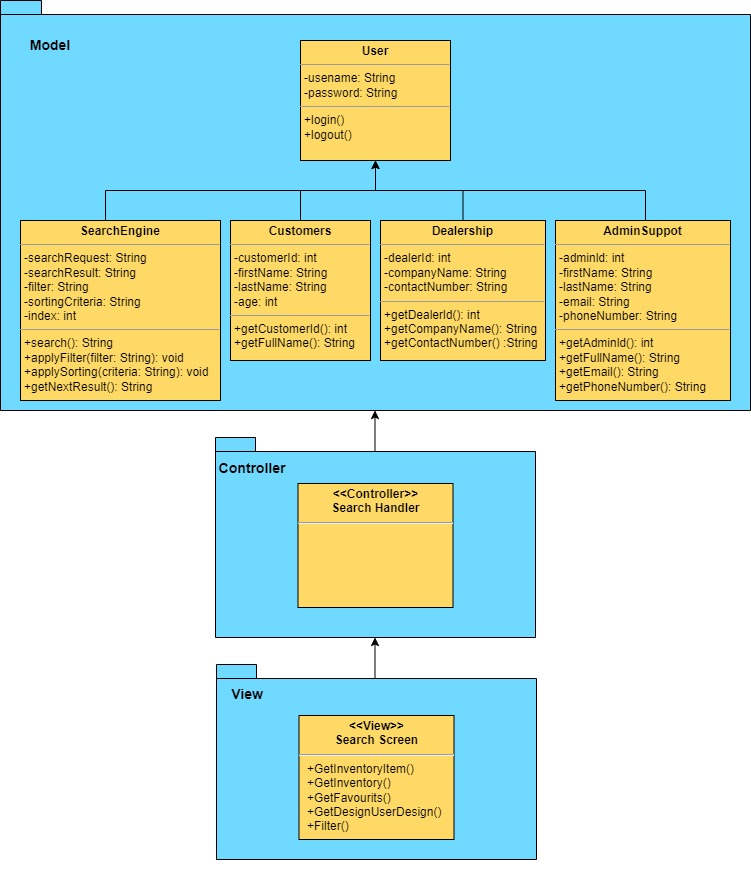
A diagram of a computer program

Description automatically generated

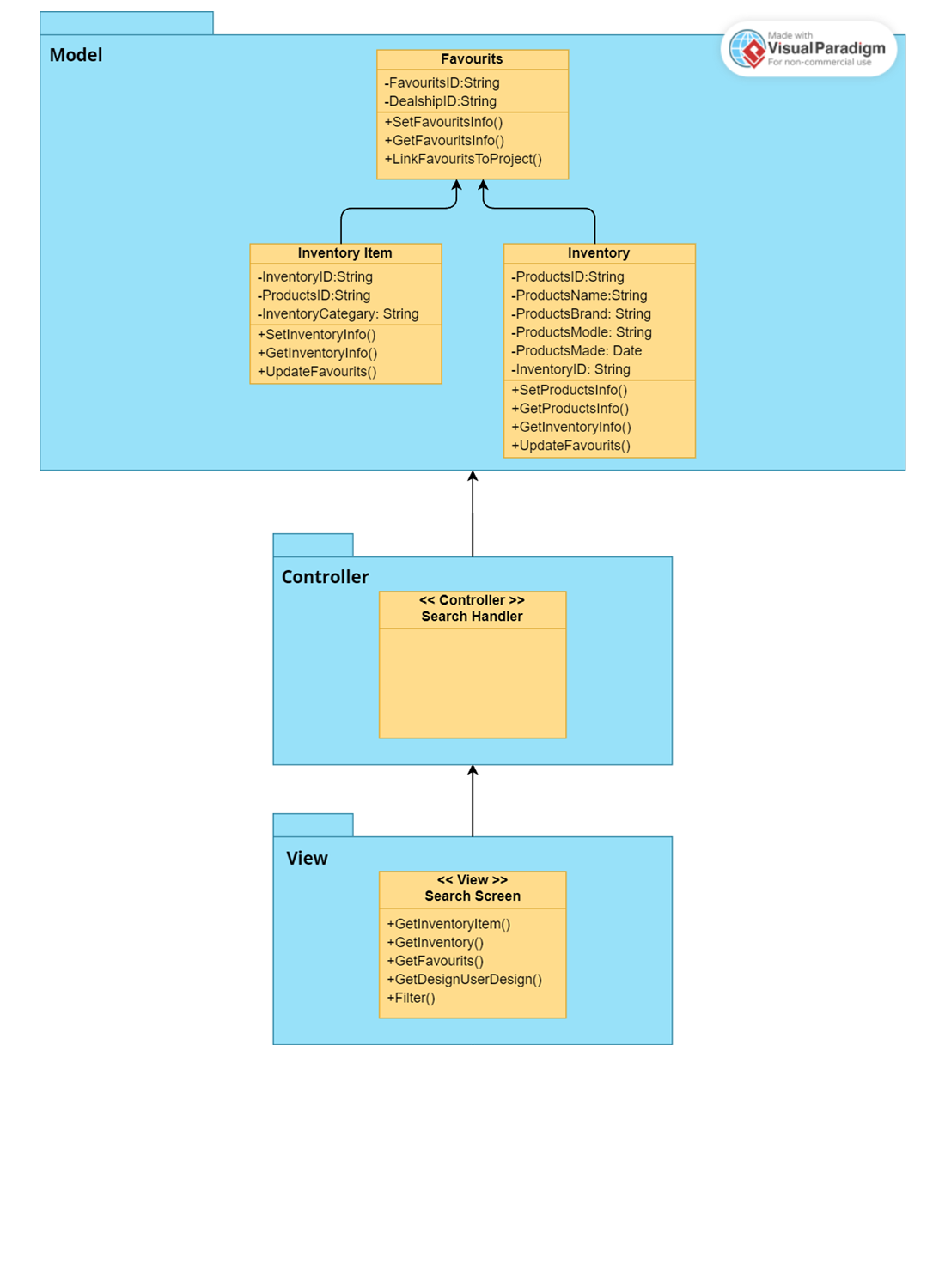
## Section 4: Framework M(odel) V(iew) C(ontroller)

