Project Report

Title:

Go4Food Project Report

Group Name:

Go4Food

Group Members:

Mohammad Tamanna | Mohammad.Tamanna@outlook.com

JJ Papeleras | justinejpapeleras@gmail.com

Adnan Uddin Mohammed | ami949@uregina.ca

Department of Computer Science

CS 476-02: Software Development Project

Spring/Summer 2023

Sirvan Parasteh

28th July 2023

Table of Contents

1. Project Title

Go4Food - an online food ordering system application.

2. Project Description

Develop a web-based application that allows users to order food from various restaurants. Implement features such as user registration, menu browsing, placing orders, and order tracking. Highlight benefits like convenience, a wide range of restaurant options, and user-friendly interfaces.

3. Problem Statement

3.1 Problem Definition

Conventional food ordering strategies lead to some inconvenience for many customers and also operational challenges for restaurant owners because of various reasons such as lengthy phone calls, miscommunications, and limited menu visibility. Furthermore, managing and tracking orders manually can lead to errors, delays, which affect customer satisfaction. The Go4Food project provides a user friendly platform within the e-commerce domain that provides effortless ways to explore menus, customize orders, and place them through frequencies decided and customized by the user. In doing so, it addresses the issues mentioned above.

Go4Food's online food ordering system aims to addresses these challenges present in already existing food ordering process. The primary objective is to develop a user-friendly web-based platform that increases customer satisfaction by providing a streamlined ordering experience. This online platform allows customers to not only place orders from various restaurants but to also do so with additional features such as customizable orders, in doing so combatting existing methods such a singular order.

3.2 Project Motivations

A key motivation behind this project is to offer convenience to customers. The current food ordering process in existing systems is repetitive and standardized. In that sense, it is good as consistency relative to features and procedures that are globalized is a positive. However, there is still room for improvement as the choice for additional features does not necessarily impede this consistency. By providing an option for users to set a frequency for a particular order – have it happen once, twice, or however many times per week – implements this feature while maintaining existing user experience from traditional order placement methods.

3.3 Application Benefits

Go4Food online food ordering system offers many benefits compared to existing systems such as SkipTheDishes and Uber Eats. It manages to do this by utilizing advanced features and functionalities. Therefore, the application increases customer experience and convenience, providing value for both customers and also restaurant owners.

One of the features offered by Go4Food is allowing users to plan and place customized orders. This feature enables orders to set up frequency of reoccurring orders. For instance if a user wants to place the same order every Monday, they don't have to go through the same process every week. Instead they do it once, set their weekly frequency, and they have successfully customized their order. In doing so, the system makes a drastic change in how people interact with online food ordering, while also enhancing overall convenience and efficiency. This feature helps those with a routine and also those who want to save time by automating their regular food orders.

Go4Food recognizes that customers all have their own preferences and routines concerning their meals. As far as a solution goes, the online food ordering system offers a more personalized customer experience. It could be a daily breakfast order, weekly meal plan, the system allows customers to not only select what type of food they want to order, but also customize how they order it. This introduces convenience and practicality to their routines.

This feature helps customers but it also provides advantages to restaurant owners. To take it into perspective, now a restaurant owner is able to view an order that is meant to be for next week. This introduces a layer of predictability for them which in return allows them to more efficiently setup their production, when it comes to ingredients and allocating resources to particular orders. The feature not only increases their productivity but also reduces operational cost, as less time will be spent when something is predictable and resources will be optimized. Go4Food also introduces a key feature in two-factor authentication (2FA) everytime an order is placed. This provides extra security in case there is unauthorized access on a user's access. 2FA will protect their key information such as credit card and personal details.

Two-factor authentication acts as a security measure by asking another form of identication for users to effectively use their accounts to place orders. The customer will be prompted to enter a second factor, such as a unique verification code sent to their mobile device. This is of course supplementary to the already existing login credentials when logging into their account.

Existing food ordering systems such as Uber Eats and SkipTheDishes give the option to save credit card information. If unauthorized access is made to the respective UE or STD account, and the user has previously saved their credit card info (as is possible on these services), the person with unauthorized is able to place orders for themselves and even if not, view key personal details that should not otherwise be shared with anyone else. This provides the base reasoning for the implementation of this feature within Go4Food.

4. Functional Requirements

4.1 Functional Requirements List and Table

Requirement ID	Requirement Name	User Role	Implementation Status	Instructor Note
R-01	User Registration	Customers	Implemented	
R-02	Browse Restaurants	Customers	Implemented	
R-03	Search For *Restaurants*	Customers	Implemented	
R-04	View Menu	Customers	Implemented	
R-05	Place Order	Customers	Implemented	
R-06	Track Order	Customers	Implemented	
R-07	Order History	Customers	Implemented	
R-08	User Reviews	Customers	Implemented	
R-09	Pre-Planned Order	Customers	Not Implemented	
R-10	Order Support	Customers	Implemented	
R-11	Share Order on Socials	Customers	Not Implemented	
R-11	Restaurant Registration	Restaurant Owner	Implemented	
R-12	Menu Management	Restaurant Owner	Implemented	
R-13	Order Notification	Restaurant Owner	Not Implemented	
R-14	Order Processing	Restaurant Owner	Implemented	
R-15	Restaurant Analytics	Restaurant Owner	Implemented	
R-16	Customer Communication	Restaurant Owner	Not Implemented	

4.2 Implemented Requirements

• User Registration: Customers can create user accounts, provide personal information like email, password, and name.

• Browse Restaurants: Customers can look through a list of available restaurants.

• Search for Restaurants: Customers can search for specific restaurants according to their name or cuisine.

• View Menu: Customers can look through the menu for each restaurants, viewing available dishes and their prices.

• Place Order: Customers can select their dishes and the number of them, add them to a cart, and then be able to place their order.

• Track Order: Customers can track the status of their order, and view whether they're in preparation, in transit, or already delivered.

• Order History: Customers can look at their past orders and review them, and also reorder any of their previous orders.

• User Reviews: Customers can leave reviews for restaurants or individual dishes to share their dining experience.

• Pre-Planned Order: Customers can customize their orders and have them placed at particular intervals such as X amount of times per week, or on specific days.

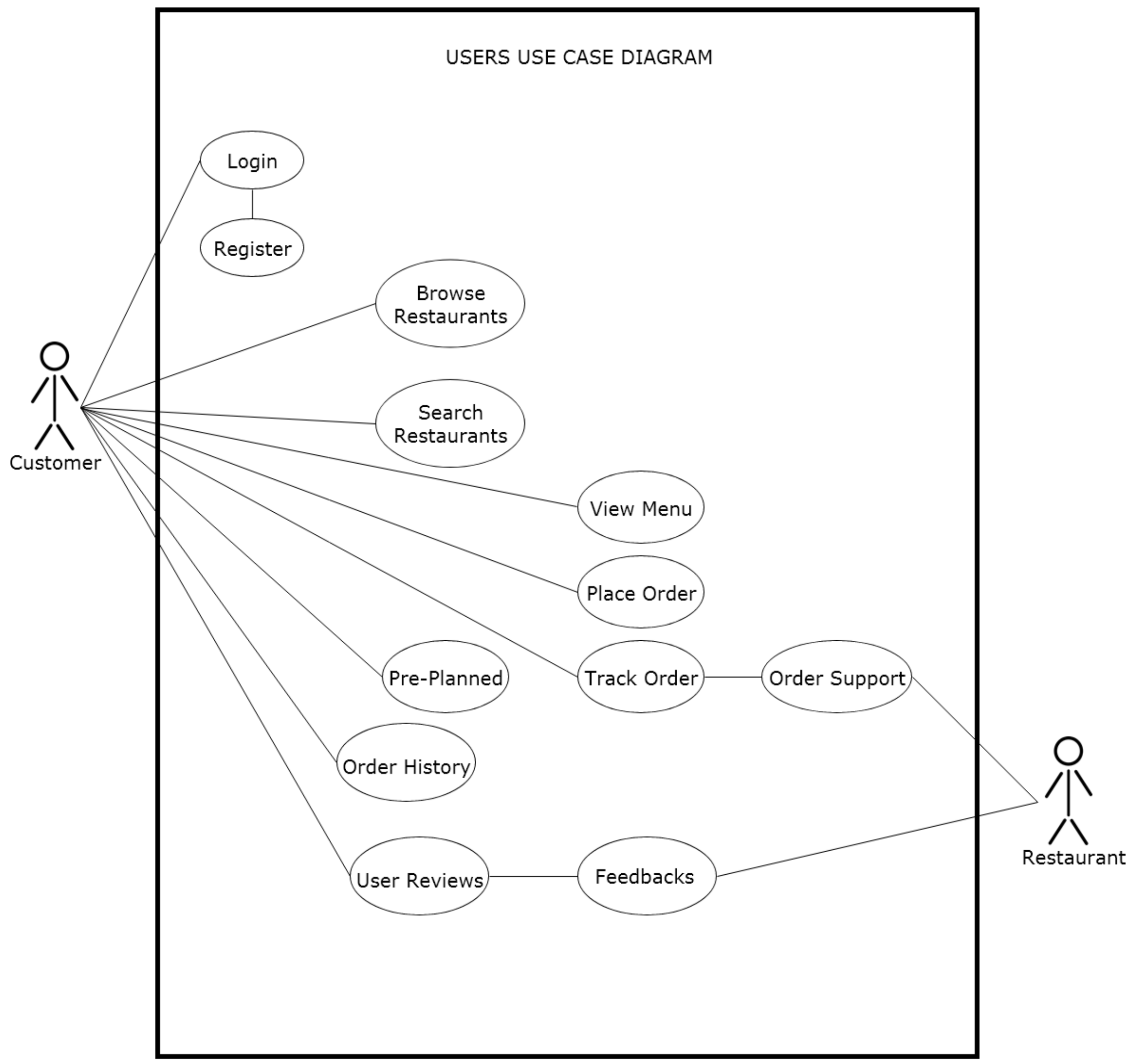
• Restaurant Registration: Restaurant owners can also register their restaurants on their system by providing necessary information such as restaurant name, address and type of cuisine.

• Menu Management: Restaurant owners can manage their menus, for example add new dishes, remove them, update and also change their descriptions, price, and availability.

• Order Processing: Restaurant owners are able to process orders, for instance have some sort of order confirmation, and also be able to change order status.

