AYULLA

Ayulla Software Engineering Tryout Task (ASETT)

GENERAL INSTRUCTIONS

• You will have 2 weeks to complete this task from the time you receive it. You will receive a final data and time of submission from your respective recruiter.

• You may submit the task before the due date and not necessarily on the due date.

• Only two resubmissions are allowed. Every new submission will render the previous one obsolete.

• Your code will be judged by black box software testers who may have little or no knowledge of the language you used in the task. You must include a README file that has execution instructions.

• If you have a database that must be populated for your system to work, you must include the steps for populating the database and sample data for your database.

• You may or may not include notes on your implementation that describe steps you took in accomplishing the task

• Add comments to your files as needed appropriately.

As part of Ayulla's Software Engineering Tryout Task (ASETT), you are required to implement a basic but scalable Car Supply Chain System that connects Car Manufacturers, Car Dealerships and Customers.

Your system is expected to be abstract, hence it is important that you skip implementation of minor details.

The system must be built to the specifications of this document, although you can implement extra features if you have fulfilled the requirements.

The full completion of this task is not necessary; however, significant progress towards completion should be demonstrated.

The following list outlines the required coding restrictions for the system:

• A database should be created along with a corresponding schema showing the relationship between the various components. The database must be fully decomposed if it's relational.

• The database must contain at least 10,000 Cars.

• The frontend should utilize Django’s template tags to reduce the amount of redundant html code. Demonstration of some custom template tags is encouraged, but not required.

• URL configurations should be easily maintainable, so slight changes to the paths will not break the code in the views.

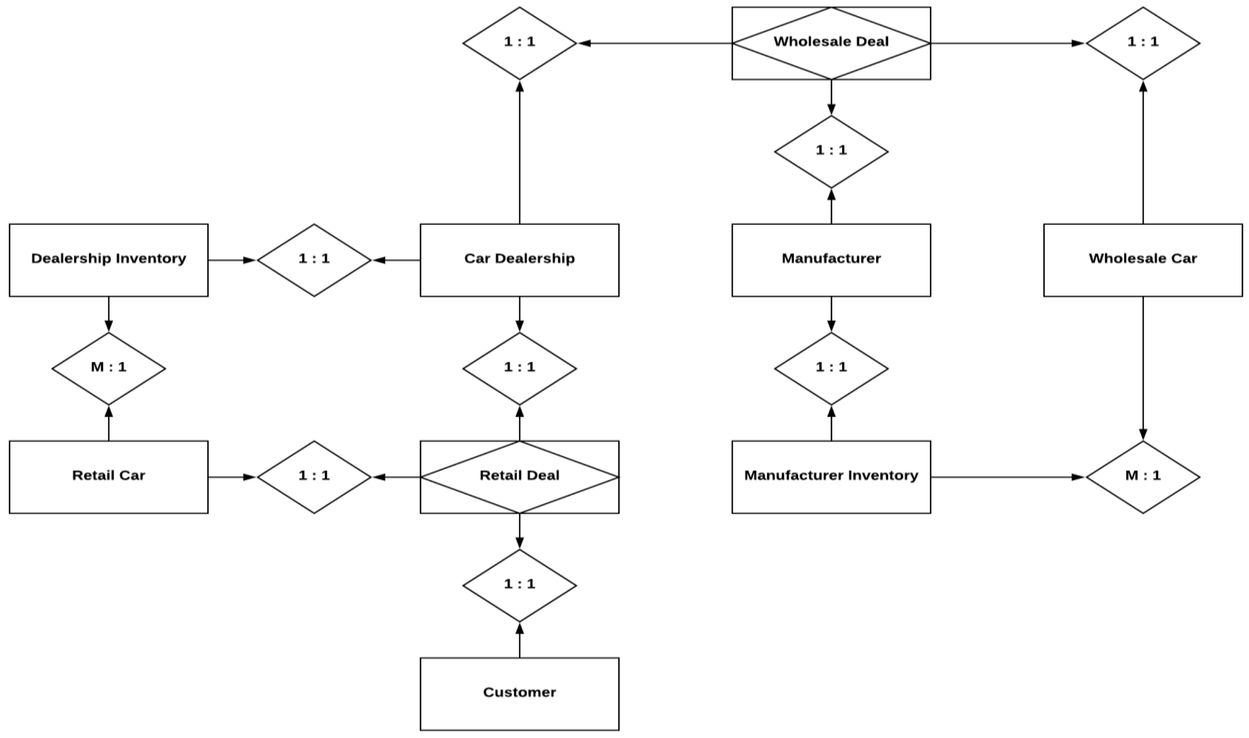
• All implemented views and models must have corresponding unit tests.

• An appropriate admin site should be implemented through which users and various components can be added and monitored.

• A reasonable understanding of class-based views must be demonstrated in the implementation.

• The entire project must be packaged with the appropriate python virtual environment.

SYSTEM COMPONENTS



A Diagram demonstrating the various components along with the relation with one another

Overview

The Supply Chain starts with the Manufacturer, a company that makes Wholesale Cars and ships them to Dealerships.

The Manufacturer Admin would initiate a Manufacturing Order, specifying a Blueprint and the amount of Cars to be manufactured.