### 4.1 MVC pattern diagram to include

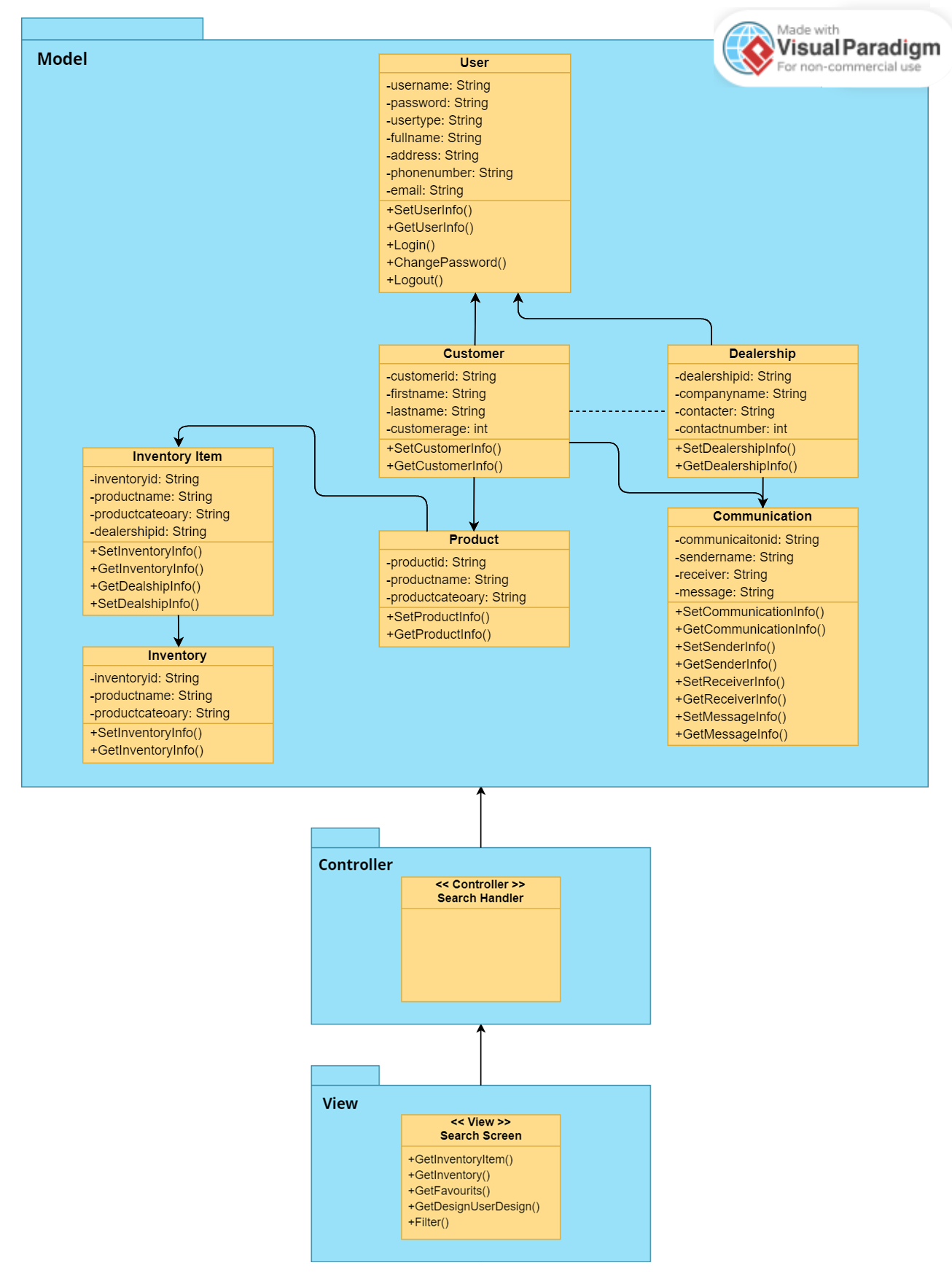
#### 4.1.1 Login Subsystem



#### 4.1.2 Search Subsystem



#### 4.1.3 Products Design Subsystem



### Full Sequence diagrams

### 4.2.1 Inventory Subsystem (use case: create vehicle inventory)

A diagram of a diagram

Description automatically generated

#### 4.2.2. Use cases: Enter information and Retrieve information (Search Subsystem)

A diagram of a graph

Description automatically generated

### State Machine Diagrams

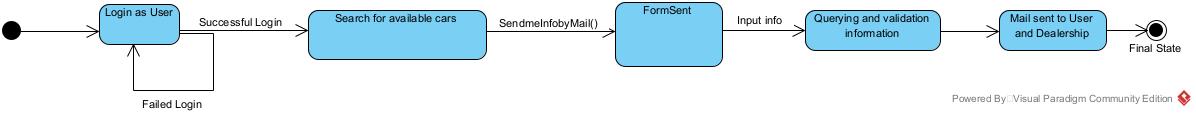
#### 4.3.1 Objects: use case: Create vehicle inventory

A diagram of a diagram

Description automatically generated