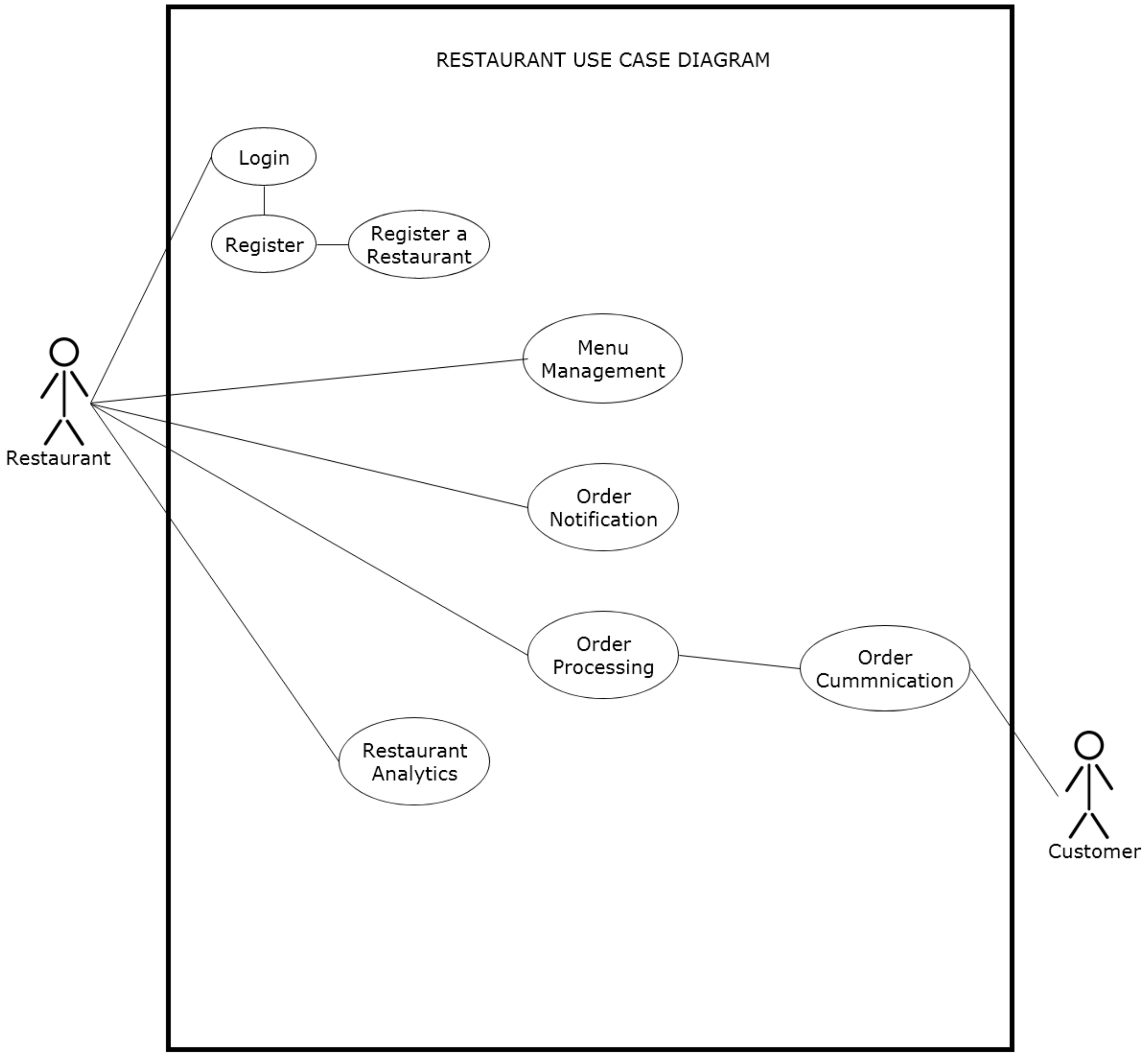
4.3 Use case diagram

4.3.1 Customer Diagram



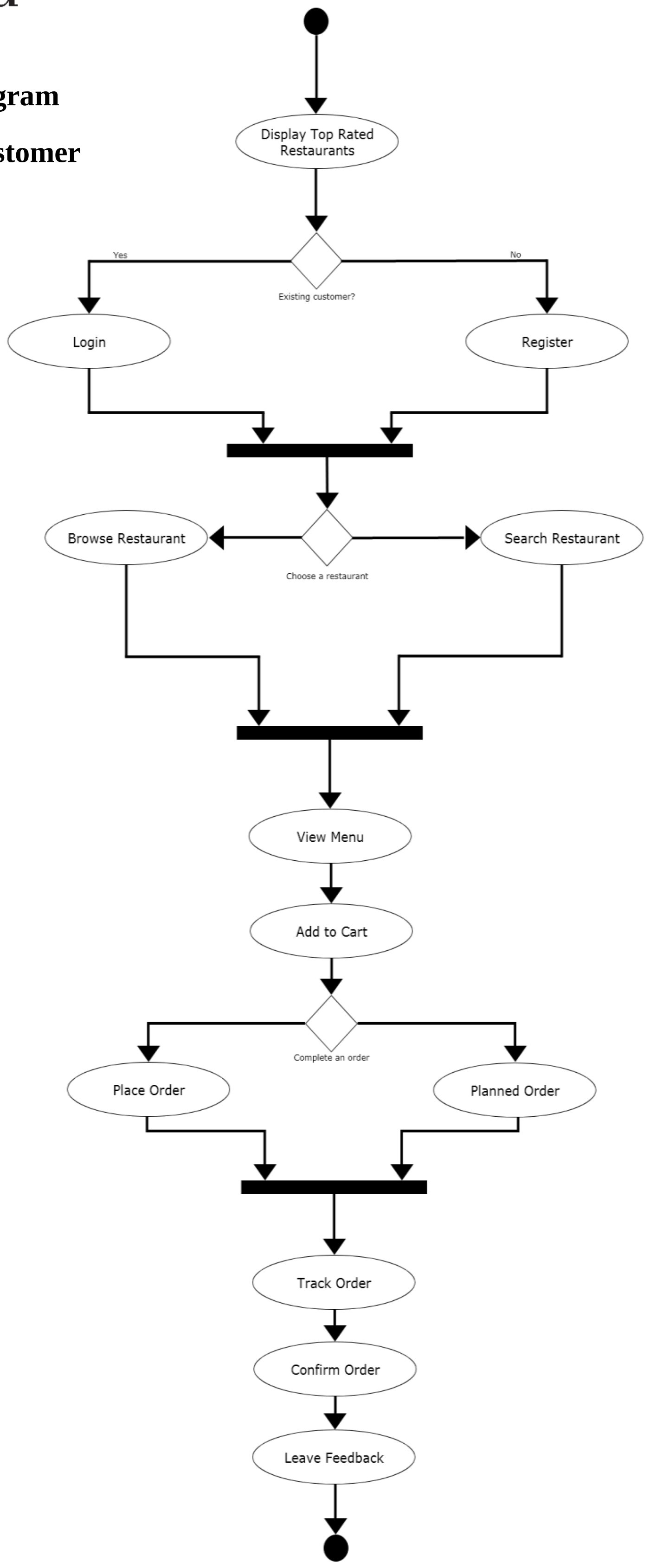
University of Regina

4.3.2 Restaurant Diagram



4.4 Activity diagram

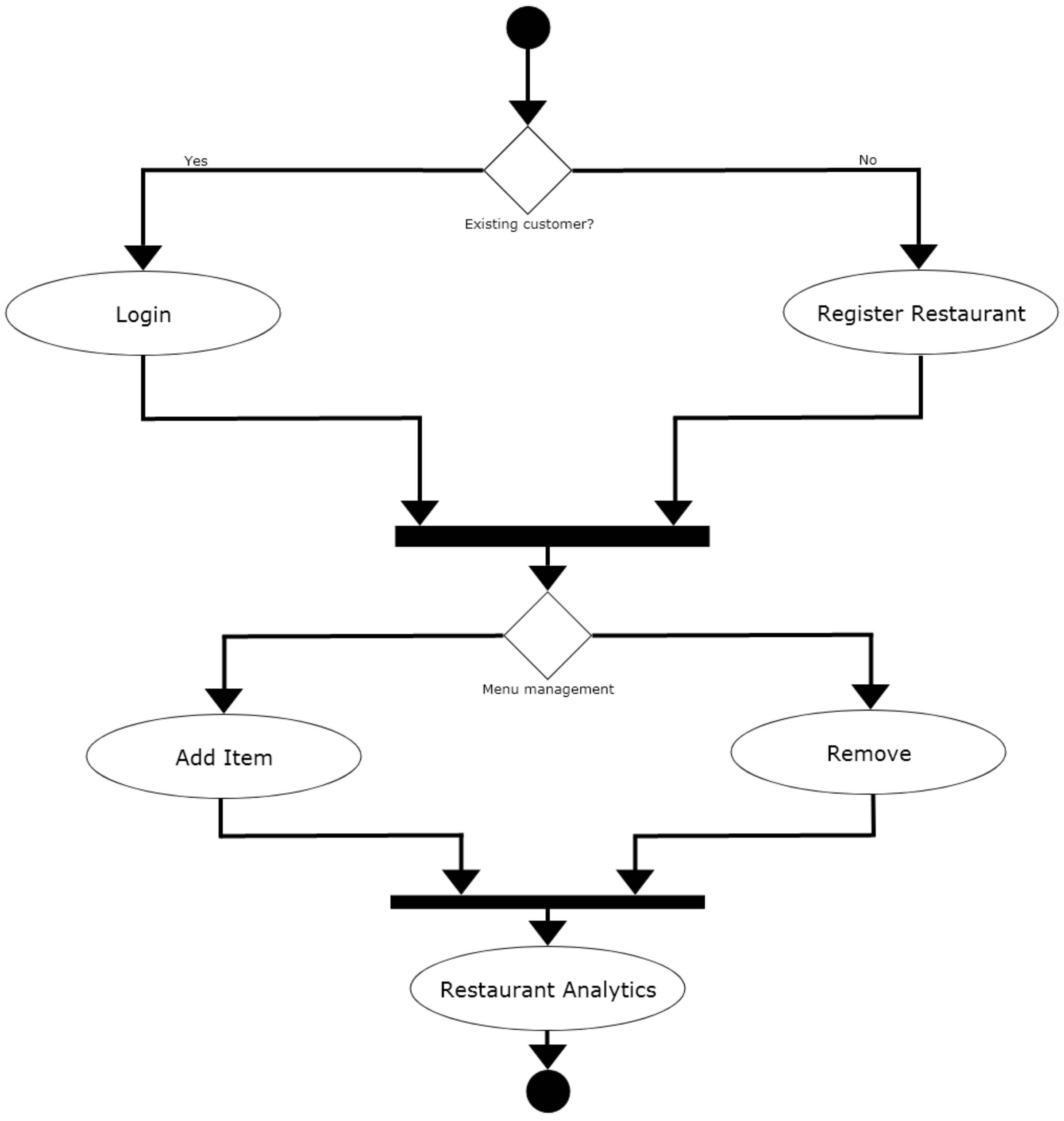
4.4.1 Customer



The customer/users will land on the landing page of the web application where top rated restaurants will be shown, then the user will have to login or create a profile in order to gain further access into the application and order food. Once it is done they will proceed and land on the main page where they get to search a specific restaurant or browse around. Then once a user has chosen a specific restaurant, they will be able to view that restaurant's full menu item and be able to add to cart. Finally once the user has decided what to order they will be able to place an order or have it as a pre-planned order where they can decided a specific time and dates and the order will go through automatically given by the users input. Immediately after placing the order, the user will then go to the track order page where they will be able to track the orders full process such as the restaurant confirming the order, preparations, and courier pickups. The user and restaurant will then be able to communicate regarding the order through order communication if they so chooses. Lastly when the items are delivered to the given address, the user can confirm the order and leave a feedback directly to the restaurant and even be able to share the feed through their chosen social media platforms.

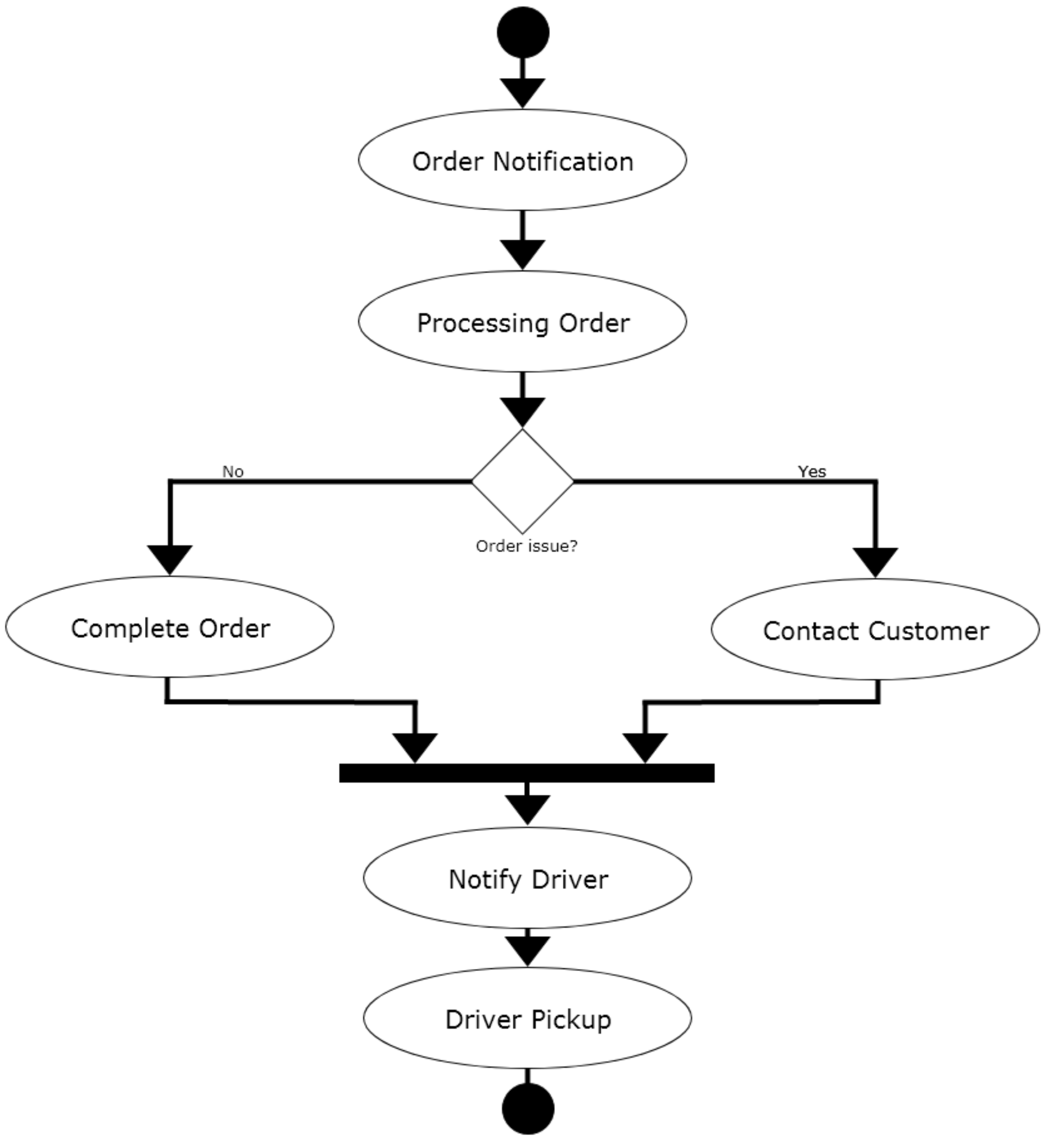
4.4.2 Restaurant Registration and Menu Management

The diagram shows the process of a restaurant login in if they are an existing customer and if not they will prompt to register the restaurant. Once a restaurant has registered and completed all the required information, they can procced through the application by signing in. They will land to the main page of the application, where the full menu and profile of the restaurant if visible. The restaurant can customize the menu in the menu management where they can remove or add items. Another option for the restaurant is the analytics board where it shows their top rated items, most ordered items and such.


