Software Design

Company: Top Deal Auto Melbourne, Australia

Software: Car Selling Website

Team name: Prestige K/DA

Team Members:

Name	ID	Roles
Pham Duc Linh	103792371	Product Owner - BA
Pham Anh Vu	103806447	Solutions, IT Architect
Nguyen Thanh Dat	103804881	Project Manager - Scrum Master
Tran Tuan Nam	103792643	Lead Developer (BE)
Phung Xuan Tung	103792054	Lead Developer (FE)

Tutorial class: Fri 1:00 PM DT7.2

Tutor's Name: Dr. Pham Thi Kim Dzung

Class Diagram

This is the team's final Design for the Class diagram for the project, which consists of the classes, objects and their relationship with each other

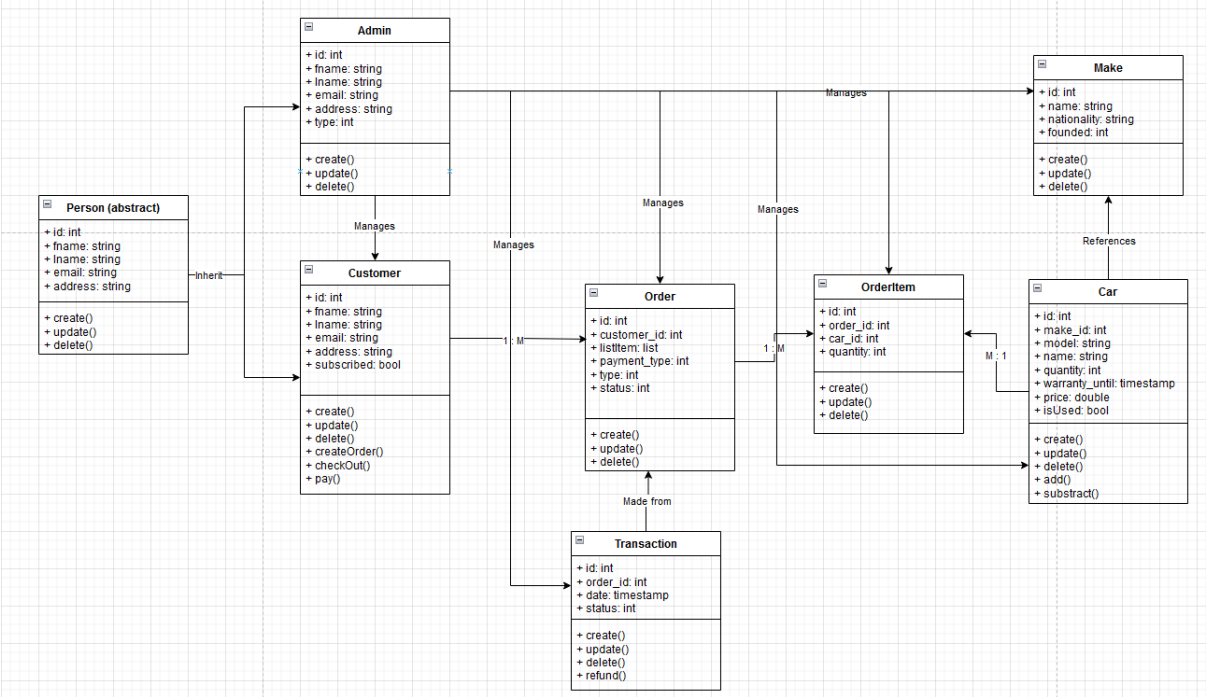


Figure 1. Class Diagram

Person Class: This class is an abstract class that symbolizes the users that will be using this application. The different types of users will be discussed later. All users share these same attributes: unique identifier, name, email, address. They also share simple create, delete and update operations which can be performed on them by the code.