*State machine diagram of “Create new item in inventory.*

#### 4.3.2 Objects: use case: Send Mail Communication – Notifications



*State machine diagram of “Send notifications to the users”.*

## Section 5: Data Layer

### 5.1. Database schema.

A diagram of a company

Description automatically generated

*Database Schema for the System, all its subclasses and properties.*

### 5.2 Technology List Update

No updates required.

## Section 6: Gantt chart update

A screenshot of a computer

Description automatically generated

# Part C: System Design Documents

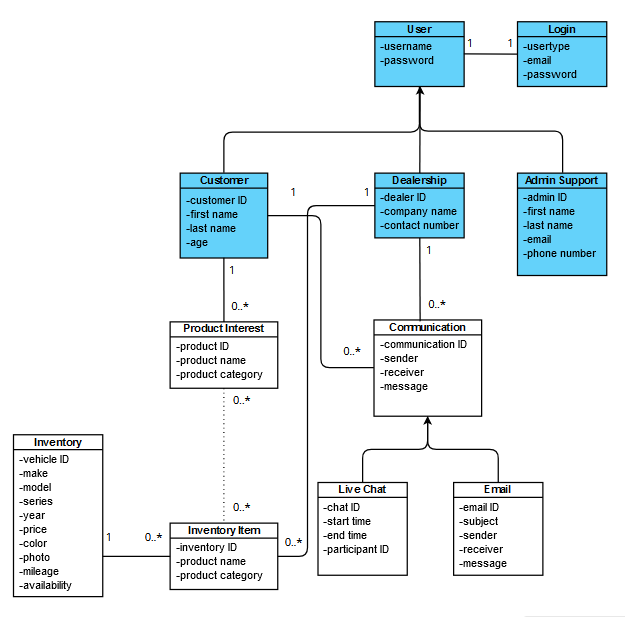
## Section 1: Corrections to Design Specifications Part B