Yes
No
Existing customer?
Login
Register Restaurant
Menu management
Add Item
Remove
Restaurant Analytics

5.4.2 Restaurant Order Process

The diagram below shows the restaurants order process. The restaurant will get an order notification whenever an order is made by the customer. Restaurant can view the extent of the order and they will move on to the processing of the order. If there is any problem with the order they can decide to contact the customer or the customer can contact them if there are any concerns or question regarding the order process. Once they are done fulfilling the order of the customer, they can complete the order and will notify the couriers and then wait for them to arrive in the restaurant to pick up.



5. Non-Functional Requirements

5.1 Software Qualities

5.1.1 Correctness

• For *Customers* - Order Accuracy: The system has to keep the choice of dishes, customization (optional), and also quantities of dishes when the order is placed, this is an example of correctness. Menu Availability: System should be able to display updated information regarding dishes, which is also updated, which then enables customers to view available dishes. This is also another example of correctness.
• For *Restaurant Owner* - Menu Consistency: The system ensures the correctness of the displayed menu for customers by accurately reflecting the dishes, prices, and descriptions set by restaurant owners. Order Accuracy Notifications: The system provides accurate order details to restaurant owners, including the selected dishes, customer preferences, and delivery/pickup instructions, ensuring correct order processing.
• For *Restaurant Owner* - Menu Consistency: When restaurant owners attempt to configure their dishes, prices, or their descriptions, the system should be able to accurately reflect those in the menu. This is an example of correctness. Order Status Accuracy: Restaurant owners should be able to change the status of an order, for example from in process to in transit, and for it to be executed accurately. This is yet another example of correctness for restaurant owners.

5.1.2 Time-Efficiency

• For *Customers* - Fast Search: For the implemented search feature, the system should be quick in handling customer searches for restaurants based on name and cuisine. Prompt Order Placement: The system should be user friendly in the process of selecting dishes, choosing quantity, etc. which in return increases timeliness.
• For *Restaurant Owner* - Efficient Order Notifications: The system promptly notifies restaurant owners about incoming orders, minimizing the time between order placement and order acknowledgment. *Streamlined Menu Management*: The system will allow the restaurant owners to manage their menus in an efficient manner, doing so by providing intuitive tools for adding, updating, or removing dishes. This streamlines the process of menu maintenance, saving time and effort for restaurant owners in keeping their offerings up to date.

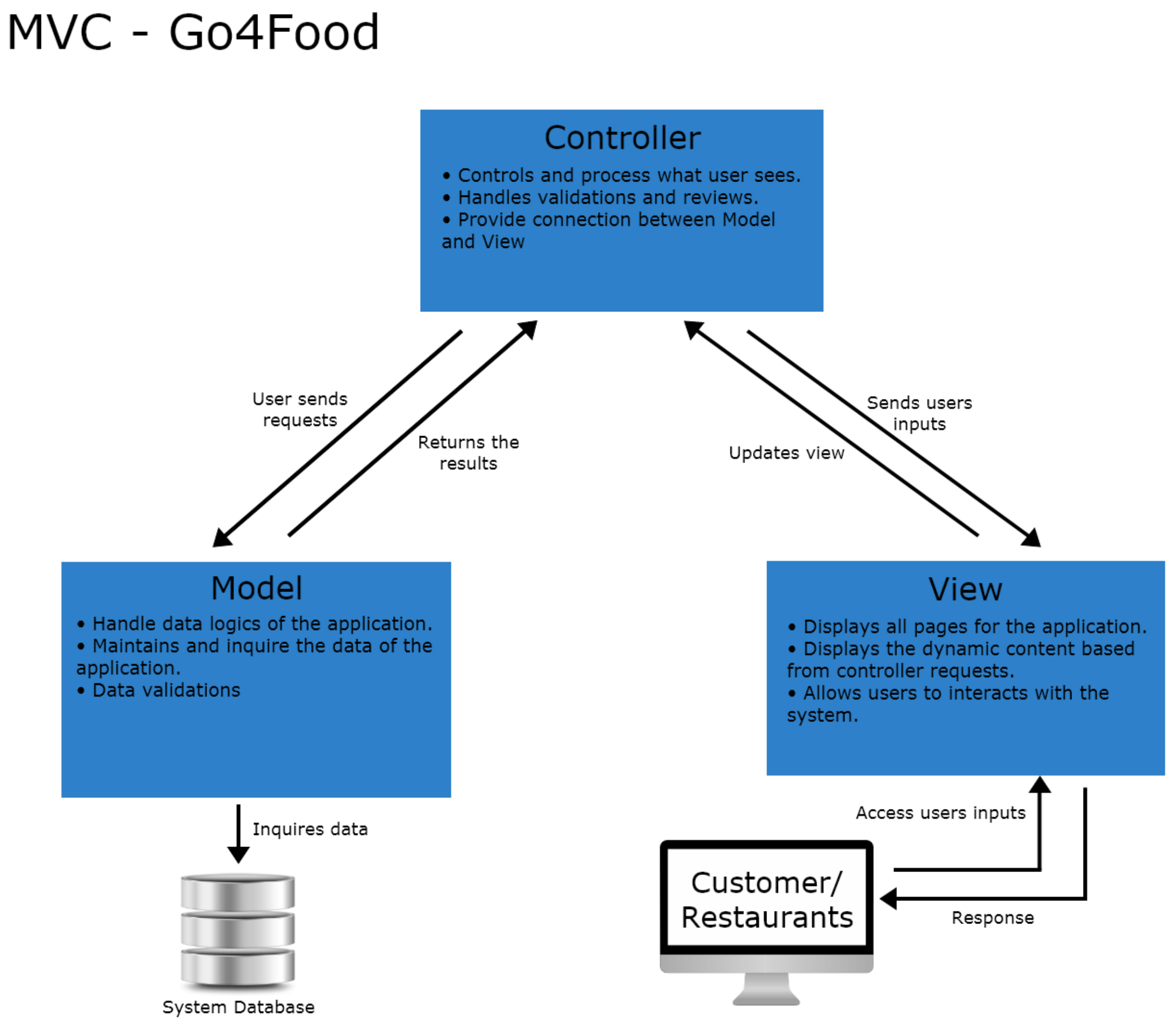
• For Restaurant Owner - Efficient Order Notifications: The system should be fast in notifying restaurant owners about incoming orders. The time between order placement and acknowledgement actually dictates the time efficiency in this application. Effective Menu Management Tools: Restaurant owners should have intuitive tools for making modifications to their menu like adding dishes, changing prices, etc. This will also improve time efficiency.

5.1.3 Robustness

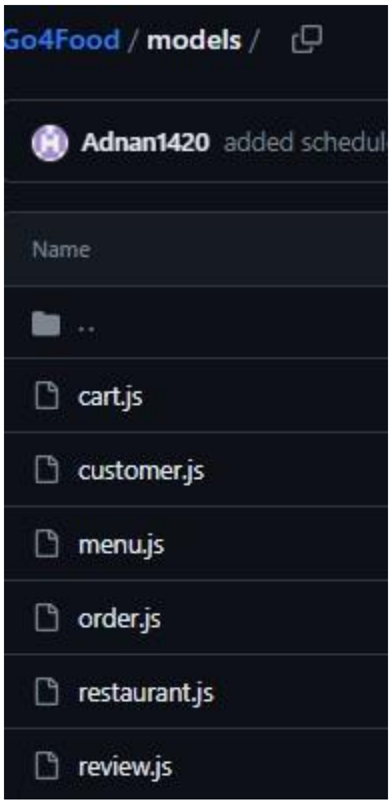
• For *Customers* - Error Handling: System should be able to handle some errors made by the user such as invalid inputs for password, or even invalid address format. System Stability: System should support many orders and be functioning in high traffic. Not sure how this will be implemented in a small application where manual entries are needed but doing so increases robustness.
• For *Restaurants* - Data Integrity: The system ensures the integrity of restaurant data, such as menu information and contact details, by implementing different things such as appropriate data validation and security measures. *Resilient Order Management*: The system should handle errors and it should handle any disruptions during the order management process, ensuring that order data is not lost or compromised, even in case of unexpected system failures.
• For *Restaurants* - Data Integrity: System should validate different inputs from the restaurant points of view, such as prices. Robust Order Management: If anything goes wrong during the order management process, such as a glitch during changing the order status, the system should be robust in the sense that it preserves already existing data.

6. Top-level and Low-level Software Design

6.1 MVC Architecture


MVC - Go4Food
Controller
• Controls and process what user sees.
• Handles validations and reviews.
• Provide connection between Model and View
User sends requests
Returns the results
Sends users inputs
Updates view
Model
• Handle data logics of the application.
• Maintains and inquire the data of the application.
• Data validations
View
• Displays all pages for the application.
• Displays the dynamic content based from controller requests.
• Allows users to interacts with the system.
Inquires data
System Database
Access users inputs
Customer/ Restaurants
Response

6.1.1 MVC Architecture Screenshots (Structure of the entire source code)



Go4Food / models /

Adnan1420 added schedul

Name

..

cart.js

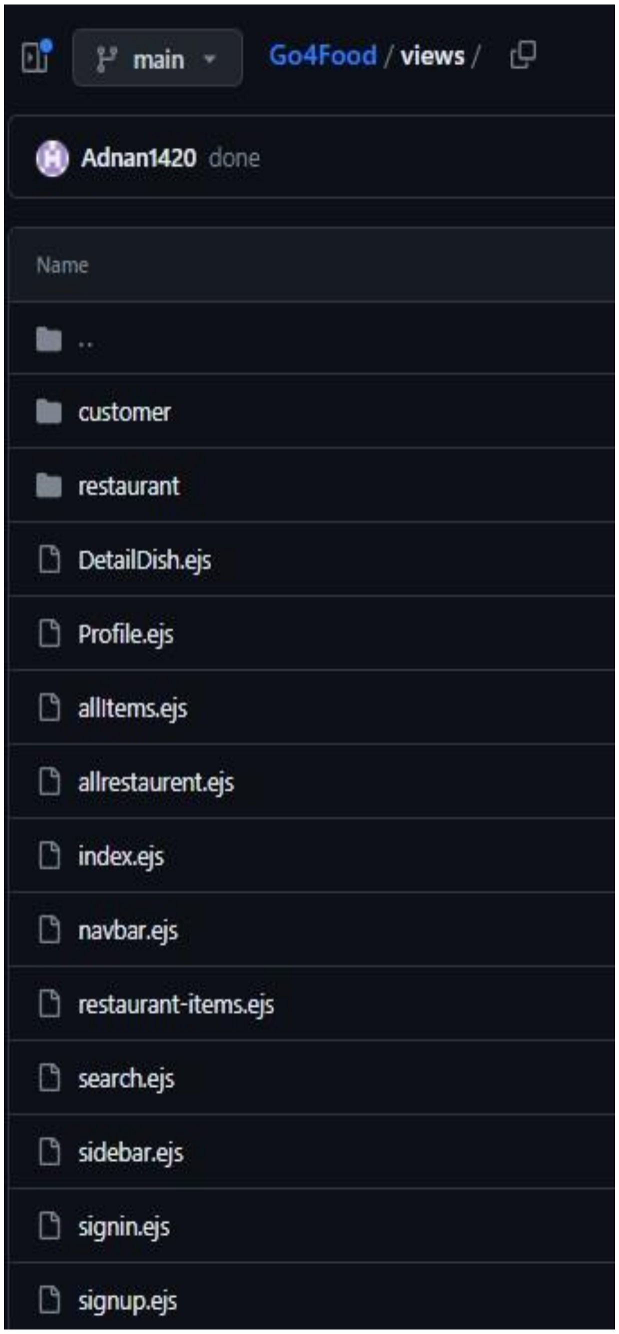
customer.js

menu.js

order.js

restaurant.js

review.js



main

Go4Food / views /

Adnan1420 done

Name

..

customer

restaurant

DetailDish.ejs

Profile.ejs

allItems.ejs

allrestaurent.ejs

index.ejs

navbar.ejs

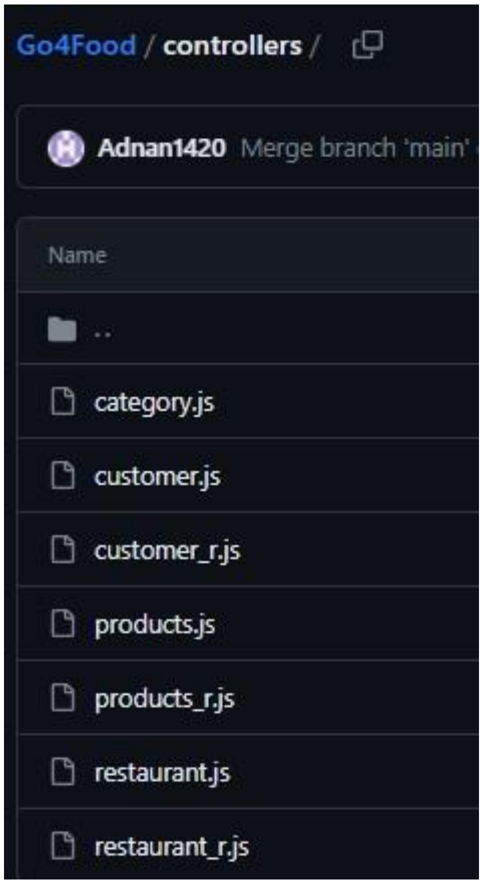
restaurant-items.ejs

search.ejs

sidebar.ejs

signin.ejs

signup.ejs



Go4Food / controllers /

Adnan1420 Merge branch 'main'

Name

..

category.js

customer.js

customer_r.js

products.js

products_r.js

restaurant.js

restaurant_r.js

6.1.2 Benefits

• Faster Development Process - since in the MVC architectures the codes are separated into three sections (View, Controller, and Data), it is possible for the work to be divided among group members. Thus, expediting the whole development process of the application. While maintaining efficiency as group members work on preferred components of the development.