There are 2 types of Users, or in Software Engineering terms, 2 classes that inherit from the base Person class:

- **Customer:** This class is for the Top Deal Auto's customer. In addition to the base class' attributes, they also have an isSubscribed attribute for the newsletter, as well as the ability to perform purchasing actions on the products
- **Admin:** This class represents the Top Deal Auto's admin. This user has extra privileges that they can perform administrative tasks like managing users, products, orders, transactions,…

Car Class: This class represents the product that is being sold by Top Deal Auto - the cars. The cars have their unique identifier, their make, models, price, quantity, warranty_until and a isUsed flag for usage details. The make actually calls to another class called **Make** through their unique identifier**,** which represents basically car brands

All of these classes have basic Create, Update and Delete functionalities.

When a **Customer** orders a **Car,** an **Order** is created. An order has its own unique identifier, as well as references to the **Customer** - i.e. the owner of the **Order.** An **Order** also has extra information like price, payment type, order types, and its status, which falls in line with Top Deal Auto's business procedures.

An **Order** can have multiple **Car**s, and vice versa, a **Car** can exist in multiple different **Order**s. So to interconnect these classes, we need a conjunction class. And that's where **OrderItem** comes in, which saves the order-car relationship record, as well as its quantity, to differentiate a **Car** in a certain **Order** to the same **Car** but in another **Order.**

When an **Order** is completed, a **Transaction** record is created, this is per Top Deal Auto's compliance business. A **Transaction** record saves a purchase of an order, with its unique identifier, the **Order**'s unique identifier, the date it was recorded and its status.