## 

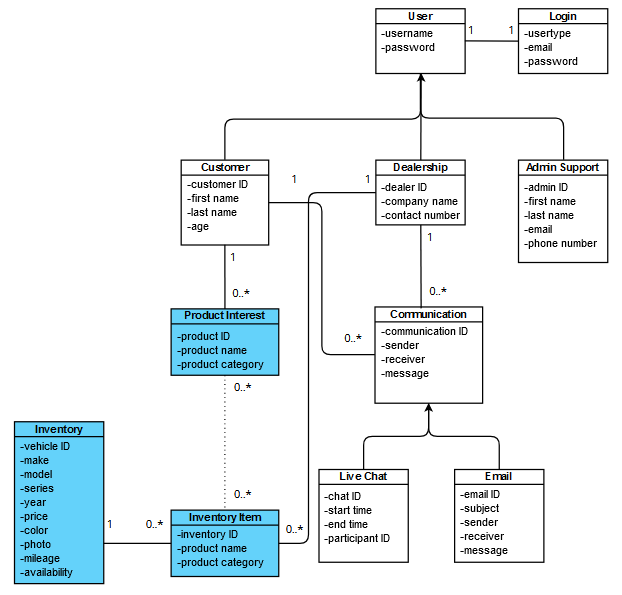
Section 3: Modularization

3.1 Partition the analysis model

3.1.1 User Account Subsystem



3.1.1 Inventory subsystem



## Section 2: Software Design Patterns

|  |  |
| --- | --- |
| **Name:** | Login |
| **Problem:** | Login Subsystem is tightly connected with the User Subsystem, and users need to interact with both subsystems when logging in and managing their user-related activities. |
| **Solution:** | Facade pattern creates a single-entry point that encapsulates the interactions between the Login Subsystem and the User Subsystem. This entry point abstracts the underlying complexity and provides a simplified interface for clients. |
| **Graph:** |  |
| **Benefits and Consequences:** | Users interact with a single, well-defined interface, abstracting the complexities of both the Login and User subsystems. Clients don't need to know the details of the internal interactions between subsystems, leading to lower coupling and improved maintainability. The Facade can shield clients from changes in the subsystems' interfaces or implementations, allowing for easier upgrades. The Facade introduces an additional layer, which could be seen as unnecessary if the interactions between the subsystems are simple. If the interactions between the subsystems are straightforward, the Facade might add a slight overhead. |

