**MileStone1**

**Group 3**

|  |  |  |
| --- | --- | --- |
| **SID** | **STUDENT NAME** | **Effort given** |
| 202110519 | Maryam Bawazir | 25% |
| 202205676 | Hala Hamdoun | 25% |
| 202003441 | Alshaykhah Alshammari | 25% |
| 202109396 | Rogayah omar | 25% |

**CMPS310– B51**

**19 Oct. 24**

**L51**

1. **Use Case Specifications**

|  |  |  |
| --- | --- | --- |
| **Use case Id:** UC01 | Transfer unregistered vehicle ownership | |
| **Brief Description** | The use case starts when the vehicle owner enters the VIN and vehicle details to start the transferring process. The system communicates with the vehicle manufacturer and Qatar Trade Service to check vehicle data. The ownership transfer is completed by creating new registration sticker and invoice. | |
| **Primary actors** | Owner, Manufacturer, Qatar Trade Service | |
| **Trigger** | An owner enters VIN and vehicle details to transfer ownership. | |
| **Preconditions:**   1. **Vehicle is unregistered** 2. **Vehicle has a valid VIN** | | |
| **Post-conditions:**   1. Vehicle ownership was transferred to the new owner 2. New registration number was generated 3. New registration sticker and invoice were created | | |
| **Main Success Scenario** | | |
| **Actor Action** | | **System Response** |
| 1. The vehicle owner inputs VIN and vehicle details | | 2. Send VIN and vehicle details to the vehicle manufacturer |
| **3. The vehicle manufacturer verifies the information** | |  |
| **4. The vehicle manufacturer sends confirmation** | | 5. Ask for current owner name and QID |
| **6. The vehicle owner enters the current owner name and QID** | | 7. Check with Qatar Trade Service to see if the vehicle was imported by providing VIN and QID |
| **8. QTS reviews its records and sends back a reply** | | 9. Re-call the vehicle’s insurance policy |
|  | | 10. Create new registration number |
|  | | 11. Allocate the registration number to the vehicle |
|  | | 12. Display number to owner |
| **13.The vehicle submits the new owner details** | | 14. Store new owner information |
|  | | 15.Identify the new owner as current owner and the current owner as previous owner |
|  | | 16.Create new registration sticker |
|  | | 17.Create transfer fee invoice |
| **Alternative flows:**  5a. If the manufacturer does not approve, end the session with an error message 9a. If the response is negative, end the session and display an error message | | |

|  |  |  |
| --- | --- | --- |
| **Use case Id:** UC02 | Renew a vehicle registration | |
| **Brief Description** | This use case begins when a vehicle owner enters VIN. The system verifies the information, issues a registration sticker, and generates an invoice for any fees. | |
| **Primary actors** | Vehicle owner, Insurance company, workshop | |
| **Trigger** | The vehicle owner enters VIN. | |
| **Preconditions:**   * The vehicle owner is logged into the iQVR system. * The vehicle onwer has an expired registertion | | |
| **Post-conditions:**   * The vehicle’s registration was successfully renewed in the iQVR system. * A new invoice was added to the owner’s account | | |
| **Main Success Scenario** | | |
| **Actor Action** | | **System Response** |
| 1. The owner enters the VIN. | | 1. retrieves the vehicle registartion deatails. |
|  | | 1. checks the age of the vehicle |
|  | | 1. retrieves the fitness certificate (see 4.a.) |
|  | | 1. retrieves the current vehicle insurance policy. (see 5.a.) |
|  | | 1. checks for any unpaid fines. (see 6.a.) |
|  | | 1. creates a new registration. |
|  | | 1. attaches the insuarnce policy and fitness cerificate with the vehicle registartion. |
|  | | 1. creates a new valid registration sticker |
|  | | 1. prepare invoice for the renewal. |
| **Alternative flows:**  4.a. if the vehicle does not have a fitness certificate, terminates the session with a message “Get fitness certificate first”  5.a. if the insurance policy is not valid or not obtained yet, terminates the session with the message "Get insurance policy."  6.a If there are unpaid fines, asks the vehicle owner to pay them with the message “Pay the bill first” | | |