User Sequence Diagram

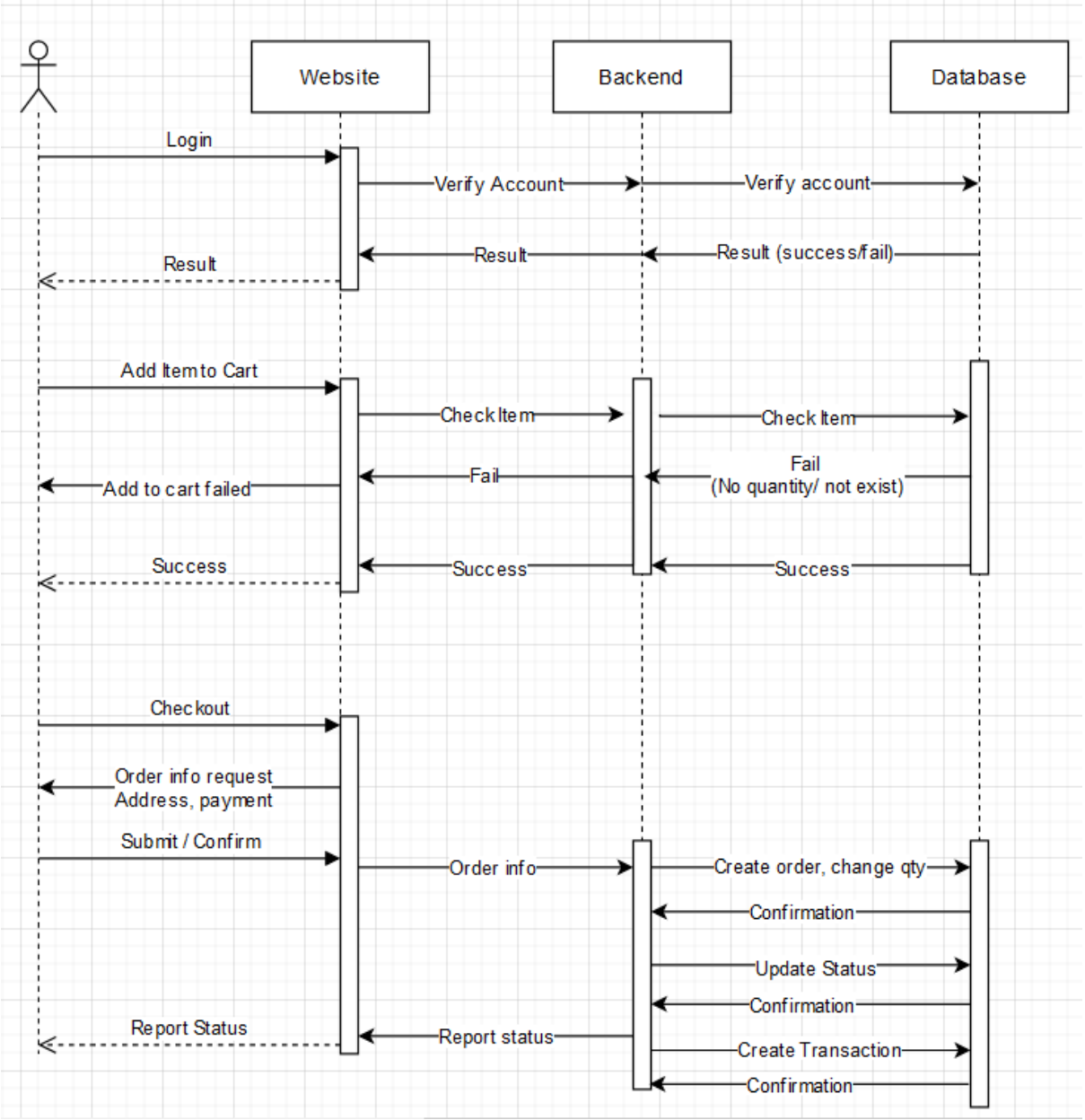


Figure 2. Customer Sequence Diagram

Next, is the Sequence Diagram but from the Customer side. This represents the workflow, and how the 3 main components of our web app communicate and interact with each other during a Customer action

There are 3 main Customer flows we want to describe

Login

The User will login through our Website's frontend. The login information is sent to the backend server. The backend Server then checks the login data sent by the user with the Database to verify it. The result of that verification process will be sent back to the backend, and then forwarded back to the user through our Website's front end.

Add Car to Order

This workflow requires the Customer to be logged in to our website.

When an user adds an Item to Order, the frontend will send that request to the Backend. The backend first checks if the Item is valid (in this case check for the quantity or the item's existence). If the item is invalid for whatever reason, the Backend will immediately send back to the frontend an error message indicating that the Car cannot be added to Order, failing the process.

If the Car does exist, it will be added to the User's Order catalog, and a message notifying the process being successful will be sent to the user through the frontend.