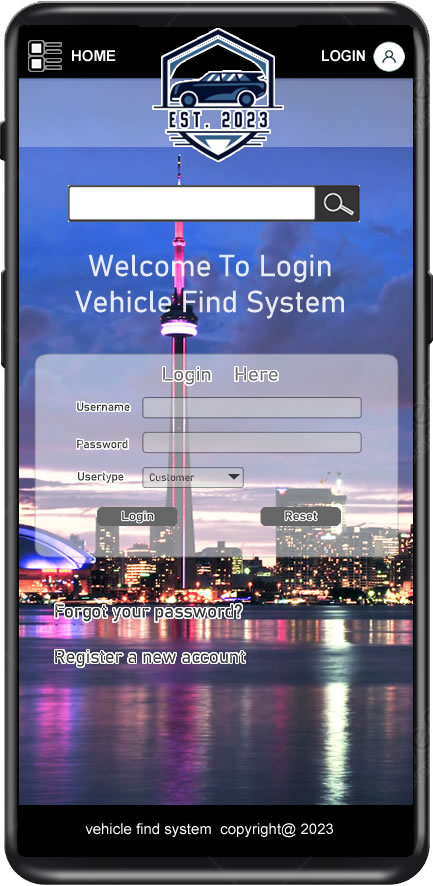
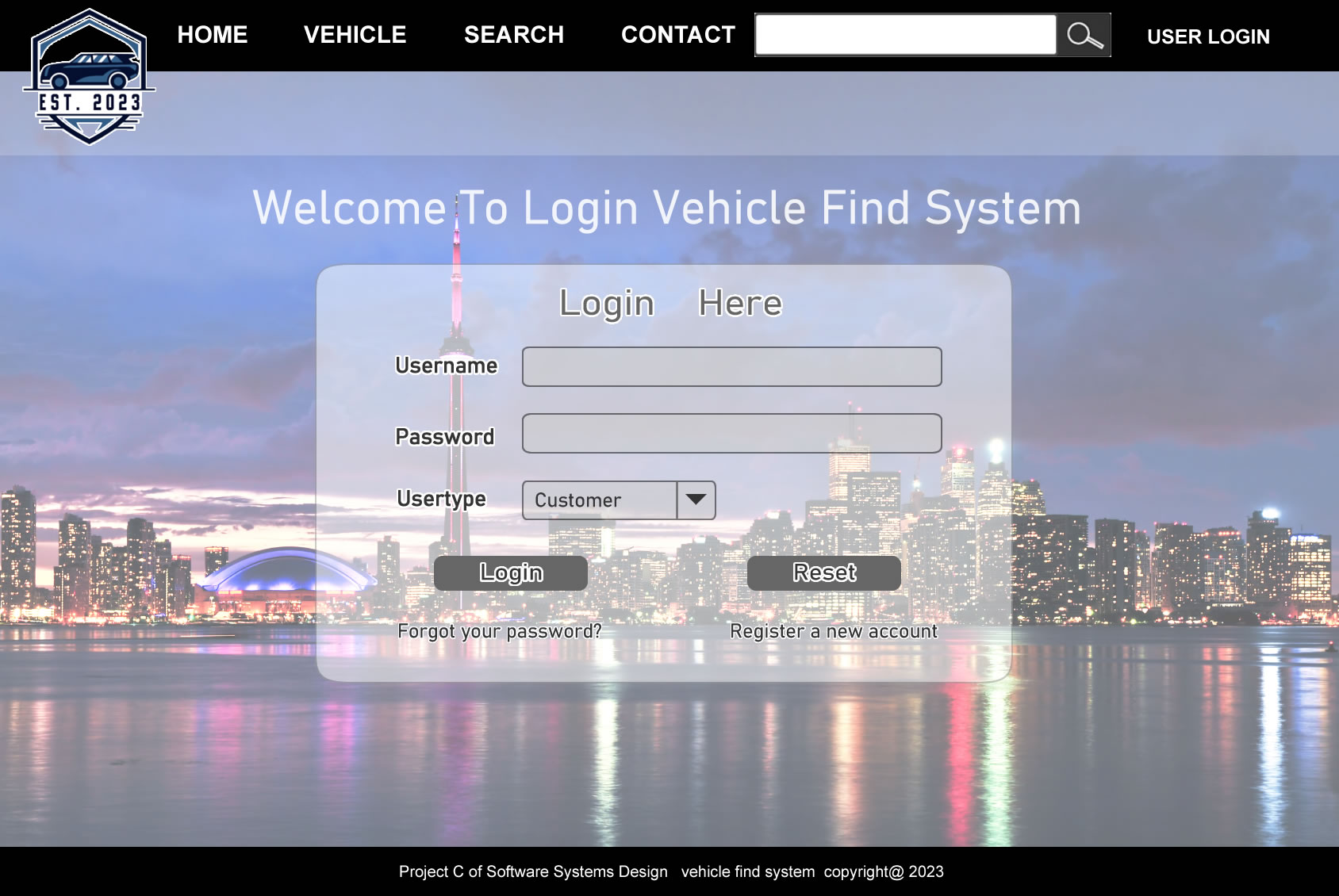
|  |  |
| --- | --- |
| **Name:** | Communication |
| **Problem:** | Communication Subsystem needs to support multiple communication methods (email, live chat) between different types of users. |
| **Solution:** | The Strategy pattern defines a common interface for communication strategies and implements separate classes for each communication method. The user of the Communication Subsystem can then select the appropriate strategy based on the context. |
| **Graph:** |  |
| **Benefits and Consequences:** | You can add new communication methods without changing the existing code, promoting flexibility and extensibility. Each communication strategy class has a single responsibility, making the codebase more maintainable and organized. Introducing multiple strategy classes could increase the complexity of the codebase, so proper organization and naming conventions are crucial. The Strategy pattern introduces an additional level of indirection, which might lead to performance overhead if not managed properly. |

