Group name: Rajesh BabuDNarayanA

Members: Dheekksha Rajesh Babu, Adithya Narayan

Online Food delivery system

- Introduction

This project aims to act as a database system for a food ordering application. Users of this application can build an order by selecting a restaurant, and selecting a subset of items available at the chosen restaurant. They can then also choose to apply a discount if it's applicable given the customer and the validity of said discount. A price calculation engine comes up with a price that is displayed to the customer, who can then choose to go ahead and confirm the order, or cancel it. Once the customer confirms the order, it is processed and a delivery agent is contacted in order to find a delivery driver for this specific order. Once someone is found, they are sent to the restaurant to pick up the delivery and in the process gather the customer details. Using these details, the driver hands the delivery to the customer, and the customer pays the driver. The customer also rates the delivery, as well as the parcel that has been delivered to them.

The system provides a UI with logins for users, delivery drivers and for restaurant manager admins.

Users (customers) of the application have access to their own personal information and contact details that they can modify. They can view restaurant options when they login and each food menu associated with it. Customers can then add the items to their cart (add, delete, modify quantity) as they see fit. Order is associated with only 1 customer and 1 restaurant. After order is placed, a driver is assigned to it by the delivery agent.

The driver login enables the driver to see the restaurant for pickup, the customer contact and address for delivery as well as total amount.

The restaurant manager login can see and edit details about the restaurant and the food items that are available that day for order.

Rates are calculated based on distance, weather and the discount provided by the customer. The amount is first shown to the user and on confirming the order is placed. The amount is paid to the driver. After this transaction the driver can mark order as completed and the order is archived into history. After this the order and delivery can be rated.

On top of this, we wish to also add a few data visualization functionalities like:

1. Most Popular Restaurant
2. Average Delivery Times in Quartiles
3. Identifying Items with the best/worst delivery reviews
4. Identifying Most common menu items and ingredients
5. Vegetarian Vs Non-Vegetarian Customer Population
6. Item vs ETA
7. Weather vs ETA

- Why does this Project/Data Domain Interest You?

Food ordering systems benefit from a templatized approach to solving supply chain as a problem. There are a variety of challenging problems that pop up, be it scalability, trend mining or end to end implementation.

One of these challenging problems include the representation and processing of complex data types like Locations represented as latitudes and longitudes and Weather, that is represented as either Celsius/Fahrenheit.

Another one of these problems include the calculation and visualization of several real time trends. These trends help us better serve customers, and also improve end to end efficiency. It is also evident that these problems don’t really concern themselves with how exactly we perform these database operations and how we setup the database itself, but concern themselves with how quickly we can actually come up with these results for direct consumption, which according to us is the most interesting part of this entire discussion.

- Data description

The data that we aim to synthesize/gather depends upon the table/relation we wish to have data for at that moment.

For example, Restaurants have locations represented by their latitude’s and longitude's. There are ways to scrape a few hundred restaurants and their locations off the web and store it in a single table.

Menus can be generated randomly out of a large selection of items, and ingredients can be manually discerned for most of the items.

The pricing engine is a formula that takes in weather data, distance from the customer's closest landmark, and a selected discount. Weather data can be updated for a given set of locations using the OpenWeatherMapAPI, and a series of landmarks with their locations can be scrapped. All of this is now fed to the pricing engine that calculates these values if necessary.

Operations that involve reviewing an item or a delivery experience can be randomized to have a number between 0 and 5.

- Software, Apps, Languages, Libraries and hardware that will be used to develop the project.

Database : SQL

We considered the following options:

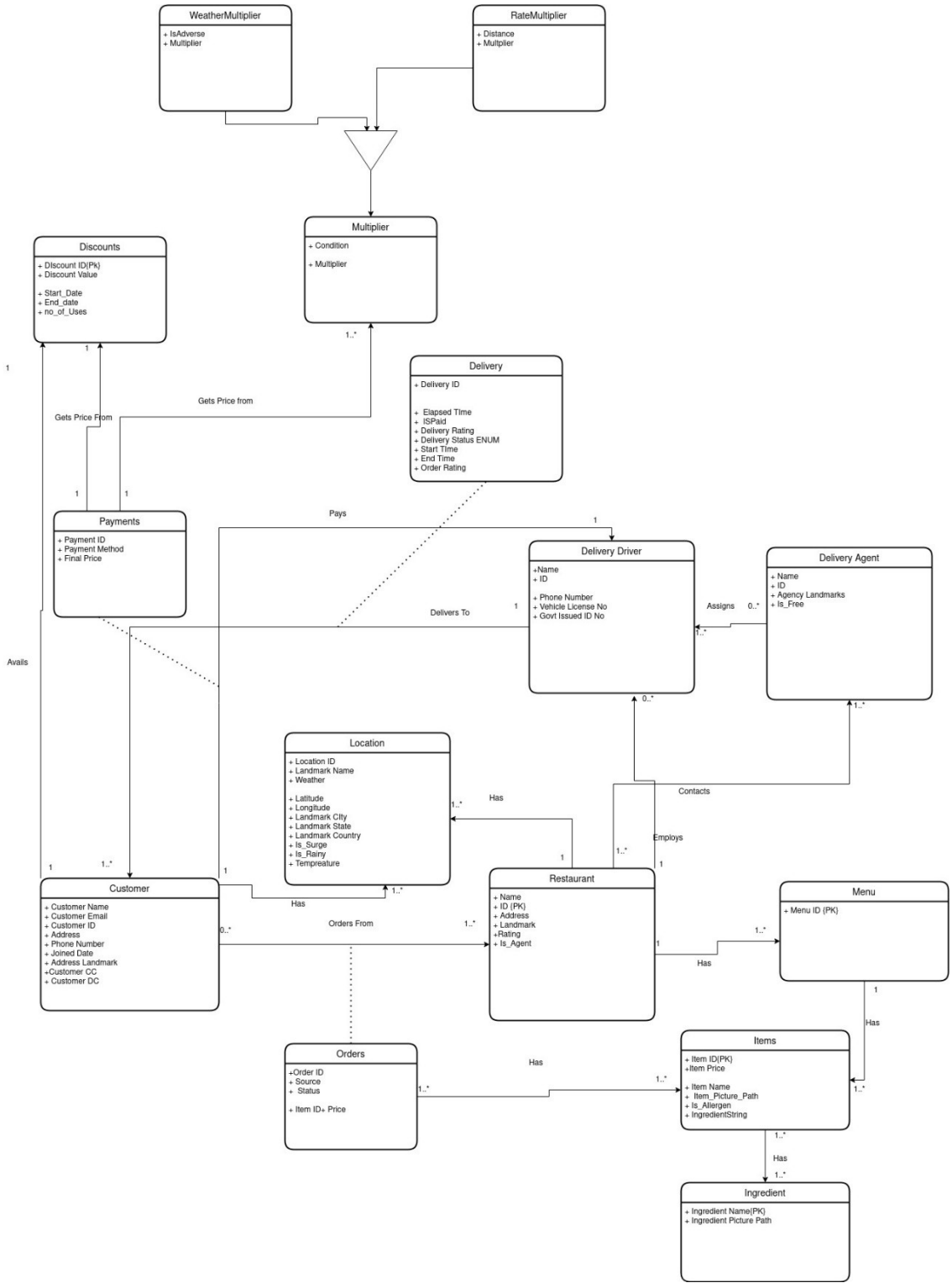
1. React frontend with Python Backend(GraphQL, MySQL, Flask/FastAPI/Django)
2. React Frontend with Java Backend(JDBC, MySQL, Springboot)
3. React Frontend with JS backend(Express.js, Node.js, mysql.js)

We are looking for a stack that is:

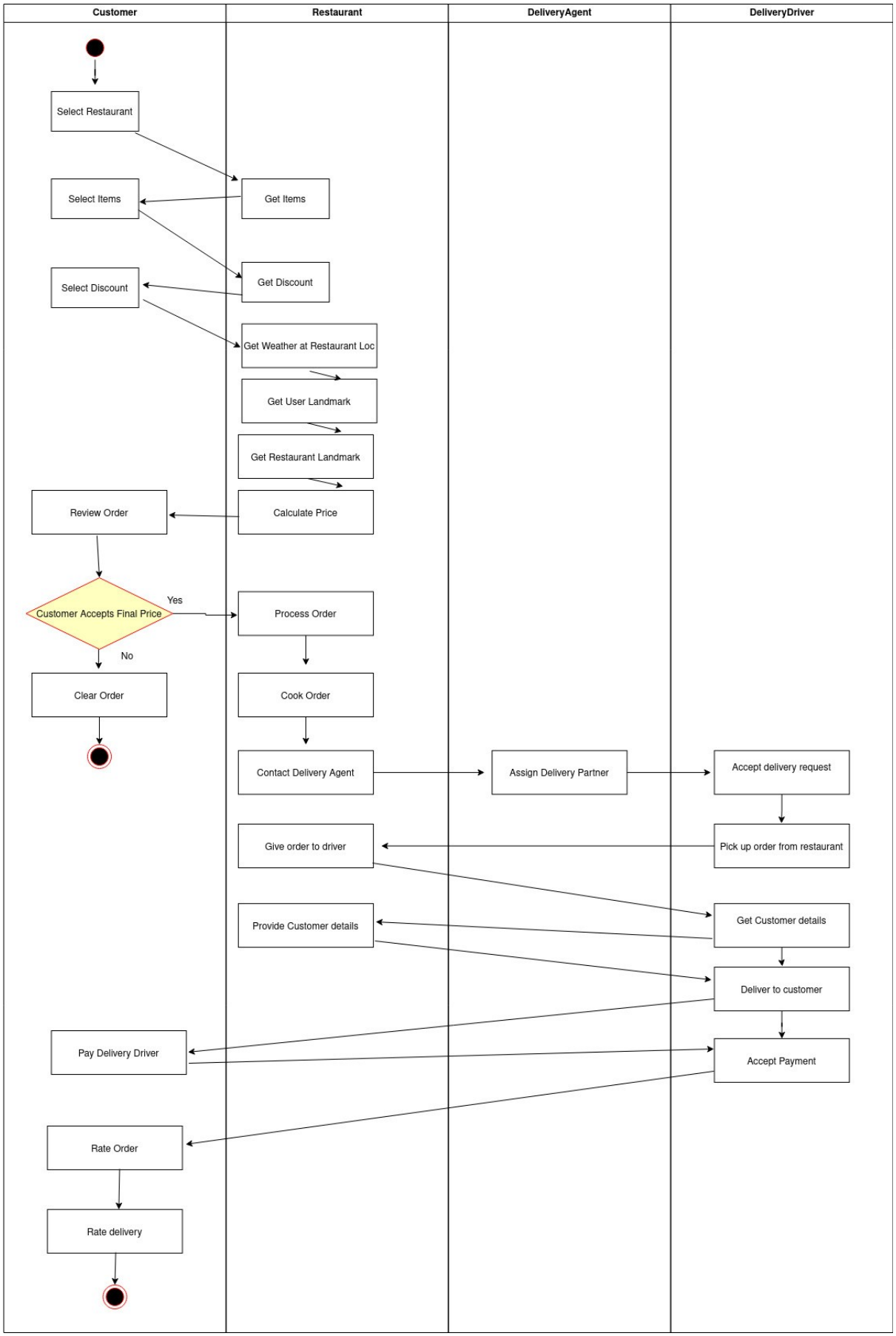
1. Scalable
2. Portable
3. Strongly Typed when it comes to the backend(Makes it easier to adhere to good coding principles)

Given the above criterion, we decided to go ahead with option 2.
We will be setting up this project on both a Linux and a macOS based system.
We will be using IntelliJ IDE to structure the project.
If there is any web data that we need scrapped, we plan to use bs4, a python based utility to filter and gather website data.
We will also be testing this project on a few simple test cases using JUnit.

- UML diagram



- Flowchart/ Activity diagram


Customer
Restaurant
DeliveryAgent
DeliveryDriver
Select Restaurant
Get Items
Select Items
Get Discount
Select Discount
Get Weather at Restaurant Loc
Get User Landmark
Get Restaurant Landmark
Review Order
Calculate Price
Customer Accepts Final Price
Yes
No
Process Order
Clear Order
Cook Order
Contact Delivery Agent
Assign Delivery Partner
Accept delivery request
Give order to driver
Pick up order from restaurant
Provide Customer details
Get Customer details
Deliver to customer
Pay Delivery Driver
Accept Payment
Rate Order
Rate delivery