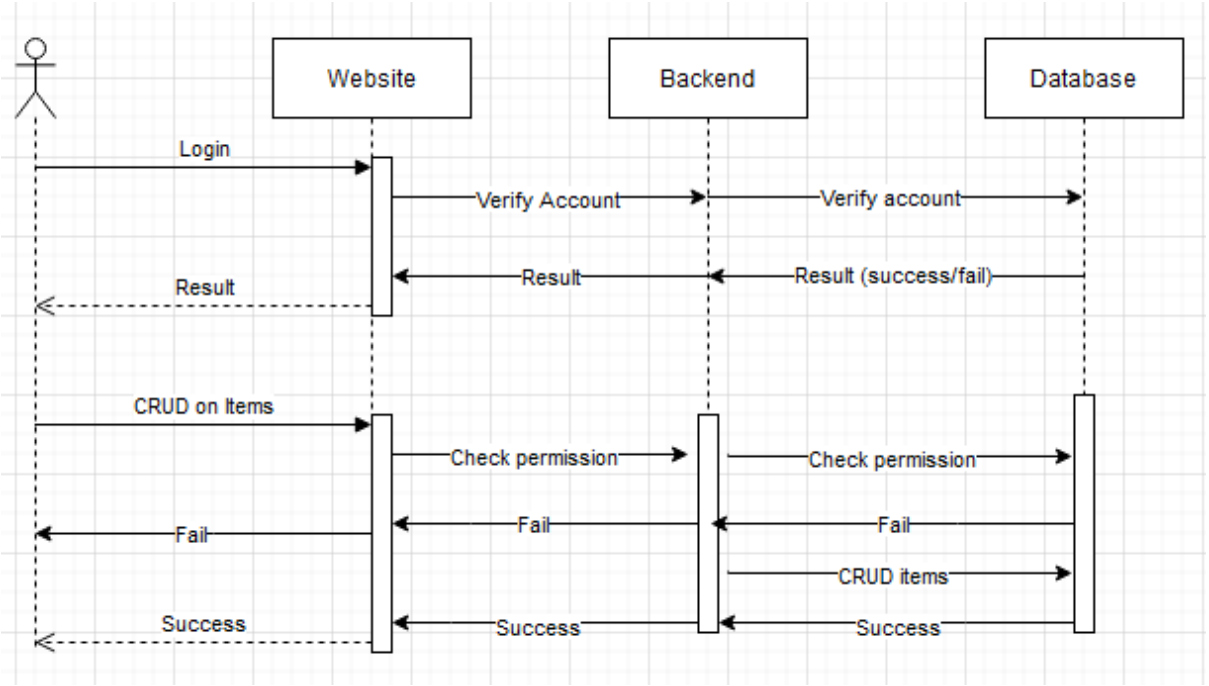
Checkout

This workflow requires the Customer to be logged in, and have at least 1 item in their Order catalog
First, the frontend will request additional checkout information like the Payment method that the Customer wishes to use for this transaction, as well as additional information like address, pick up time,…

The order information, along with the additional information collected will be sent to the backend server for processing. The processing procedure includes creating records, modifying quantity, payment API, updating status. When all of those procedures are completed, the User will be updated with the status of the Order.

When the payment and Checkout process is complete, a Transaction record is then created. This process is done internally, and the Transaction info will be saved. The Customer can request these Transaction information at will

Admin Sequence Diagram



Admin Sequence Diagram

This sequence Diagram is similar to the User Sequence Diagram, will display the interaction process between components of the app when a site Admin (Top Deal Auto) interacts with the website.

There are 2 workflows that we want to focus on

Login

This process is very similar to the Customer login. But the Admin will perform this process in their own Admin Portal, which is not accessible to anyone outside of Top Deal Auto.

The underlying process is similar though, as the credentials entered in the frontend will be sent to the backend to be verified with the Database. The result of this process will be sent back to the Front end and displayed to the site admin.

CRUD on items

This process requires an Admin user to be logged in to the Admin portal

This process is a shorter term for describing administration tasks performed by the site admin. These tasks may include but are not limited to: Changing item quantity, item details, modifying user records,…

When an Admin performs this task, the request is sent from the Frontend to the backend, the backend then will check if the Admin user has enough permission to perform the requested action. If the Admin's privileges are not sufficient, an error message will be returned and displayed to the Admin through the frontend.

Otherwise, if the Admin has enough rights, the action will be performed and reflected on the database.

Team Discussion	
Pham Anh Vu	The designing process, especially the Sequences, while I believe could be more decoupled with the use of external 3rd party managed services, given the budget of the project, both monetary and time. I am content with the solution that we were able to come up with.
Nguyen Thanh Dat	I share Vu's viewpoint that incorporating third-party managed services could have further decoupled our design process, particularly in the Sequences aspect. Given our project's financial and time limitations, though, the solution we arrived at does indeed meet our needs effectively. I too am pleased with the outcome we've achieved under these constraints.
Pham Duc Linh	I echo Vu's perspective on the potential benefits of integrating third-party managed services, especially concerning Sequences. Despite our project's constraints in terms of finances and time, I agree that the solution we have devised adequately fulfills our requirements. I am also satisfied with the outcome we have attained within these limitations.
Tran Tuan Nam	I find the Class Diagram and Sequence Diagrams to be well-structured and informative. The Class Diagram neatly defines the relationships between users, cars, orders, and transactions, while the Sequence Diagrams effectively depict the flow of interactions between customers and administrators. The design emphasizes clarity and functionality. However, to ensure the system's reliability and security, robust error handling and stringent security measures, particularly in processes like authentication and CRUD operations, will be crucial considerations.
Phung Xuan Tung	I fully support the streamlined process for adding cars to orders. When a user adds an item to their order via our website's frontend, the request is promptly sent to the backend. The backend performs essential checks, such as verifying item validity (quantity or existence). If the item is invalid, an error message is immediately relayed to the frontend, preventing the addition to the order. However, if the car exists and is eligible for inclusion, it seamlessly becomes part of the user's order catalog. A success message is then communicated to the user through the frontend, ensuring a smooth

	experience.