|  |  |
| --- | --- |
| **Name:** | Search |
| **Problem:** | Users need to search through dealerships' inventories and select vehicles of interest, which might be individual items or collections of vehicles. |
| **Solution:** | Apply the Composite pattern to create a common interface for both individual items (vehicles) and collections of items (inventories). Dealerships' inventories can be treated as composite objects that contain individual vehicles as leaf nodes or other inventories as child composites. |
| **Graph:** |  |
| **Benefits and Consequences:** | The Composite pattern provides a consistent way to treat individual items and collections of items, simplifying client code. Allows you to build complex hierarchies of objects while treating individual and composite objects uniformly. Introducing the Composite pattern might increase the complexity of the codebase, so proper design and clear interfaces are important. If the hierarchy is deep or contains a large number of nodes, there could be performance overhead when traversing the structure. |

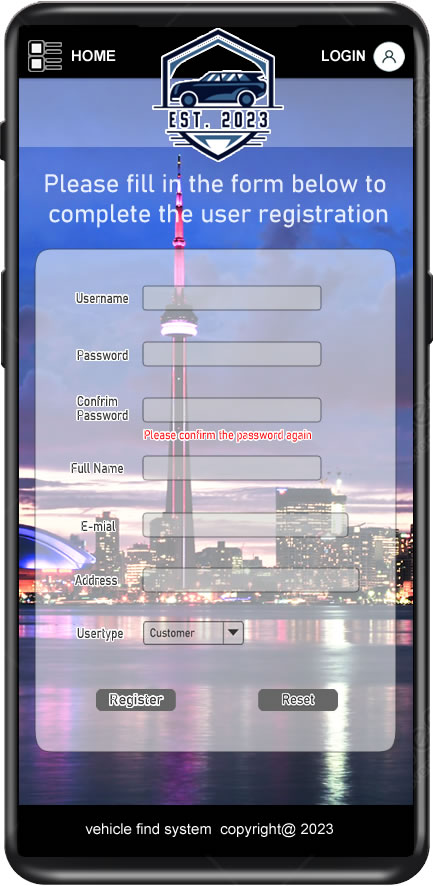
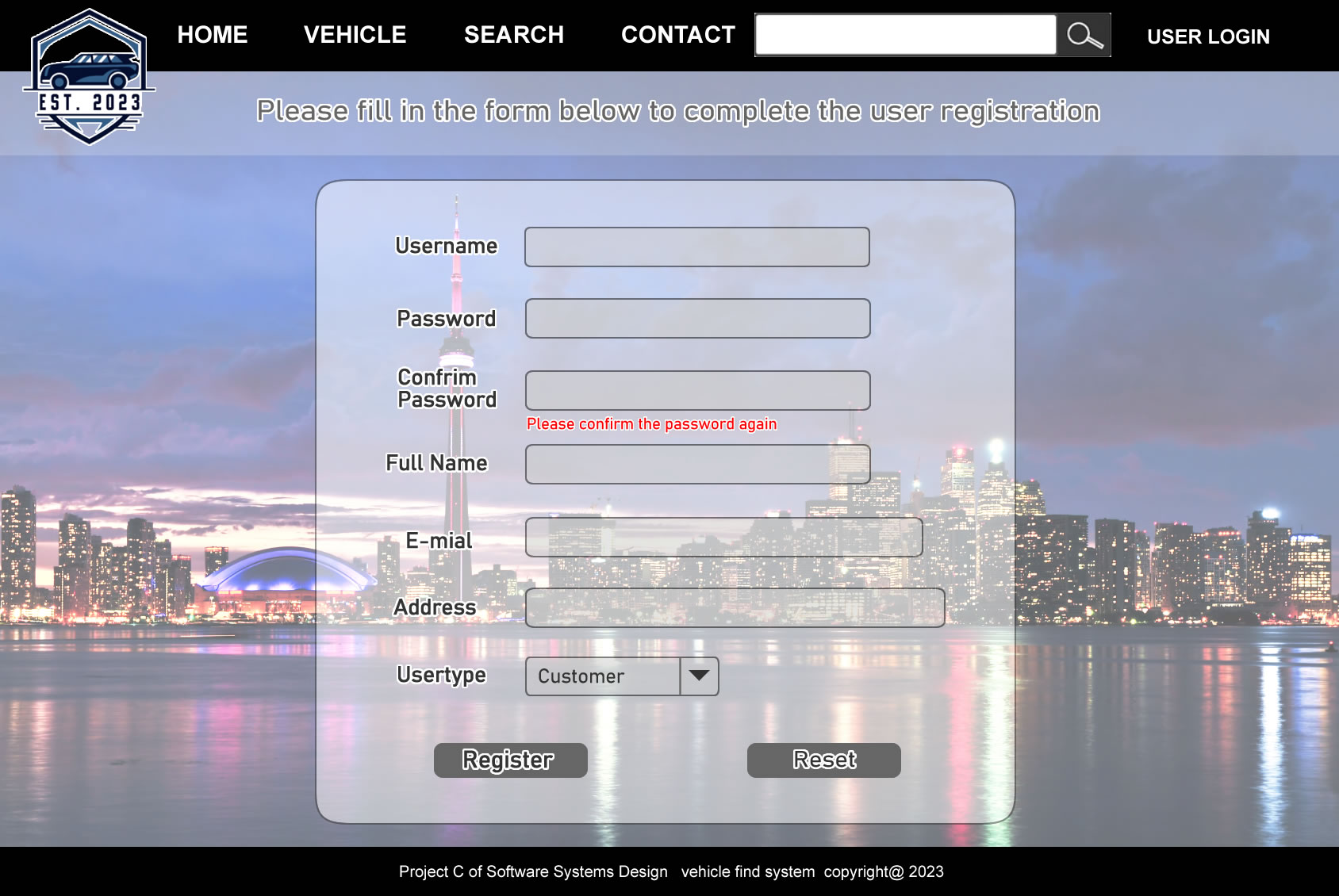
|  |  |
| --- | --- |
| Name: | **Inventory Adapter** |
| Problem: | The vehicle finder application requires consistent reconciliation with an outside stock administration framework to monitor vehicle accessibility, oversee stock, and handle updates. Nonetheless, the current vehicle finding framework has an alternate connection point and information structure from the external inventory management system, making direct integration challenging and disruptive. |
| Solution: | Implement the "Inventory Adapter" design pattern to bridge the gap between the vehicle finder application and the external inventory management system. Create an adapter class that translates calls from the vehicle finder system's interface into calls that the external inventory management system understands. |
| Graph: |  |
| Benefits and Consequences: | It enhances vehicle finder application by seamlessly integrating an external inventory management system. It maintains modularity, allowing updates or replacements of the external system without core changes. However, the pattern introduces an extra layer of code, potentially increasing complexity and requiring careful data mapping within the adapter class for effective communication between systems. |

