SW Engineering CSC648/848
Application Name: PizzaCraze
Section 04, Team 07

Monisha Mekala - Team Leader/co-Back-end Lead
Aishwarya Magar - Front-end Lead
Malieka Sutaria - Scrum Master
Nicholas Pagcanlungan - co-Back-end Lead
Joey Palanca - GitHub Master

Milestone 2

October 11, 2023

Revision History

Document Version	Date	Changes
1.0	27th Sep, 2023	Initial Submission
2.0	11th Oct, 2023	Milestone 2 submission

1. Data Definitions V2

Primary Data Name	Definition with examples	Usage
User	A person who is registered on the platform e.g., “Rishi Verma” rishiverma@gmail.com, +1 628 246 ***	Information about registered users is collected for order history, address and to get feedback.
Menu_Item	Represents a specific food item available for order, e.g., "Margherita Pizza".	This list of food items will be displayed for users to choose from when placing an order.
Address	Storing address information involves recording the location details of users, which can include address lines, zip codes, and additional information like apartment or suite numbers.	Accurate address information ensures orders reach the correct location and is vital for various operational aspects.
Category	A classification that groups related food items, e.g., "Pizza", "Pasta", "Dessert".	Users can filter and view food items by their respective categories.
User_Preference	This feature enables users to choose the spice level, cheese level and quantity of additional toppings they want on their food item. It offers options like mild, medium, and extra.	Different people have varying preferences for spiciness. Some may prefer milder flavors, while others enjoy a more intense kick. By offering spice levels, you're catering to a wider range of tastes.
Cart_Item	An item added to a user's shopping cart.	Temporarily stores selected food items for review before finalizing an order.
Order	Represents a specific food order placed by a user.	Keeps track of individual orders, including items, quantities, total price, and order status (e.g., pending, completed).

Feedback	Feedback provided by a user regarding a specific order, e.g., "Great pizza, fast delivery!".	Helps improve service and provides information for other users considering the same food item.
Share_Order	The Share Order Poster feature allows users to share a visually appealing representation of their order on their social media platforms after the order has been successfully delivered. This poster includes details such as the items ordered, any special customizations, the restaurant's branding, and a message from the user.	This feature enables users to celebrate their food ordering experiences by sharing them with their social network, creating a sense of enjoyment and community around food. Shared order posters act as organic advertising, increasing the visibility of the platform on various social media channels.
Admin	Storing admin details include username and password.	The Admin table stores login credentials for administrators. This allows authorized personnel to access and manage the system, including overseeing orders, menu items, user accounts, and other administrative tasks essential for the smooth operation of the food ordering platform.

Primary Data Name	Sub-Data
AdminDetails	username password
LoginDetails	user_id email password phone date_created
MenuTable	menu_id name category

	description price calories ingredients is_veg is_vegan is_nonveg image_url
AddressTable	address_id user_id address_line1 address_line2 apt_suite zipcode address_added_on
CartTable	cart_id user_id
OrderTable	order_id user_id total_price address_id status order_date
FeedBack	feed_id user_id order_id feedback rating feedback_date