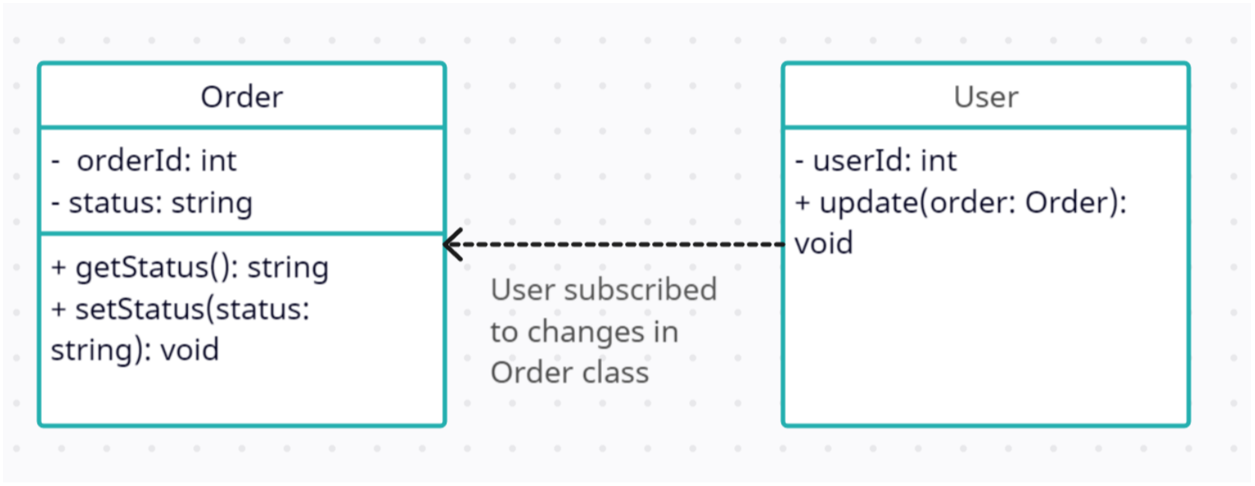
• Modifiable - the MVC architecture allows straightforward modifications as such when adding new or modifying functionality it does not affect the overall architecture of the application. Locating certain codes to be improved or revised can be quickly done so due to having sections. By doing so, the applications overall scalability and flexibility is much more improved.

• Decoupling of Codes - using the MVC architectures ensures decoupling of codes through the Model, View and Controller separation method. The View acts as the user interface of the application. Secondly, The Controller handles all user interactions. And, the Model provides the data part of applications. This separation allows code to be maintained efficiently and understood.

6.2 Design Patterns

6.2.1 Observer Design Pattern

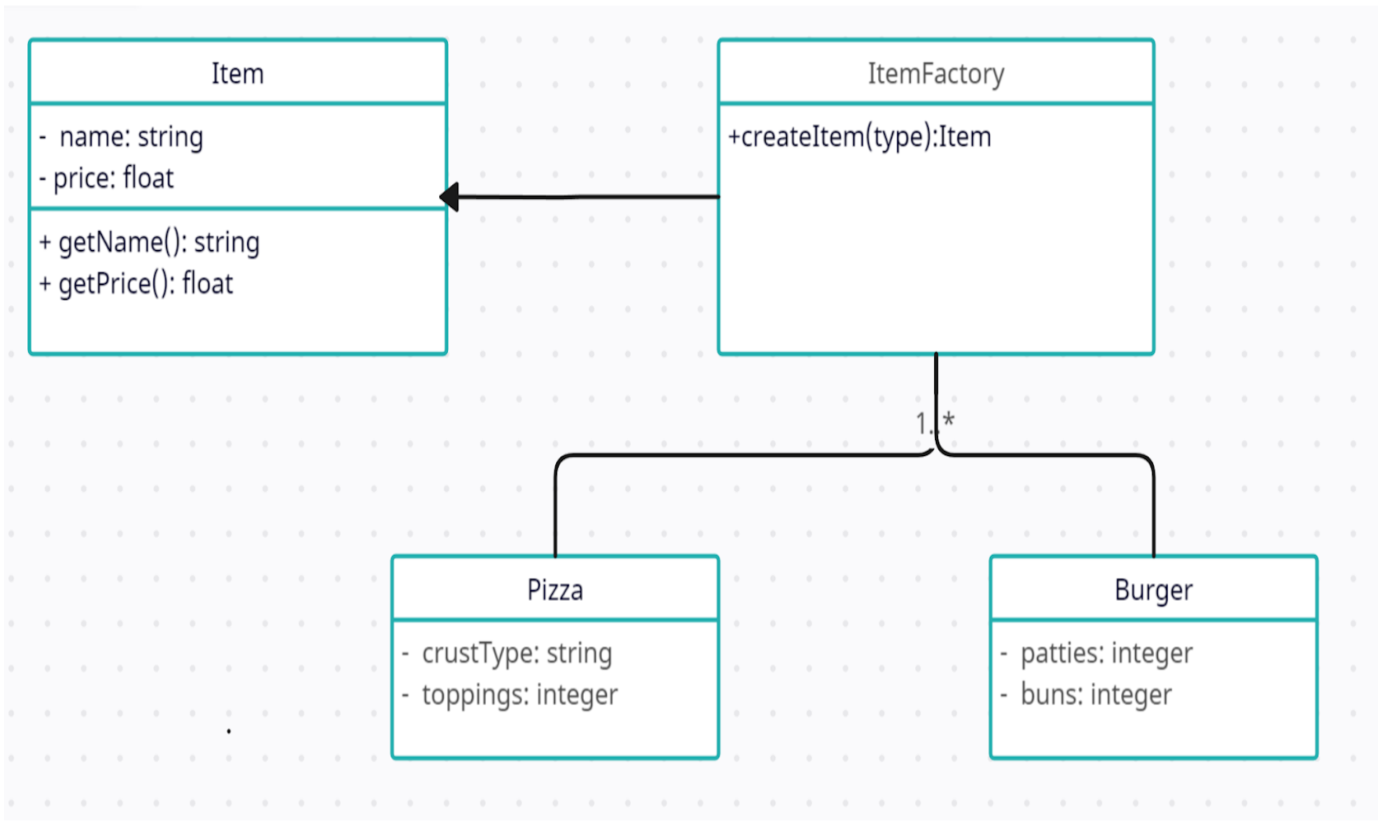
The observer design pattern is used in Go4Food concerning new changes to the order status. Parties that are affected by this should therefore be notified of such changes. For example, as each order status changes, the observers which would be the users in our case would be notified of the order status change. Below is a class diagram of this implementation of the observer design pattern within this application.


Order
- orderId: int
- status: string
+ getStatus(): string
+ setStatus(status: string): void
User subscribed to changes in Order class
User
- userId: int
+ update(order: Order): void

Order class shows a single order and it has certain properties. Some of its properties are order id and order status. The user class itself represents a single user and it has a property for user id. Note that in the final product, the user may have more properties in the database such as username. However, just to illustrate how the design pattern works, only the relevant details have been included. As visible, there is a one to many relationship between user class and order class. This means a user can make multiple orders. There is also the update method which is called by the order class when a status changes. The get status method is going to return the status of an order whereas the set status method is going to set the status for a given order.

6.2.2 Factory Design Pattern

The factory design pattern is used in Go4Food to provide a design functionality where there exist generic food items, from which there are concrete examples of a particular item. The following class diagram shows how this is achieved. For example, considering a restaurant that has food items such as pizzas and burgers, a factory design pattern can be used to create different object types of burgers and pizzas based on their unique properties, etc. For each of these, we may have different sets of toppings, crust types, etc. The factory class is responsible for creating these specific pizza instances based on user selections. Consider the class diagram below:

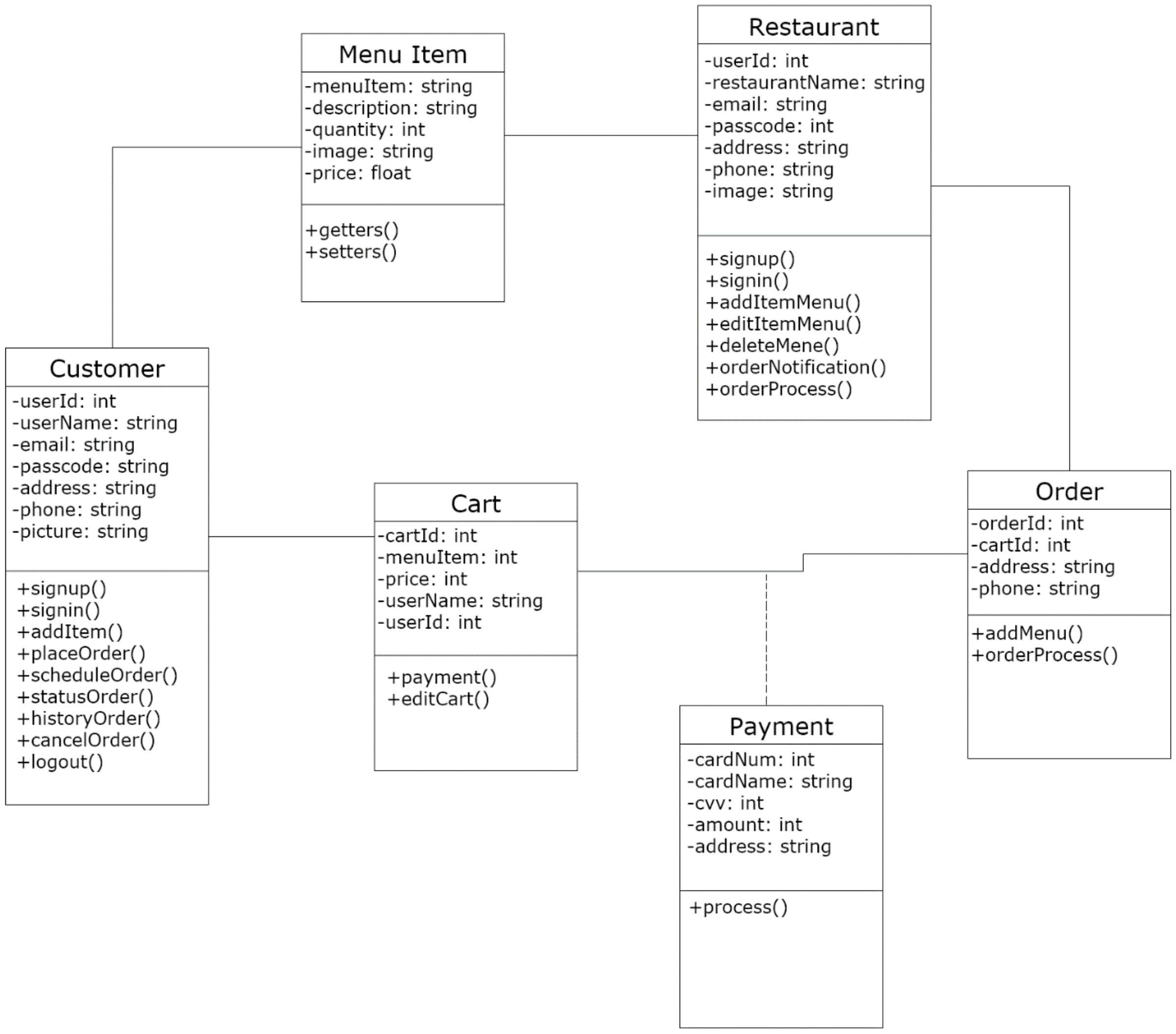


The item class is a generic class for an item that has properties for its name (string) and price (float). It also has two methods getName and getPrice which both return strings and floats respectively. The item factory class is responsible for creating particular item objects. It has a createItem(type) method which takes type as the type of item. Based on this type, it returns a concrete type of restaurant. Pizza and Burger are concrete examples of how the item class extends.