## **Section 3: UI/UX design**

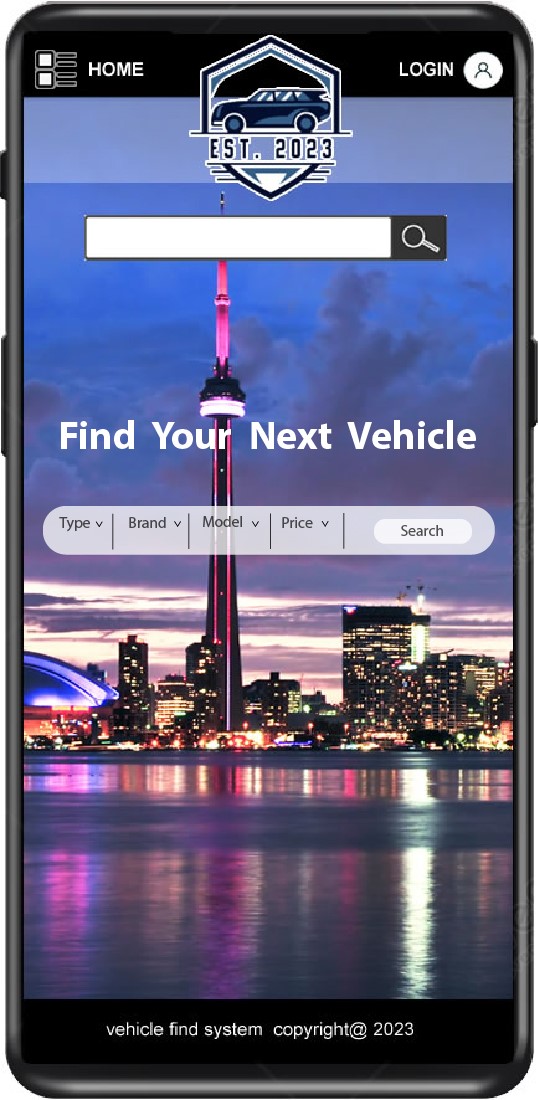
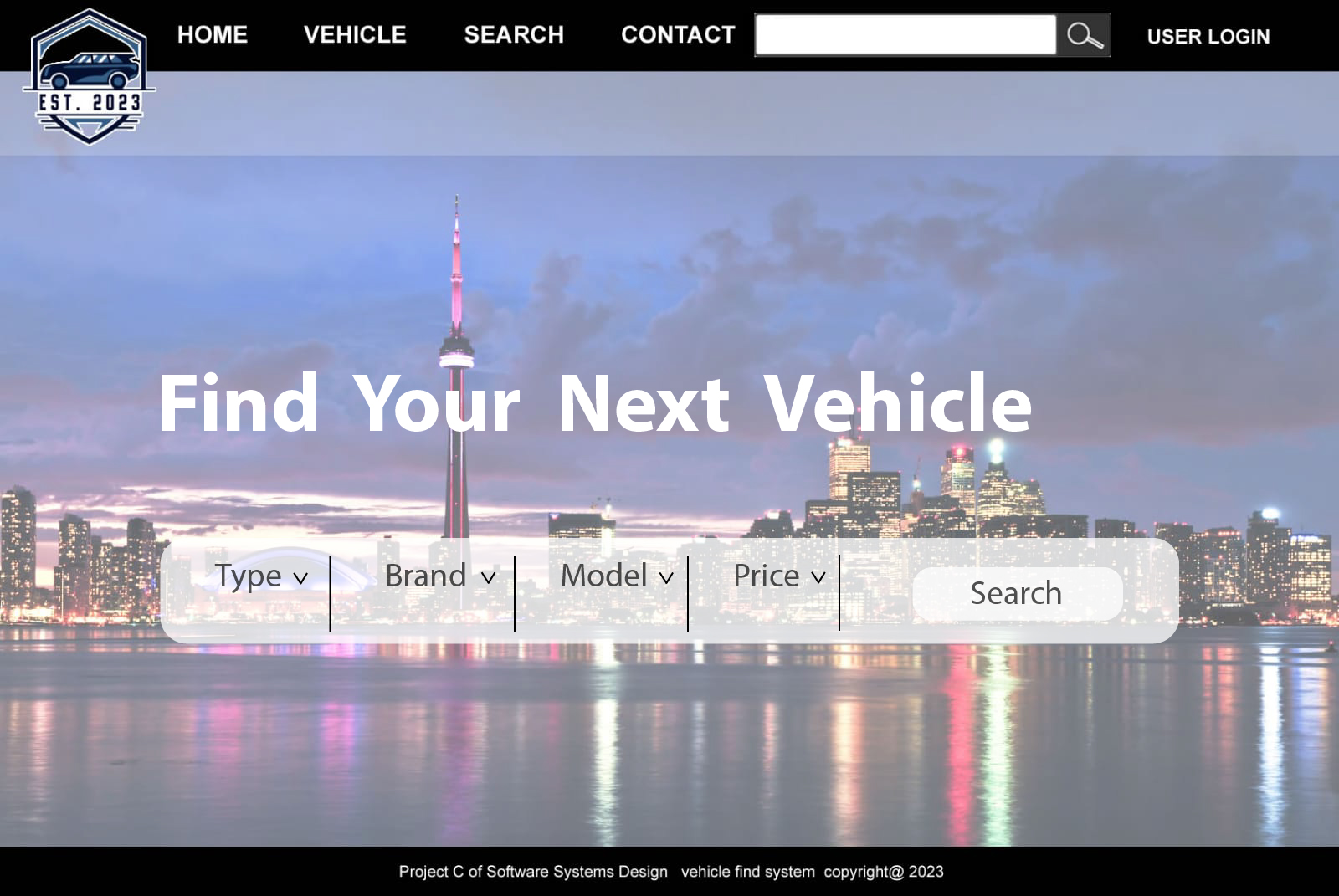
### **3.1 Sign In (login)**