2. Functional Requirements V2

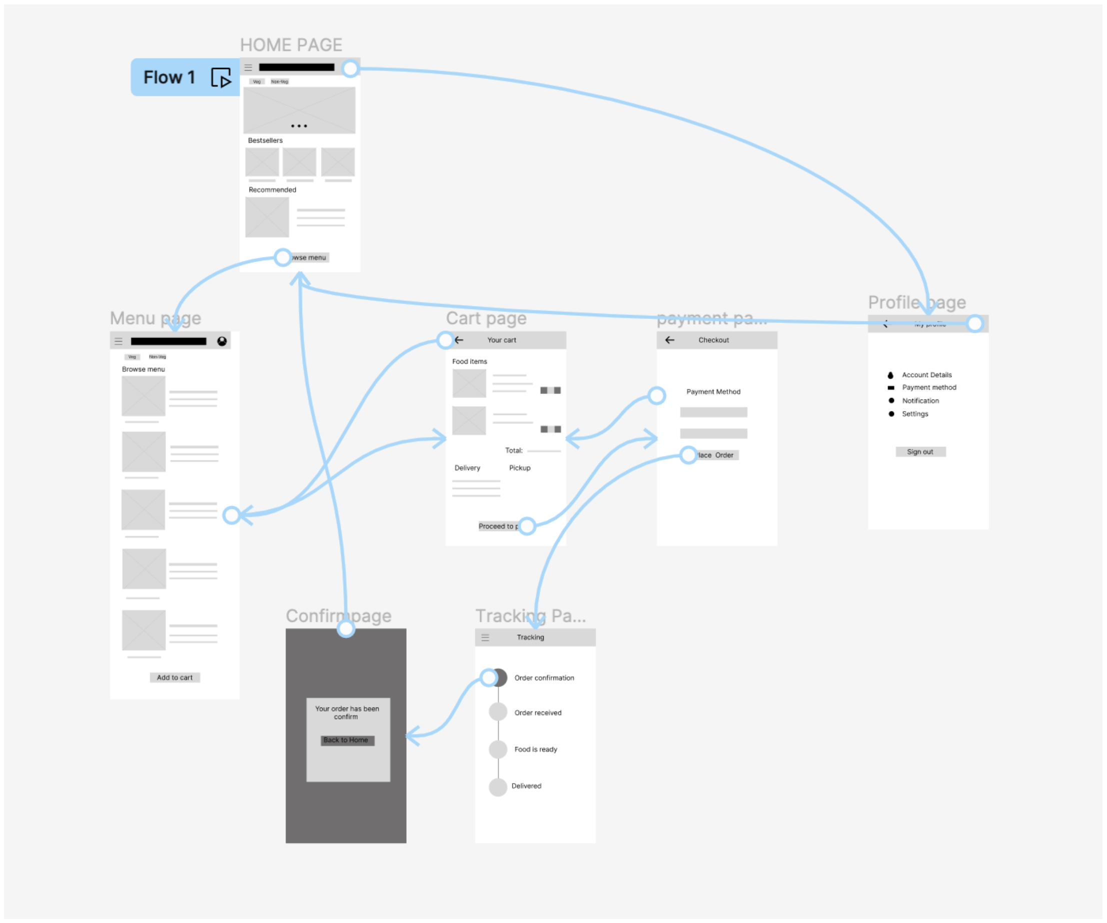
🟦	Must-have	🟥	Desired	🟩	Opportunistic

	ID	Functional Requirement Description	Details (As Needed)
🟦	1	Users can Sign up and register for their own accounts	1.1) User can sign up for a new account
🟦	2	Users can Log in to the system once they have registered.	2.1) They can place an order only once they have logged in to the system.
🟦	3	Users should be able to browse the menu.	3.1) Users should be able to check out all the food items in the menu list by browsing the menu page. 3.2) The menu page should have all the available items. 3.3) Every item should have a name, a short description, and a picture.
🟦	4	Users should be able to add items to the cart.	4.1) Users should be able to add the food items they want to the cart.

🟦			4.2) They should also be able to remove the food items he no longer wants in the cart.
🟦	5	Users should be able to specify the quantity of food items.	5.1) Users should be able to increase or decrease the quantity of the food items they want to add to the cart.
🟦	6	Users should be able to place an order.	6.1) Users should be able to place an order for all the items in the cart.
🟥	7	Users should be able to filter the items.	7.1) Users should be able to filter the food items they want to see in the menu list. 7.2) Users should be able to filter the items based on the category and name.
🟥	8	Users should be able to customize the food items.	8.1) Users should be able to specify additional changes that they want in their food 8 .2) Users should be able to select the level of spice they want and the toppings of their choice.