The algorithm based on the class diagram above follows that if type is pizza, an instance of the item class will be created (demonstrated by concrete class pizza). It sets this particular instance's attributes, such as name and price, and then returns its object. The same logic would apply if the type were to be something different, for example burger. Notice that the same factory design pattern could be implemented for the type of pizza itself as explained above.

The key feature here is that each pizza class or burger class inherits key properties from the generic item class, such as name and price, but with the factory design it shows how it is able to introduce new properties tailored to its own implementation.

6.3 Comprehensive Class Diagram

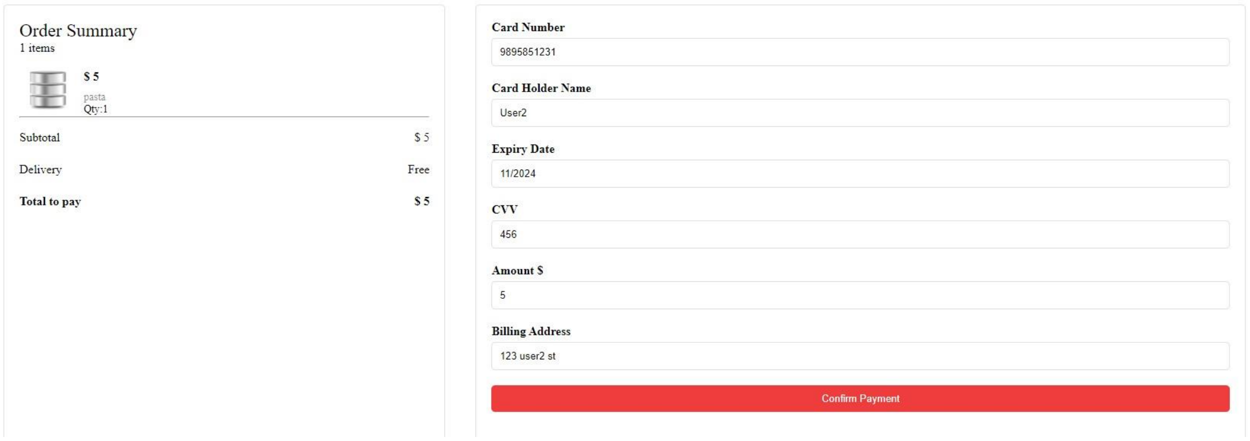


University of Regina

7. Acceptance Testing

7.1 Correctness

7.1.1 Test Case 1



Order Summary
1 items

$ 5
pasta
Qty:1

Subtotal	$ 5
Delivery	Free
Total to pay	**$ 5**

Card Number
9895851231

Card Holder Name
User2

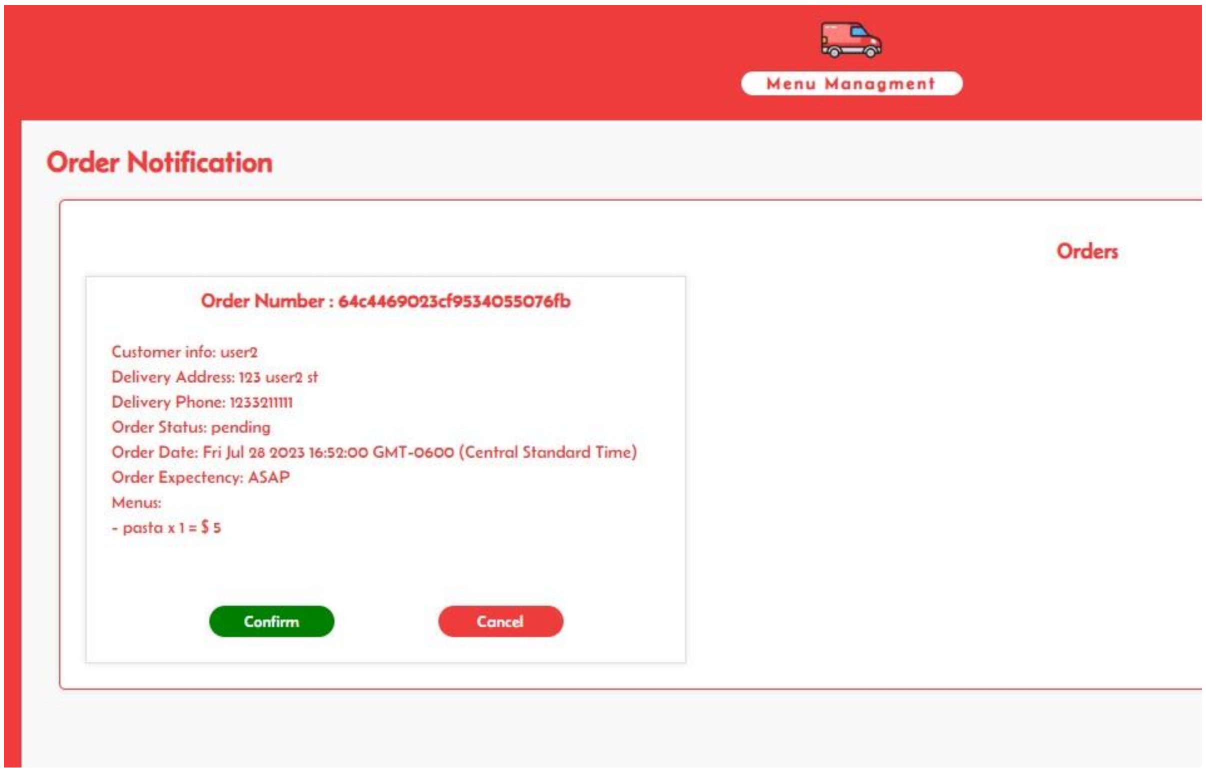
Expiry Date
11/2024

CVV
456

Amount $
5

Billing Address
123 user2 st

Customer input: order payment.

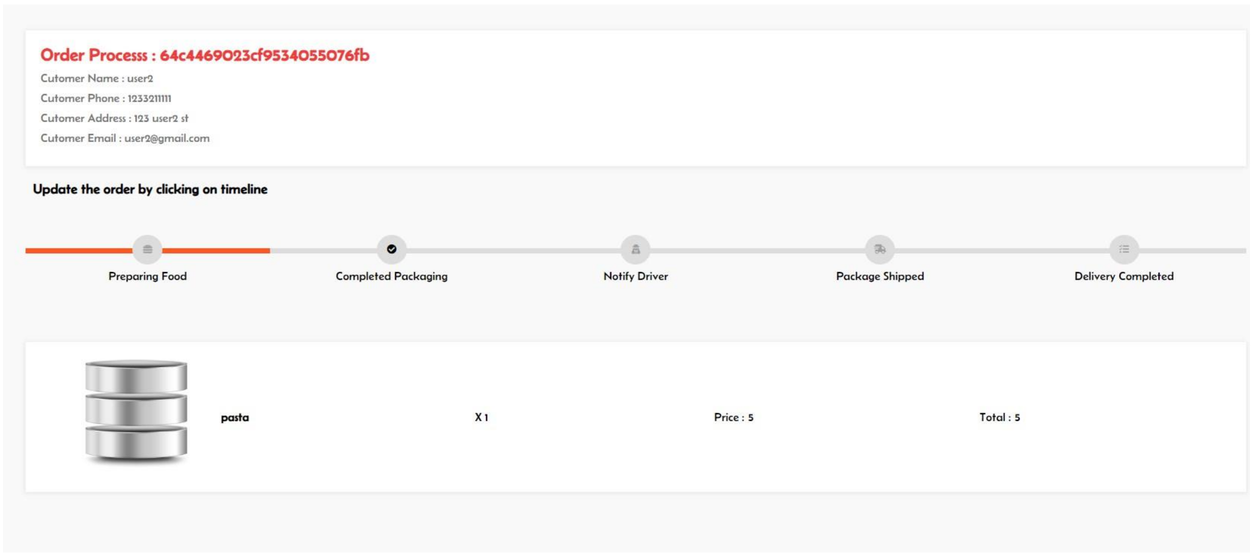


Order Number : 64c4469023cf9534055076fb

Customer info: user2
Delivery Address: 123 user2 st
Delivery Phone: 1233211111
Order Status: pending
Order Date: Fri Jul 28 2023 16:52:00 GMT-0600 (Central Standard Time)
Order Expectency: ASAP
Menus:
- pasta x 1 = $ 5

Restaurant output: order notification.

7.1.2 Test Case 2



Order Processs : 64c4469023cf9534055076fb

Cutomer Name : user2

Cutomer Phone : 1233211111

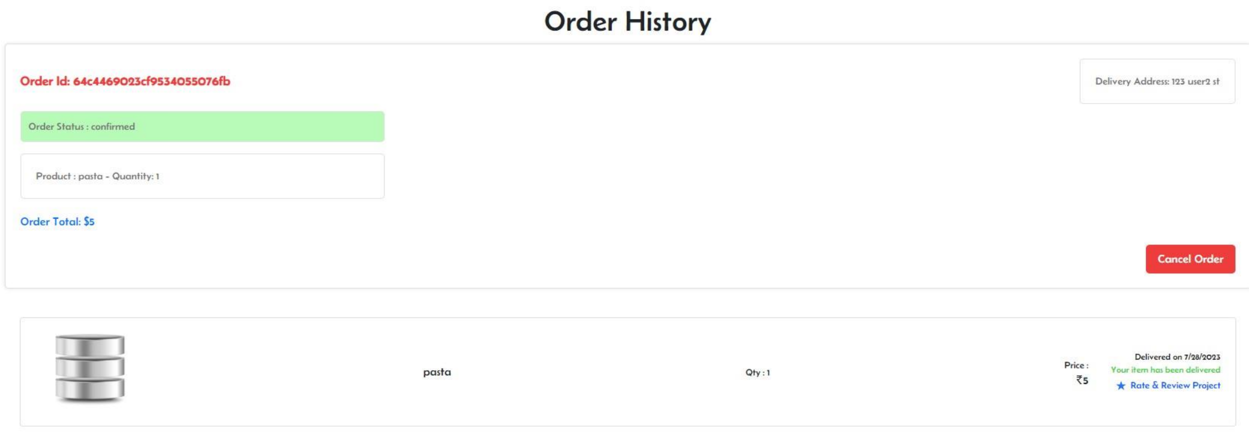
Cutomer Address : 123 user2 st

Cutomer Email : user2@gmail.com

Update the order by clicking on timeline

pasta X 1 Price : 5 Total : 5

Restaurant input: order process.