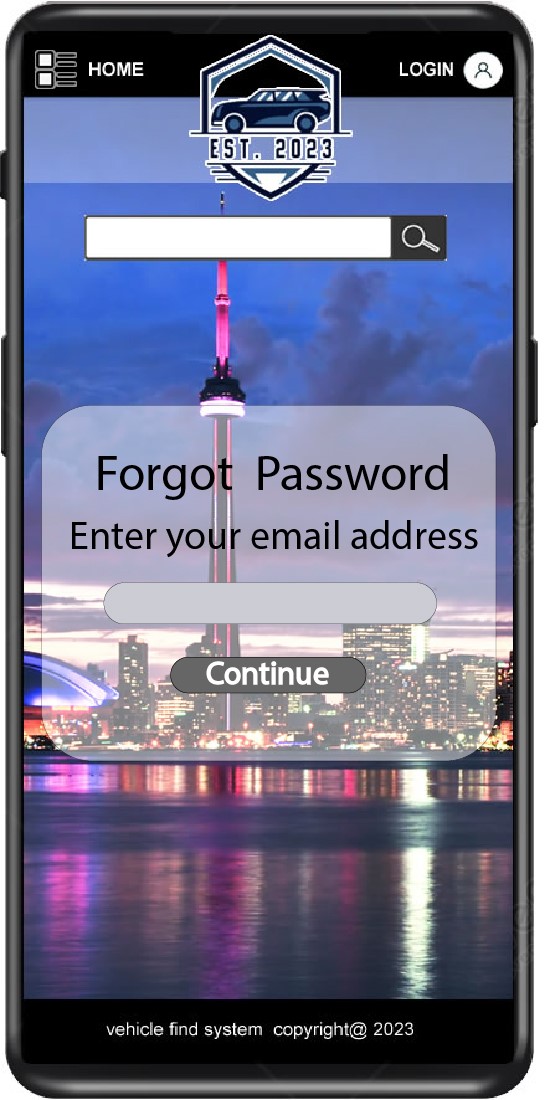
**3.2 Sign Up**



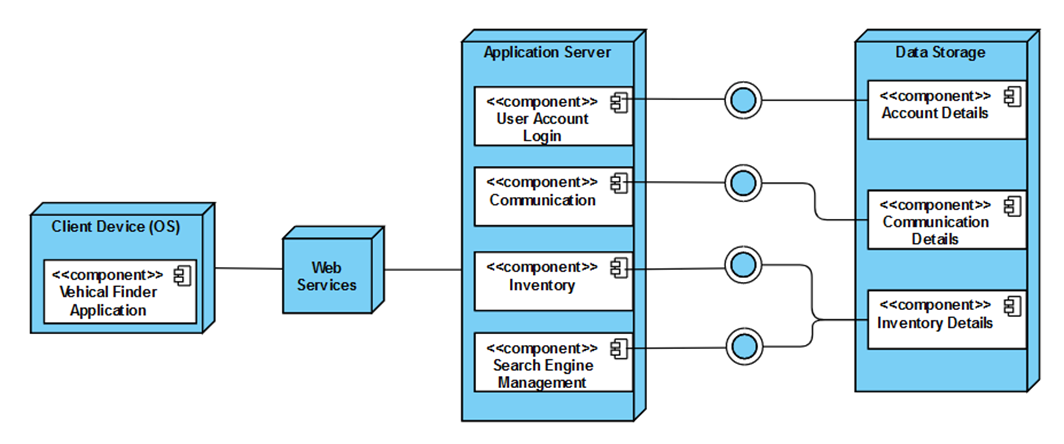
**3.3 Home**



**3.4 Change Password**



## Section 4: High Level Component / Deployment Diagram



## Section 5: Updated Gantt Chart for part C