|  |  |  |
| --- | --- | --- |
| **Use case Id:** UC03 | Handle accident | |
| **Brief Description** | Use case begins when the owner of the offending vehicle enters the VINs of both vehicles involved. The system retrieves vehicles registration details. The system creates an accident report which. The report is sent to the insurance company and the company sends back a receipt. | |
| **Primary actors** | Owner, Insurance company | |
| **Trigger** | The owner of the offending vehicle enters the VINs of both vehicles involved in the accident | |
| **Preconditions:**   1. **An accident happens between two vehicles** 2. **The owner of the offending vehicle accepts responsibility for the accident** | | |
| **Post-conditions:**   1. **An accident report was created** | | |
| **Main Success Scenario** | | |
| **Actor Action** | | **System Response** |
| 1. The offending vehicle owner enters the VINs of both vehicles involved in the accident | | 2.Retrieve vehicles registration info and other vehicle data |
|  | | 3.Request accident details |
| **4. The offending vehicle owner enters accident details** | | 5.Save accident information |
|  | | 6. Assign offending and victim vehicles |
|  | | 7. Request offending vehicle’s confirmation |
| **8. The offending vehicle owner sends a confirmation** | | 9. Store confirmation |
|  | | 10. Retrieve offending vehicle’s insurance policy |
|  | | 11. Generate an accident report |
|  | | 12. Provide the insurance company with the accident report |
| **13. The insurance company replies with an acknowledgement receipt** | | 14. Save receipt |
|  | | 15. Store accident report |

|  |  |  |
| --- | --- | --- |
| **Use case Id:** UC04 | Pay invoice | |
| **Brief Description** | Use case starts when the vehicle owner enters the VIN credit card details. The owner selects invoice to pay. The system then contacts the qPay to check the credit card’s validity. The invoice status is changes to ‘Paid’. | |
| **Primary actors** | Vehicle Owner, qPay System | |
| **Trigger** | The vehicle owner enters the VIN and credit card details. | |
| **Preconditions:**   * The vehicle owner is logged into the iQVR system. * The vehicle has unpaid invoices or fines. * The vehicle owner has a credit card. | | |
| **Post-conditions:**   * A payment receipt is generated and displayed to the vehicle owner. * The unpaid invoice list is updated. | | |
| **Main Success Scenario** | | |
| **Actor Action** | | **System Response** |
| 1. owner enters the VIN, and credit card details. | | 1. retrieves the vehicle’s registration details. (see 2.a.) |
|  | | 1. displays the list of unpaid invoices. |
| 1. the owner selects the invoices and fines they want to pay. | | 1. saves the onwer selection of the invoices. |
|  | | 1. computes the total amount of the invoice(s). |
|  | | 1. sends the card details and the total amount to the qPay system |
| 1. The qPay contacts the bank to check the card's validity. | |  |
| 1. The qPay returns the approval. | | 1. receives the qPay response, and saves it, with the approval number. |
|  | | 1. generates a payment receipt |
|  | | 1. displays the payment receipt to the owner. |
|  | | 1. Update invoices status associated with the vehicle. |
| **Alternative flows:**  2.a. If there is no regiteration information, displays an error message and prompts for a correct VIN.  10.a. if no approval was received from qPay, promts the vehicle onwer to enter valid credit card detials. | | |

1. **Software Requirement Analysis**

**[DFD]**

**[assumptoins]**

**[UC]**

**[assumptoins]**

**Explain which of these two you prefer best to analyze a system and why (*5%*).**

We prefer the Data flow digram **DFD** because it is more detailed. It allows us to see how the system interacts with every entity (actor) in the iQVR system. Additionally, it shows how data moves from the source (actor), is transformed by each process, and is stored. With the DFD, we can also see how the system manages the database.

As for the **Use Case** diagram, while it summarizes the use cases for each actor, we find it to be too general. It does not show the specific movement of data or how it is transformed within the system or saved in it, unlike the DFD.

[CD]

[**assumptoins]**

**Explain briefly how you applied design principles in your class diagram. In your explanation, give examples from your class diagram (*5%*).**

In our class diagram we have:

* **Association lines:** We created association lines between classes like the InsurancePolicy and Owner to indicate relationships between these entities.
* **Inheritance:** The classes UnregisteredVehicle and RegisteredVehicle inherit from the Vehicle class, allowing them to share common properties and behaviors such as make, model, and registrationNumber, while also allowing specific attributes or methods for each subclass.
* **Abstract Class:** The Vehicle class is abstract, meaning it is not meant to have objects instantiated directly from it. Instead, it provides a template for its subclasses (RegisteredVehicle, UnregisteredVehicle), which must implement or extend the abstract methods.
* **Aggregation:** We used aggregation to represent a "whole-part" relationship, such as between the CreditCard and Owner. The CreditCard is part of the Owner, but it can exist independently if necessary (i.e., the card doesn't necessarily get destroyed if the owner object is deleted).
* **Responsibility Principle:** Each class in our diagram has a clear responsibility. We considered both the properties (attributes) and behaviors (methods) of each class.
* For example, the Vehicle class has the method updateOwner(cOwner, pOwner) which interacts with the Owner objects to fulfill the responsibility of updating the vehicle's ownership information.
* **Multiplicity:** We added multiplicities to define how many instances of one class can be associated with instances of another class. For example, the relationship between Owner and AccidentReport shows that "an owner can retrieve 0 to many accident reports, and an accident report can be retrieved by 0 to a maximum of 2 owners."
* **Encapsulation:** all of our attributes in our classes are private, while the methods are public.

**Assumptions**