Order History

Order Id: 64c4469023cf9534055076fb

Delivery Address: 123 user2 st

Order Status : confirmed

Product : pasta - Quantity: 1

Order Total: $5

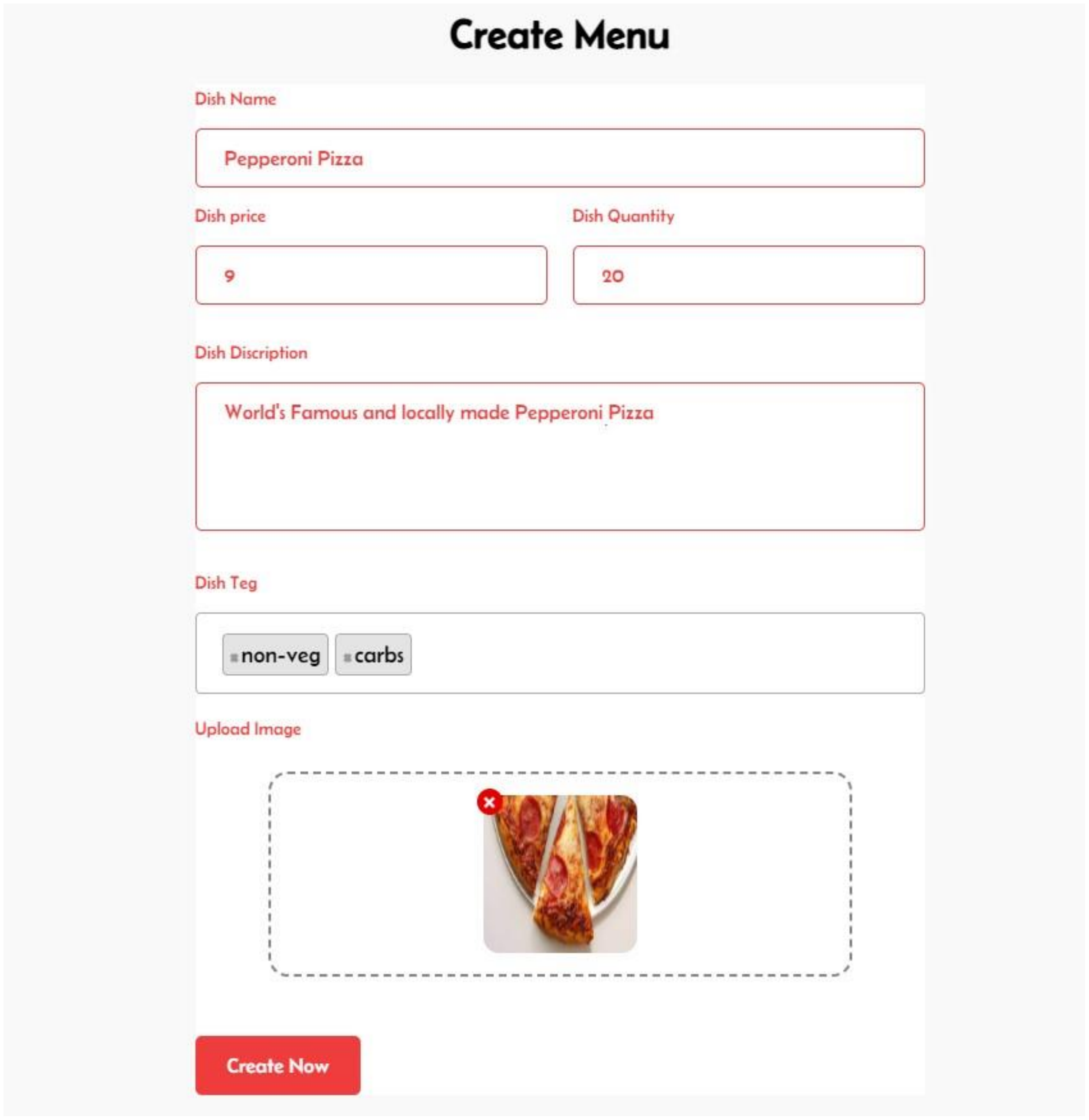
pasta Qty : 1 Price : ₹5

Delivered on 7/28/2023

Your item has been delivered

Customer output: order tracking process.

7.1.3 Test Case 3



Create Menu

Dish Name: Pepperoni Pizza

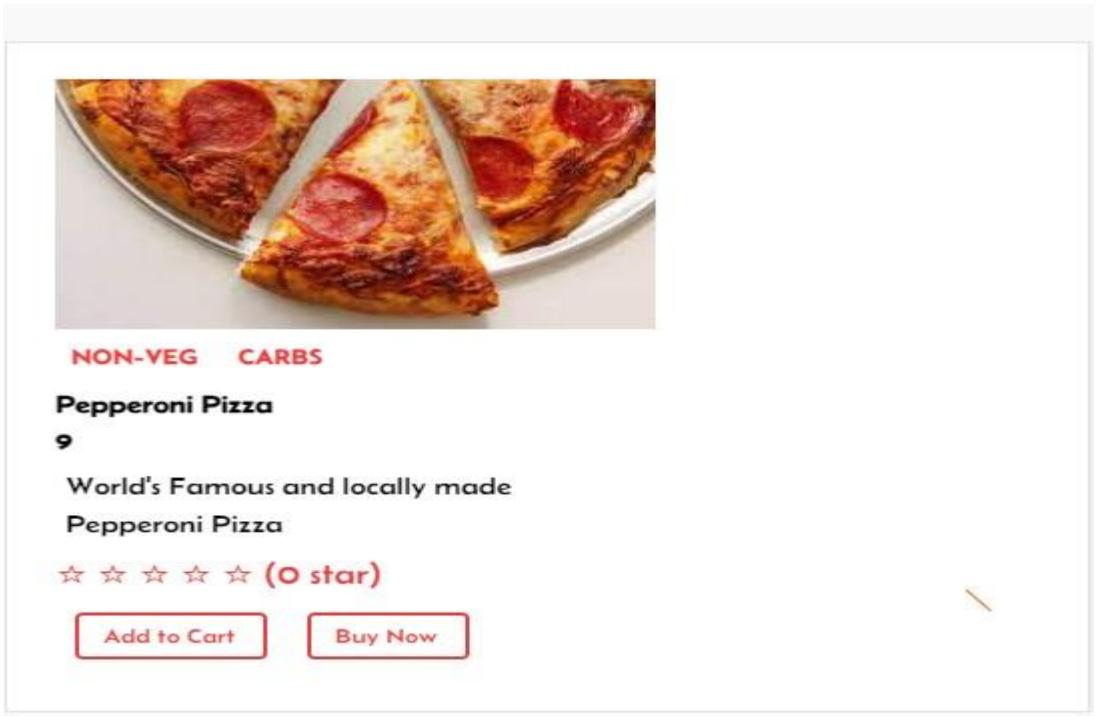
Dish price: 9

Dish Quantity: 20

Dish Discription: World's Famous and locally made Pepperoni Pizza

Dish Teg: non-veg, carbs

Restaurant input: menu creation.


NON-VEG CARBS

Pepperoni Pizza

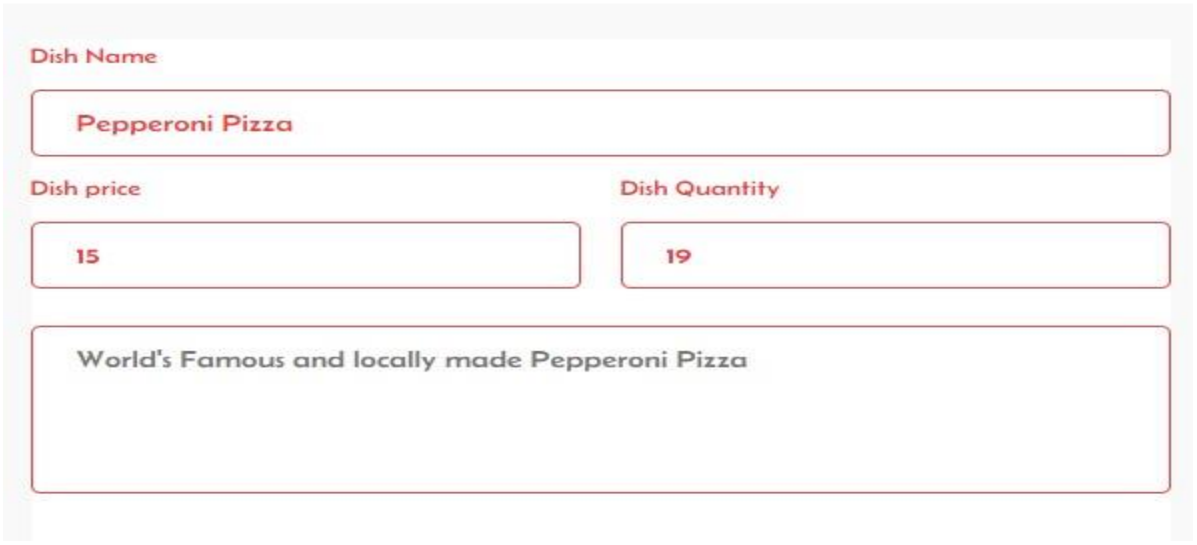
9

World's Famous and locally made Pepperoni Pizza

☆ ☆ ☆ ☆ ☆ (0 star)

Customer output: item menu.

7.1.4 Test Case 4

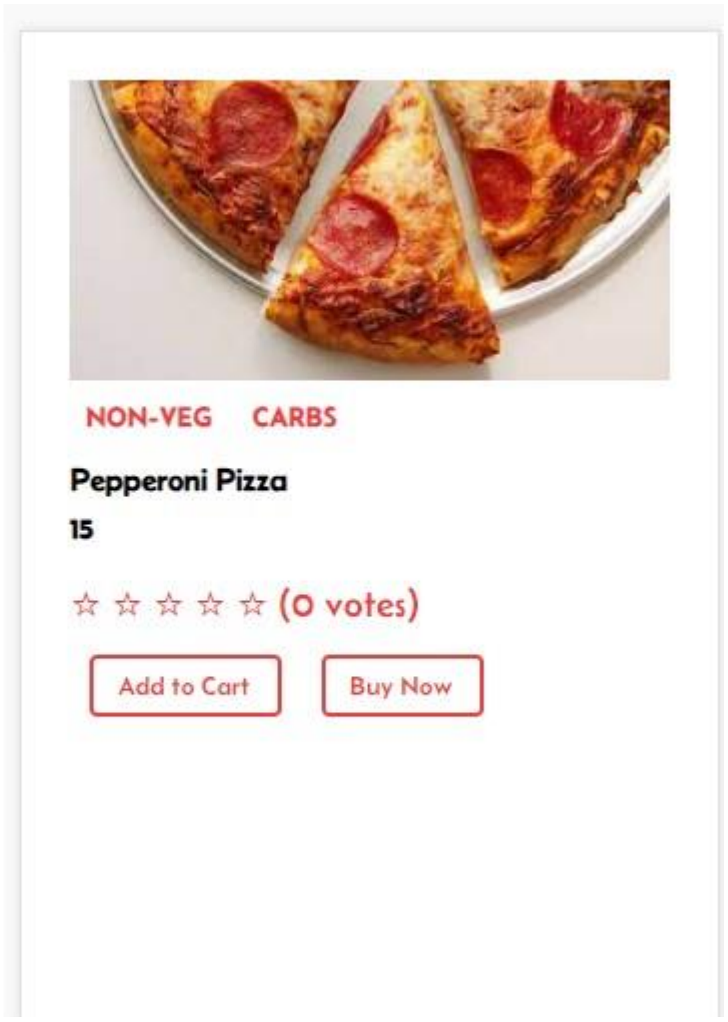

Dish Name: Pepperoni Pizza

Dish price: 15

Dish Quantity: 19

World's Famous and locally made Pepperoni Pizza

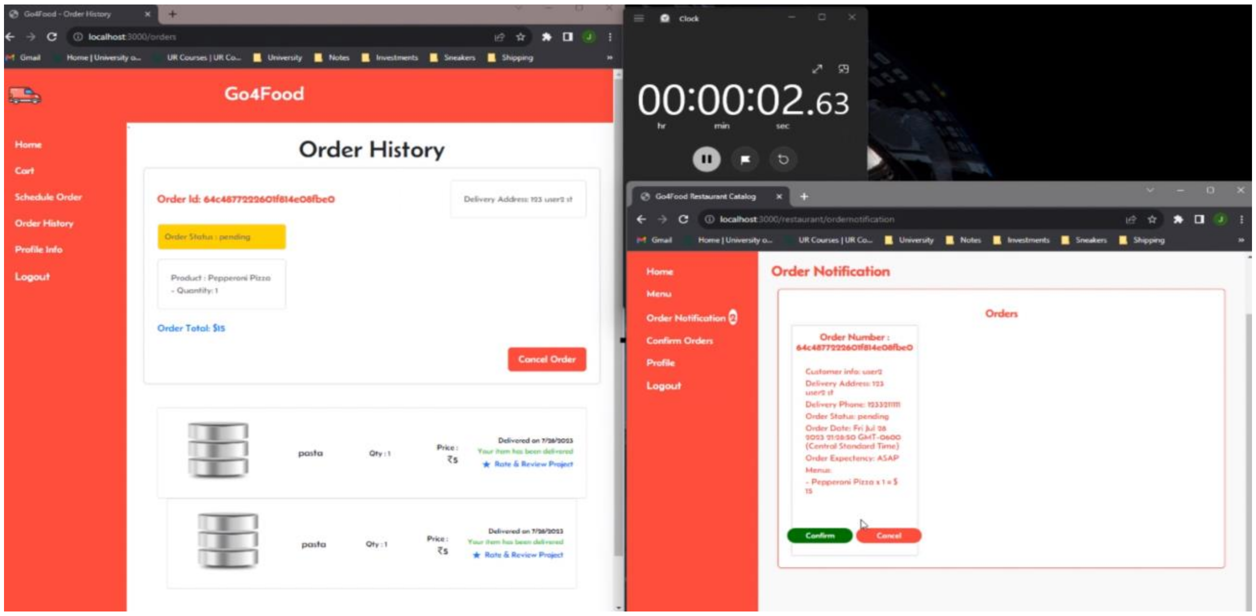
Restaurant input: menu update.



Customer output: updated menu item.