This would result in the following:

- Calculating total manufacturing cost, which would be equal to the specified Blueprint's manufacturing cost multiplied by the manufactured amount
- Deducting total manufacturing cost from the Manufacturer's Balance
- Creating Wholesale Cars equal to the specified amount
- Adding created Cars to the Manufacturer's Inventory

The Dealership Admin submits a Wholesale Deal application to the Manufacturer he/she's willing to work with, specifying the Car's name, amount and total cost.

The Manufacturer Admin of the specified Manufacturer reviews the application and either accepts or refuses it.

If the Manufacturer Admin accepts the Deal, the following would occur:

- An amount of Wholesale Cars equal to the amount specified in the Deal application and have the same specifications as mentioned in the application would be removed from the Manufacturer's Inventory and added to the Dealership's Inventory as Retail Cars
- Total cost would be deducted from the Dealership's Balance and added to the Manufacturer's Balance

Else if the Manufacturer Admin refuses, the application shows up to the Dealership Admin as "Rejected".

The newly purchased Cars would show up to the Customer as available along with the available stock.

The Customer would decide to buy a Car and submit a Retail Deal application to the Dealership that owns the Car he/she's willing to purchase, specifying the Car's name and the price he/she's willing to pay.

The Dealership Admin of the specified Dealership reviews the application and either accepts or refuses it.

Should the Dealership Admin accept the Deal, the following would occur:

- 1 x Car of the specified name would be removed from the Dealership's Inventory
- The price specified in the Deal application would be deducted from the Customer's Balance and added to the Dealership's Balance

Else if the Dealership Admin refuses, the application shows up to the Customer as "Rejected".

Breakdown

- **Manufacturer :** A company that manufactures Wholesale Cars and distributes them to Dealerships
  Required Traits: name, country, balance

- **Manufacturing Order:** An order initiated by a Manufacturer Admin to add Wholesale Cars to the Manufacturer’s Inventory using a Blueprint
  Required Traits: car count

- **Blueprint :** A car model that’d be used in a Manufacturing Order
  Required Traits: name, manufacturing cost

- **Dealership :** A store that sells Cars to interested Customers
  Required Traits: name, country, balance

- **Wholesale Deal :** A deal that a Dealership Admin initiates to purchase Wholesale Cars from a Manufacturer via a Manufacturer Admin
  Required Traits: car name, car amount, total price

- **Manufacturer Inventory :** A component that keeps track of what Cars are currently owned by a Manufacturer

- **Dealership Inventory :** A component that keeps track of what Cars are currently owned by a Dealership

- **Retail Deal :** A deal that a Customer initiates to purchase a Retail Car from a Dealership via a Dealership Admin
  Required Traits: car name, price

- **Wholesale Car :** The naming of cars in a Manufacturer’s Inventory
  Required Traits: name, wholesale price

- **Retail Car :** The naming of cars in a Dealership’s Inventory
  Required Traits: name, retail price

USERS

A user refers to an entity who will use the system under design to achieve a certain goal. The user is the most important part of the system as they will add data, modify data, destroy (remove) data, and retrieve data from the system. The kind of data a user may interact with and the process of this interaction is solely dependent on the permission of user. Users must have a role assigned to them. The role of a user determines the degree of their permission. They are three major roles in the system: Manufacturer Admin, Dealership Admin, and Customer. It is important to implement all three major roles.

The attributes and permissions of these roles are described below:

- **Manufacturer Admin**: a Manufacturer representative in the system whose role is to maintain his/her Manufacturer's data.

  A Manufacturer Admin can do the following:

  - Initiate a Manufacturing Order
  - Accept/Refuse a Wholesale Deal
  - Create/Edit/Delete a Blueprint
  - Modify/Remove Cars from his/her Manufacturer's Inventory
  - Add Balance to his/her Manufacturer's account

  However, he/she cannot do the following:

  - Contact a Customer in any way
  - Initiate Wholesale Deals
  - Initiate Retail Deals

- **Dealership Admin:** A Dealership representative in the system whose role is to maintain his/her Dealership's data.

  A Dealership Admin can do the following:

  - Initiate a Wholesale Deal
  - Accept/Refuse a Retail Deal
  - Modify /Remove Cars from his/her Dealership's Inventory
  - Add Balance to his/her Dealership's account

- **Customer:** An interested buyer willing to conduct Retail Deals with Dealerships.

  A Customer can do the following:

  - View Cars showcased by Dealerships
  - Offer a Retail Deal to a Dealership
  - Add Balance to his/her account

However, he/she cannot do the following:

- Contact a Manufacturer in any way
- Initiate a Wholesale Deal

USER INTERFACE AND USER EXPERIENCE

You will not be provided with any UI/UX frameworks and/or wireframes. Your creativity is judged here. You should be able to display your data to the different users as well as have buttons, controls, and forms that the users can use in their interaction with the system. You can exhibit any kind of skill set in this section without restrictions. However, you must have at least an abstract user interface that can allow a user to communicate with the system. Do not spend most of your time working on a UI. Although it is important, your skill is mainly judged on your implementation of the backend of the system.