🟥	9	Users should be able to share their order on their social media.	9.1) Users should be able to share the food they ordered on their social media accounts.
🟥	10	Users should be able to give feedback on their order.	10.1) Users should be able to contact the admins to give any feedback they want for any specific order.
🟩	11	Users should be able to track their delivery status.	11.1) Users should be able to track their order. 11.2) While the order is in transit, the employees will be able to update the status of delivery.

3. UI Mockups and UX Flows



4. High level Architecture, Database Organization

DB organization:

Customers

LoginDetails	
user_id	int → pk
email	varchar(100)
password	varchar(45)
phone	varchar(45)
date_created	timestamp

AddressBook	
address_id	int → pk
user_id	int → fk → LoginDetails.user_id
address_line1	text
address_line2	text
apt_suite	text
zipcode	int
address_added_on	timestamp

Admins

AdminDetails	
username	varchar(100) → pk
password	varchar(45)

Menu

MenuTable	
menu_id	int → pk
name	varchar(255)
category	enum('Pizza','Pasta','Dessert')
description	text
price	decimal(8,2)
calories	int

MenuTable	
ingredients	text
is_veg	tinyint(1)
is_vegan	tinyint(1)
is_nonveg	tinyint(1)
image_url	text

Orders

OrderTable	
order_id	int → pk
user_id	int → fk → LoginDetails.user_id
total_price	decimal(8,2)
address_id	int fk → AddressBook.address_id
status	varchar(40)
order_date	timestamp

OrderItemTable	
order_item_id	int → pk
order_id	int → fk → OrderTable.order_id
menu_id	int → fk → MenuTable.menu_id
quantity	int
cheese_level	int
spice_level	int
quantity_toppings	int

Cart

CartTable	
cart_id	int → pk
user_id	int → fk → LoginDetails.user_id

CartItemTable	
cart_item_id	int → pk
cart_id	int → fk → CartTable.cart_id

CartItemTable	
menu_id	int → fk → MenuTable.menu_id
quantity	int
cheese_level	int
spice_level	int
quantity_toppings	int

Feedback

Feedback	
feed_id	int → pk
user_id	int → fk → LoginDetails.user_id
order_id	int → fk → OrderTable.order_id
feedback	text
rating	int
feedback_date	timestamp

Add/Delete/Search architecture:

1. **Login** - Search for entries in LoginDetails
2. **Signup** - Search for duplicates in LoginDetails, then add an entry if none are found
3. **Browsing the Menu** - Display entries of MenuTable
4. **Filter Items** - Search for and display entries in Menu matching a certain filter (is_veg, is_vegan, is_nonveg)
5. **Adding Items to Cart** - Add entries to or search for and delete them from CartItemTable
6. **Customize the Food Items** - Alter properties of CartItemTable entries, such as ‘spice_level’, ‘cheese_level’, or ‘quantity_toppings’
7. **Quantity of Food Items** - Alter ‘quantity’ in CartItemTable entries

8. **Place an Order** - Search for, then delete entry from CartTable and associated entries from CartItemTable, then add them to OrderTable and OrderItemTable
9. **Tracking the Delivery Status** - Search for and display ‘status’ from OrderTable entry
10. **Share on Social Media** - Search for OrderTable entry, then display associated OrderItemTable entries
11. **Feedback Form** - Add entries to Feedback

APIs*:*
Backend Endpoint APIs:

1. **GET ‘/’**

   Description: This API serves as a basic test route to verify if the backend is operational.

   Usage: It returns a simple JSON message indicating the backend's status.

2. **GET ‘/GetMenu’**

   Description: Retrieves a list of all registered users from the database.

   Usage: This API fetches user information including their names, categories, description, price, calories, ingredients, and other relevant details.

3. **GET ‘/searchBar/:query’**

   Description: This API is used to search for menu items based on a provided query term. It performs a case-insensitive search on both the menu item names and categories.