7.2 Time-efficiency

7.2.1 Test Case 1



Order History

Order Id: 64c4877222601f814e08fbe0

Delivery Address: 123 user2 st

Order Status : pending

Product : Pepperoni Pizza - Quantity: 1

Order Total: $15

pasta	Qty : 1	Price : ₹5	Delivered on 7/28/2023 Your item has been delivered ★ Rate & Review Project
pasta	Qty : 1	Price : ₹5	Delivered on 7/28/2023 Your item has been delivered ★ Rate & Review Project

Order Notification

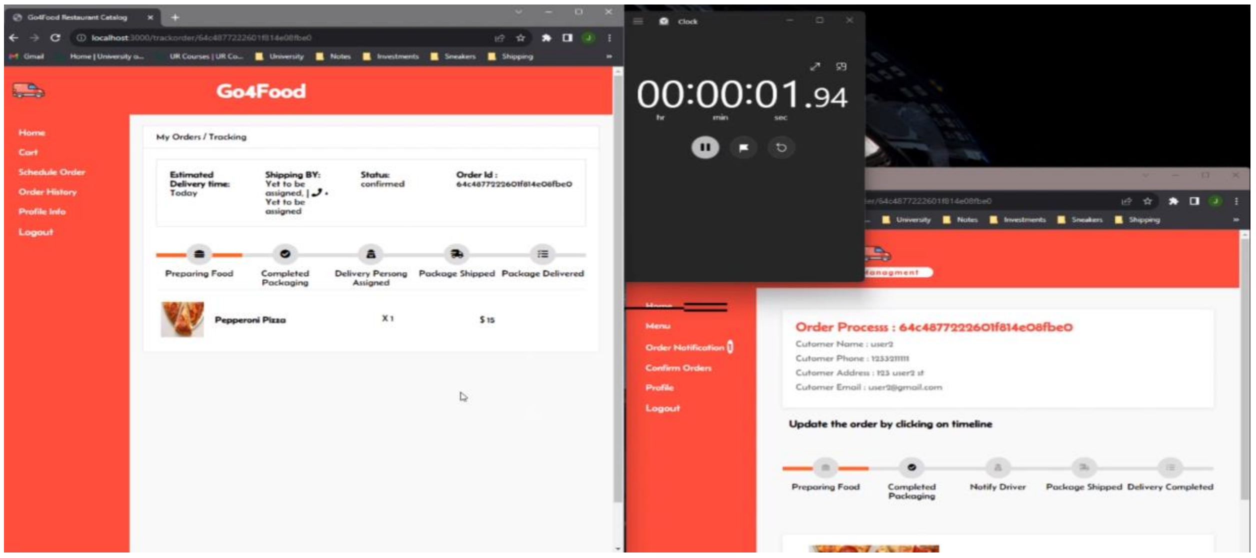
Orders

Order Number : 64c4877222601f814e08fbe0

Customer info: user2
Delivery Address: 123 user2 st
Delivery Phone: 1233211111
Order Status: pending
Order Date: Fri Jul 28 2023 21:28:50 GMT-0600 (Central Standard Time)
Order Expectency: ASAP
Menus:
- Pepperoni Pizza x 1 = $ 15

Time-efficiency test customer for order and order notification on restaurant side.

7.2.2 Test Case 2



My Orders / Tracking

Estimated Delivery time: Today	Shipping BY: Yet to be assigned, | Yet to be assigned	Status: confirmed	Order Id : 64c4877222601f814e08fbe0

Preparing Food — Completed Packaging — Delivery Persong Assigned — Package Shipped — Package Delivered

Pepperoni Pizza	X 1	$ 15

Order Processs : 64c4877222601f814e08fbe0

Cutomer Name : user2
Cutomer Phone : 1233211111
Cutomer Address : 123 user2 st
Cutomer Email : user2@gmail.com

Update the order by clicking on timeline

Preparing Food — Completed Packaging — Notify Driver — Package Shipped — Delivery Completed

Time-efficiency test for the restaurants order process and order tracking for customer side.

8. Project Members

Name	SID	Email	GITID
Adnan Uddin Mohammed	200463605	ami949@uregina.ca	adnan1420
Justine Jhon Papeleras	200405197	justinejpapeleras@gmail.com	jmp808
Mohammad Tamanna	200376962	mohammad.tamanna@outlook.com	tamannamo

8.1 Roles and Responsibilities

Name	Roles	Responsibilities
Adnan Uddin Mohammed	• Full Stack • Requirements Documentation	• Modified the server and front end to fit with the backend • Created the backend for both Customer and Restaurant users • Presentation PPT Design patterns
Justine Jhon Papeleras	• Project Manager • Full Stack • Requirements Documentation	• Implemented the server • Static pages for index, sign in and registration pages • Creation of diagrams, charts, and tables • GitHub organizations • UI designs and flow • Presentation PPT and Project Report • In charge of tasks managements and allocations
Mohammad Tamanna	• Project Manager • Front End • Requirements Documentation	• In charge of tasks managements and allocations • Created static pages for the customer • Presentation PPT and Project Report

8.2 Project Reporting

Project reporting structure was completed through the GitHub Projects section, arranged and organize resembling a functioning Kanban Board. The Project section is a form of a task-board spreadsheet and road map. It aided through visualization in showcasing and coordinating the group's strategic planning, task assignment and completion, and the project's overall efficiency. Group members meets twice a week, after the class lectures and every Fridays afternoons. The after class meeting is a quick touch based meeting where tasks were assigned, status reporting, and general questions/concerns. Friday meetings were a group work meeting where team members collaborate and work together on their tasks. This projects reporting structure approach falls under the Agile Scrum Methodology.

9. Development Environment

9.1 Software Tools

The software tools that were used to complete the project are VS Code, Adobe XD, WebStorm, GitHub, Express.js, mongoDB, and moongose.

Server Environment: OS: Ubuntu, Architecture: 64-bit. CPU 1 Virtual CPU. RAM:512mb. Storage: 10 GiB

Node Packages: + bcrypt 5.1.0 + body-parser 1.20.2 + connect-mongodb-session 3.1.1 + dotenv 16.3.1 + ejs 3.1.9 + express 4.18.2 + express-session 1.17.3 + mongoose 7.3.4 + multer 1.4.5-lts.1 + nodemon 3.0.1 + passport 0.6.0 + passport-local-mongoose 8.0.0 + start 5.1.0

9.2 Programming languages

Project’s application programming languages were HTML, CSS, JavaScript, N Node.js.

9.3 Project folders

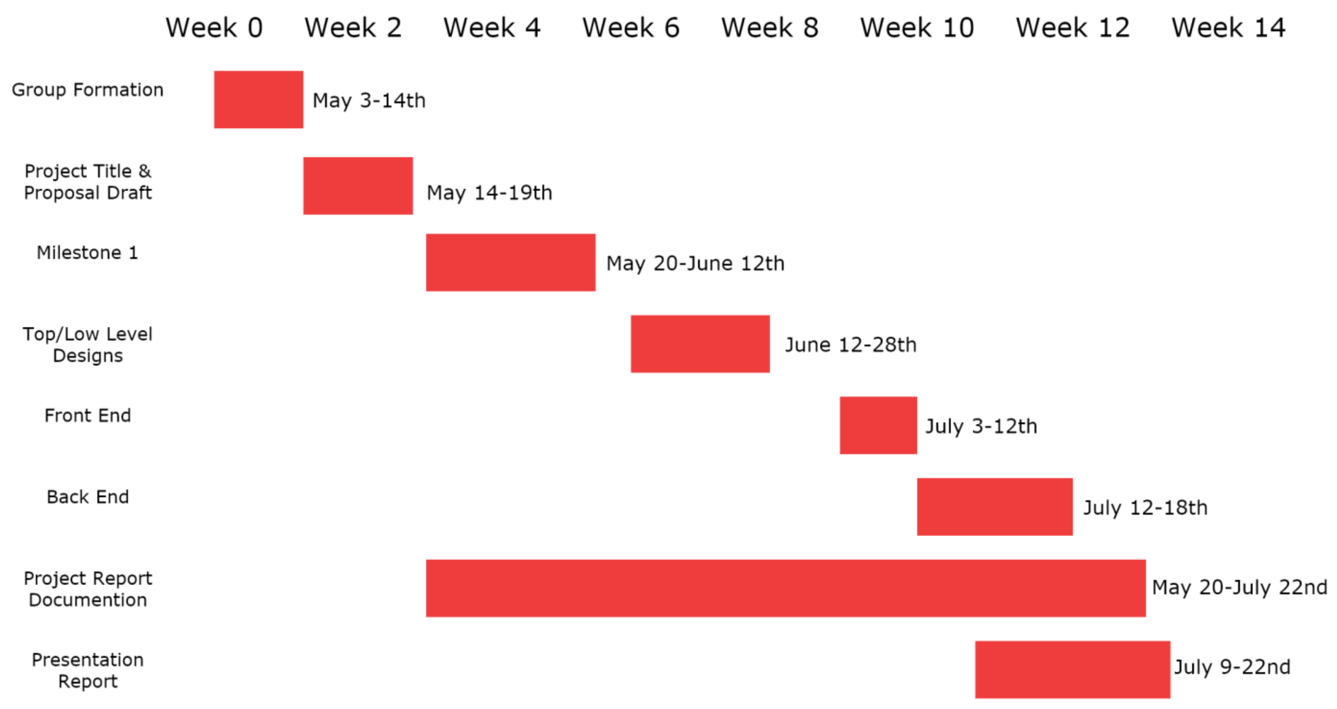
The folders in the project roots in the GitHub are the Documents, config, controllers, middleware, models, public, utils, and views. The Documents folder contains the key project documentations. The config folders contains the database JavaScript. The controllers, models, and views are folders that follows and contains the respective MVC architectural patterns. Middleware folders contains user authorizations for both the Customer and Restaurants users.

10. Project Timeline

10.1 Project Milestones

Milestones	Dates
Group Formation	14th May 2023
Group’s Project Proposal	15th May 2023
Project Title and Proposal Draft	19th May 2023
Milestone 1	12th June 2023
Hi-Fi UI and UX Design Completion	28th June 2023
Top-Level and Low-Level Software Design	28th June 2023
Front End Completion	12th July 2023
Back End Completion	18th July 2023
Project Report Completion	22nd July 2023
Presentation Report Completion	22nd July 2023
Project Deadline	21st July 2023
Presentation Opens	26th July 2023
Presentation	TBA

10.2 Gantt Chart



11. Expected Outcome and Products

11.1 Deliverables

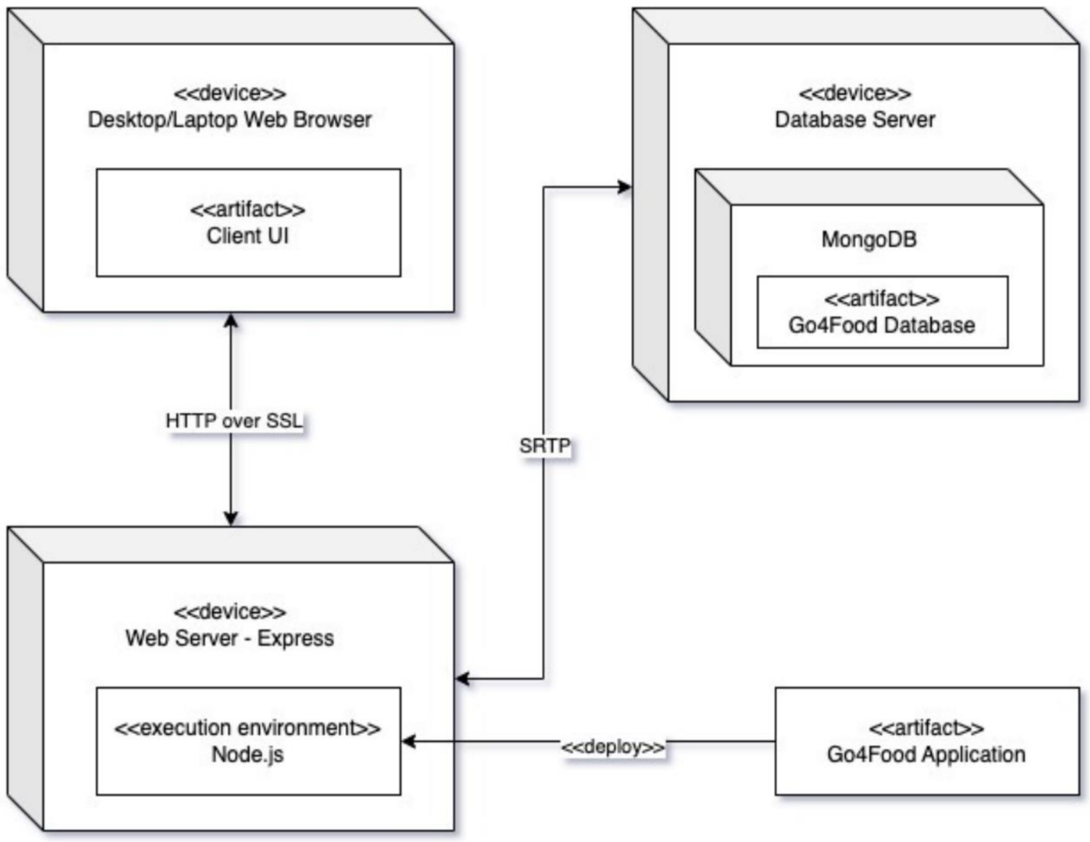
The project deliverables are the programs code, required diagrams, project proposals, the final report, presentation PowerPoints, and lastly an online web application server for our final product. These deliverables are also available to view on our GitHub project repository: https://github.com/jmp808/Go4Food. Link to Web: http://go4food.xyz/

11.2 Risks and Constraints

Due to the Spring/Summer semester the time constraints of roughly three months for the projects formation and proposal, to designing phase, coding of the application, implementing and testing, and lastly the final Project Reports and presentation were significant. The three person group instead of four member means that extra tasks for each team member. In addition the Scope constraint, if the scope were too big there is a risk that the project will miss some requirements. The constraints explained above lead us to unable to develop a few features, such as customized orders.

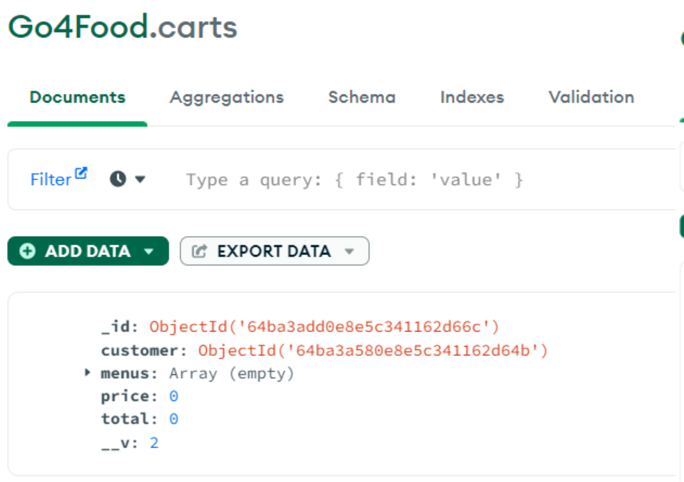
The risks of the development project are group members not being able to complete given tasks with due dates leads to delaying other timed planned tasks. Project scope being to extensive and labour consuming given the limited time constraint. Lastly technical risks, it was our group’s first major project like this and using different new types of technologies could lead to technical/design/integration and/or implementation delays

12. Deployment Diagram

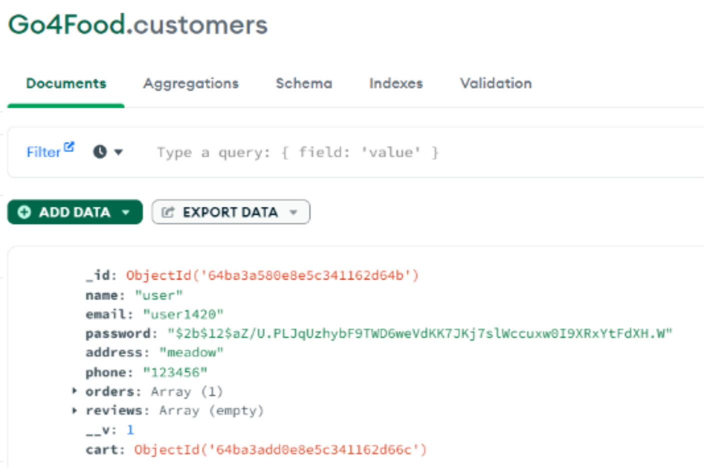


13. Table Contents of the System Data

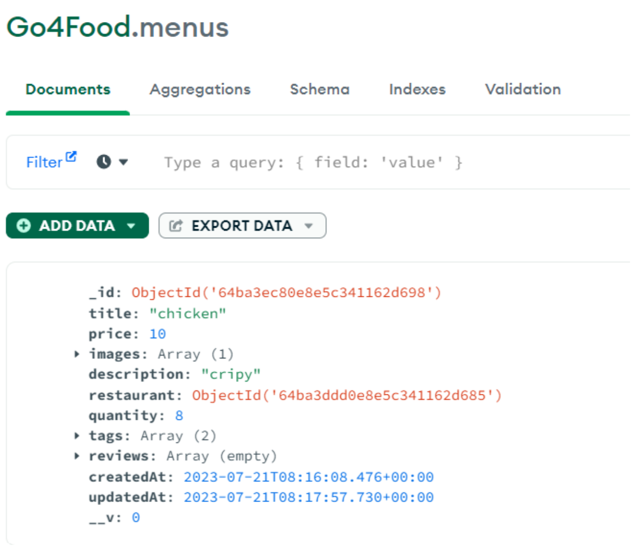
We used MongoDB which is a NoSQL database. Unlike traditional relational database system we don't have table and schema. So we are submitting the screenshots of the collection and one document.



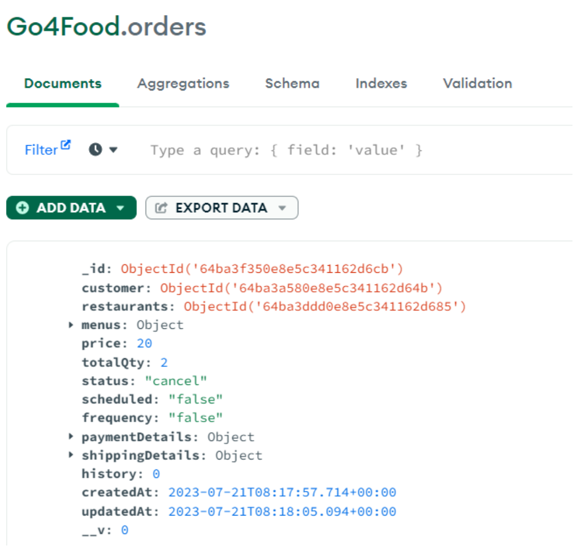
```
_id: ObjectId('64ba3add0e8e5c341162d66c')
customer: ObjectId('64ba3a580e8e5c341162d64b')
menus: Array (empty)
price: 0
total: 0
__v: 2
```



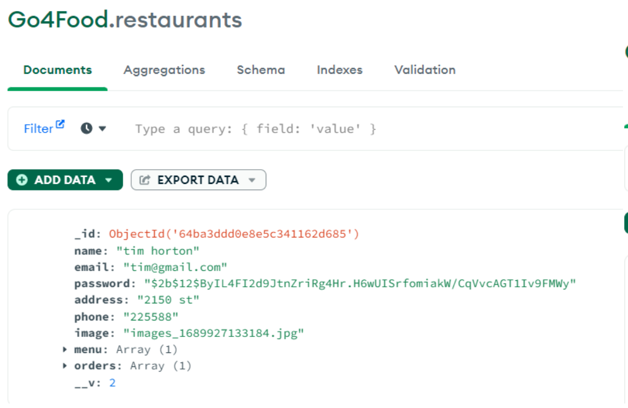
```
_id: ObjectId('64ba3a580e8e5c341162d64b')
name: "user"
email: "user1420"
password: "$2b$12$aZ/U.PLJqUzhybF9TWD6weVdKK7JKj7slWccuxw0I9XRxYtFdXH.W"
address: "meadow"
phone: "123456"
orders: Array (1)
reviews: Array (empty)
__v: 1
cart: ObjectId('64ba3add0e8e5c341162d66c')
```



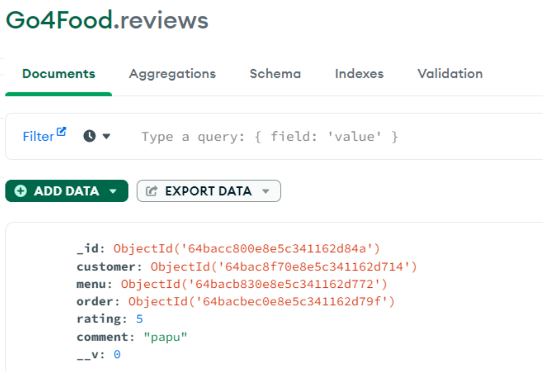
```
_id: ObjectId('64ba3ec80e8e5c341162d698')
title: "chicken"
price: 10
images: Array (1)
description: "cripy"
restaurant: ObjectId('64ba3ddd0e8e5c341162d685')
quantity: 8
tags: Array (2)
reviews: Array (empty)
createdAt: 2023-07-21T08:16:08.476+00:00
updatedAt: 2023-07-21T08:17:57.730+00:00
__v: 0
```



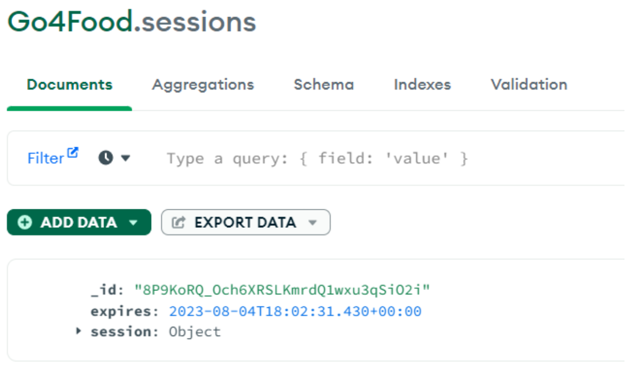
```
_id: ObjectId('64ba3f350e8e5c341162d6cb')
customer: ObjectId('64ba3a580e8e5c341162d64b')
restaurants: ObjectId('64ba3ddd0e8e5c341162d685')
menus: Object
price: 20
totalQty: 2
status: "cancel"
scheduled: "false"
frequency: "false"
paymentDetails: Object
shippingDetails: Object
history: 0
createdAt: 2023-07-21T08:17:57.714+00:00
updatedAt: 2023-07-21T08:18:05.094+00:00
__v: 0
```



```
  _id: ObjectId('64ba3ddd0e8e5c341162d685')
  name: "tim horton"
  email: "tim@gmail.com"
  password: "$2b$12$ByIL4FI2d9JtnZriRg4Hr.H6wUISrfomiakW/CqVvcAGT1Iv9FMWy"
  address: "2150 st"
  phone: "225588"
  image: "images_1689927133184.jpg"
▸ menu: Array (1)
▸ orders: Array (1)
  __v: 2
```



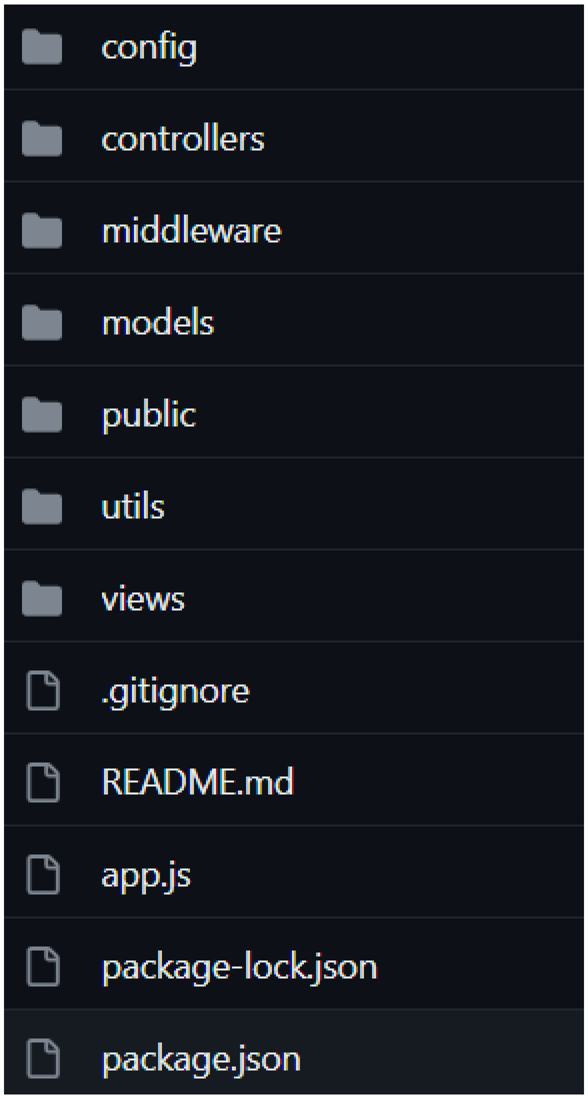
```
_id: ObjectId('64bacc800e8e5c341162d84a')
customer: ObjectId('64bac8f70e8e5c341162d714')
menu: ObjectId('64bacb830e8e5c341162d772')
order: ObjectId('64bacbec0e8e5c341162d79f')
rating: 5
comment: "papu"
__v: 0
```



```
  _id: "8P9KoRQ_Och6XRSLKmrdQ1wxu3qSiO2i"
  expires: 2023-08-04T18:02:31.430+00:00
▸ session: Object
```

Entire Source Code for the Framework

Config: This folder contains all the configurations files such as database configurations.
Controllers: This folder contains all the controllers and the associated routes. Routes files have _r.js suffix.
Middleware: This directory contains application middleware. These middleware are used to validate customer and authentication.
Models: This folder consists of all the models of the application. We used Mongoose ORM.
Public: This directory contains all the CSS and JavaScript assets.
Uploads: All the uploaded images are stored in this directory.
Utils: It contains all the helper functions. views: This directory contains all the views of the application.
app.js: It is the entry point of the application.
package.json / package-lock.json: It contains the packages required to run the application.



```
config
controllers
middleware
models
public
utils
views
.gitignore
README.md
app.js
package-lock.json
package.json
```