   Usage: To use this API, make a GET request to /searchBar/:query, replacing :query with the search term you want to use. This API fetches all the menu items which have a search term (“query”) in its name or in category.

4. **POST ‘/PostUsers**

   Description: Registers a new user in the system.

   Usage: This API allows the frontend to send user registration details including their username, password, phone, and email. It then stores this information in the database.

3rd Party APIs / Components:

Axios: Axios is a popular library used for making HTTP requests from the frontend to the backend. It enables seamless communication between different parts of the application.

Twitter/X API: Twitter, or now known as X, is a popular social media platform designed for short quick messages to be posted out to those who follow you and as well as their own followers. This API enables us to make calls to their database and allows us to create posts for our pizza.

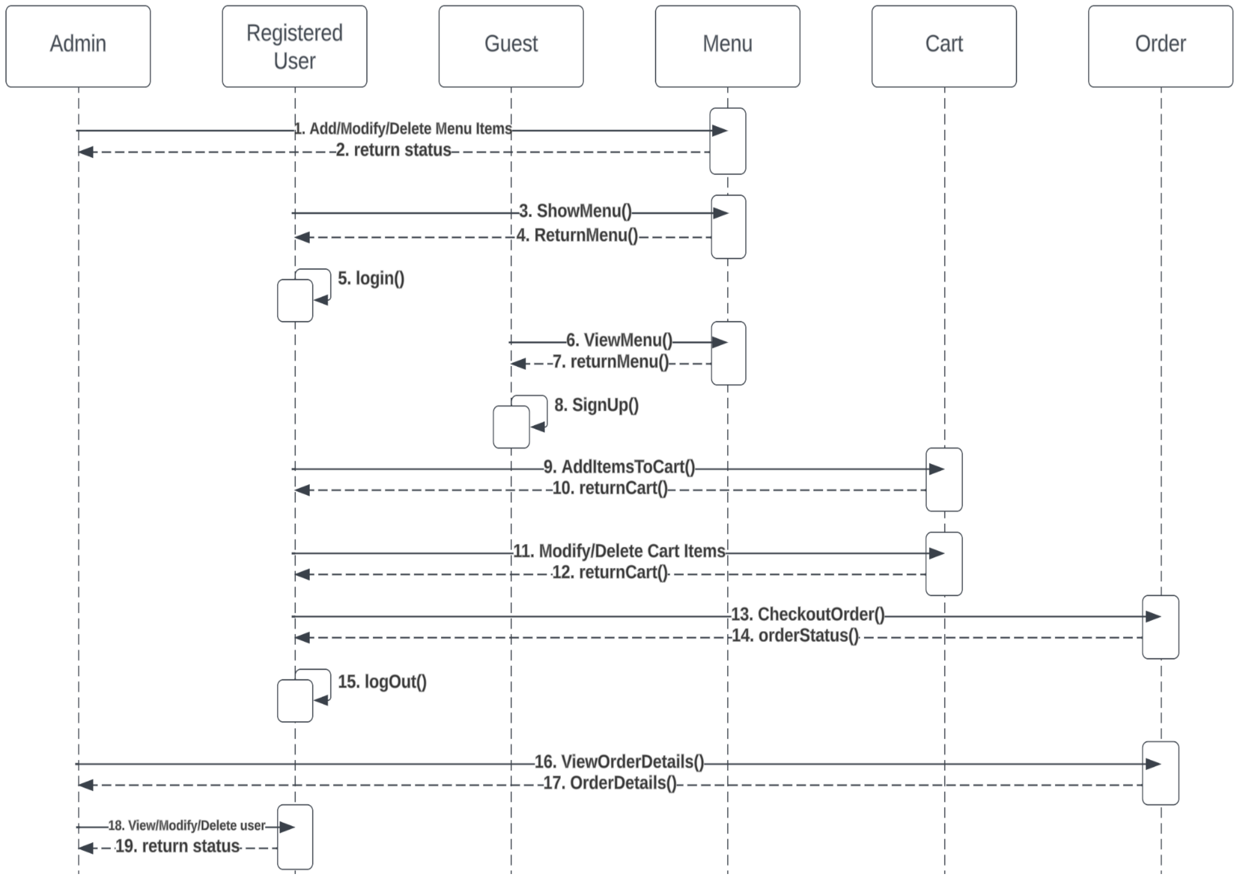
Open-Source Components:

- **Express**: Express is a fast, unopinionated, minimalist web framework for Node.js. It forms the backbone of this backend, handling routing and middleware functionality.
- **MySQL**: MySQL is an open-source relational database management system. It's used for storing and managing data related to users, menus, and orders.
- **CORS**: CORS (Cross-Origin Resource Sharing) is a security feature implemented by web browsers. It allows or blocks web applications running at one origin (domain) to make requests to a different origin.

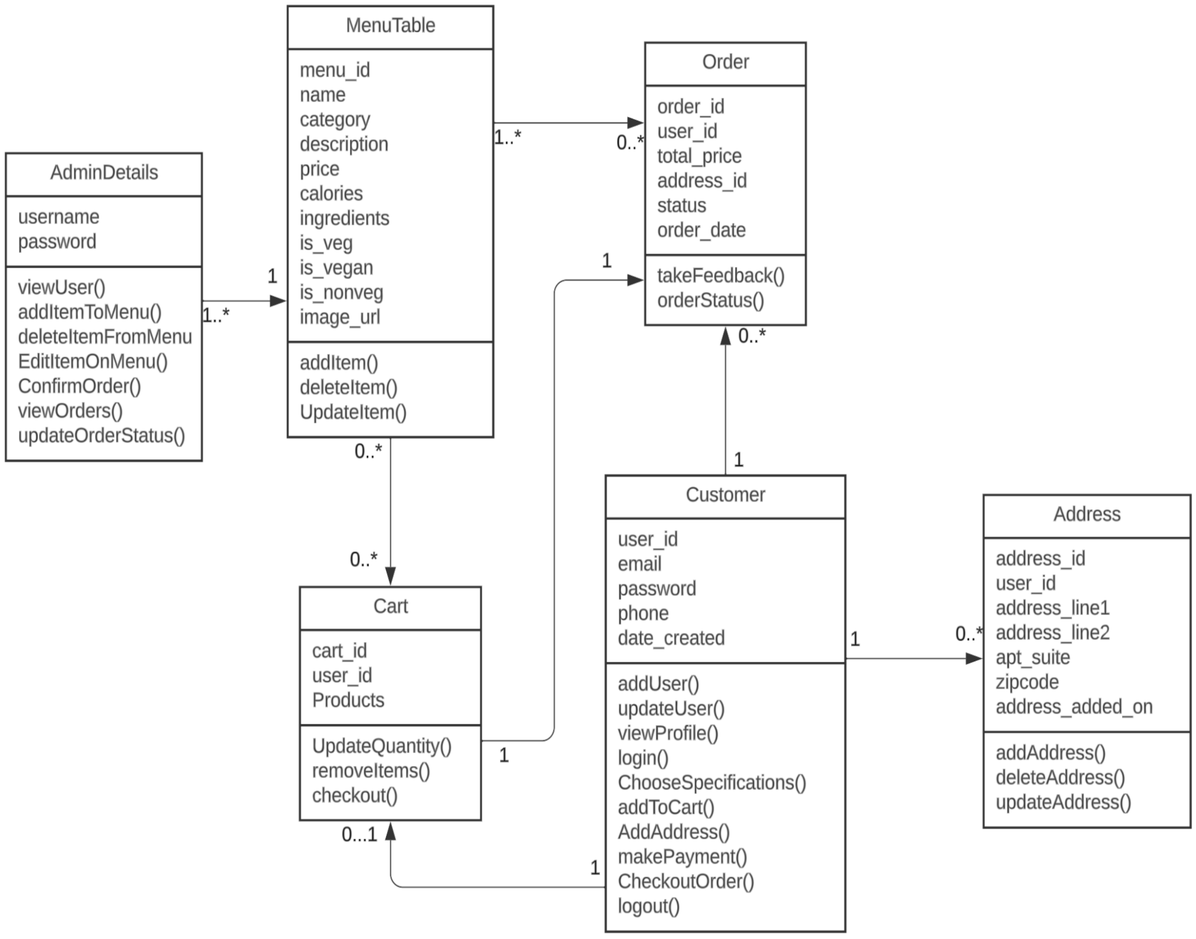
Architecture Changes/Additions:

- **CORS Middleware**: The CORS middleware has been integrated to handle cross-origin requests, enabling the frontend to interact with the backend without security restrictions.
- **Axios**: Axios has been added as an HTTP client to facilitate communication between the frontend and backend.
- **MySQL Database**: The backend interacts with a MySQL database hosted on an AWS RDS instance. This is where user information, menu data, and other application-related details are stored.
- **Database Connection Configuration**: The backend code contains the configuration settings required to establish a connection with the MySQL database, including host, user, password, and database name.

5. High Level UML Diagrams



High-level sequence diagrams:



6. Identify actual key risks for your project at this time

- **Skills risks and mitigation plan**
  - *Do you have a proper study plan to cover all the necessary technologies?*

    This is the first time that many of us are touching these technologies, and there have and will be many issues with understanding these tools we decide to utilize. To mitigate these issues we have each member try to be expert on a given field of these technologies, so that they will be able to help the rest of the members study and improve at them.
- **Schedule risks**
  - *Does your team have a team schedule for every member including their detailed task?*

    Being able to organize our tasks is important, so to make sure that each member is on the same page with their tasks, and to mitigate any confusion that will likely pop up, we utilize Jira to schedule each member's tasks, so that it is clear what exactly each member needs to do. If there is any confusion about who is doing what, we have an organized system that each member can refer back to.
  - *If change happens, does it update transparently? Does your team use project management tool (e.g. Jira, Trello)*

    Changes with our plan and tasks happen especially in these early stages of development. To make sure that everyone is informed on any changes, especially on their assigned task, we use Jira to assign these tasks, and if there are any changes that need to be made, Jira will inform us outside of just our verbal communication.

- Teamwork risks (any issues related to teamwork)
    - *Everybody in the meeting regularly?*

      Given everyone's busy schedules, there are times when some people are not available to attend scheduled meetings. To accommodate this, we attempt to reschedule the meetings so that we are all able to attend. If this is not possible, then we record the meeting, and communicate with the missing member to inform them about what happened in the given meeting, and to mitigate any confusion or loss of information.
    - *Everybody keeps his/her pace? If not, what is your plan to mitigate the risks?*

      It is inevitable that we encounter times when a member might not be able to keep up their pace, so to mitigate the potential risks that may occur, we communicate with that member, and figure out potential solutions that will keep them on track.
- Legal/content risks (can you obtain content/SW you need legally with proper licensing, copyright).

There is a likely possibility that content we use is not originally made by us, and any software or tools we use that have certain licenses we need to acquire before we are legally able to use. We have thoroughly checked that we have the rights to use our software and tools for this project, and any content that we use for our web application(eg. pictures,artwork, music. etc.) is either free, or we have the express permission and/or license for the content.

7. Project management

For M2 specifically, during our class time we discussed what exactly we needed to do for this milestone by going through the document, and after we were on the same page with each other on what needed to be done, we then began throwing ideas together to complete these tasks. During the scrum meeting, we began by sharing our work and progress with each other, and discussing the direction of the project. After that we then looked back at the document and then discussed what still needed to be done. For this milestone, we started to utilize an application named Jira, which allowed us to schedule tasks for each member. We decided to use this as an easy way for our members to keep on track with their assigned tasks, and can be notified if there were any changes.