Foodie Technical RFC

Problem statement

We are on a mission to make eating well, effortless for everyone, everywhere. We want to give people more options when choosing how to eat. We want to help restaurants reach more customers and build their businesses. We want to make a great experience for everyone in the food ecosystem.

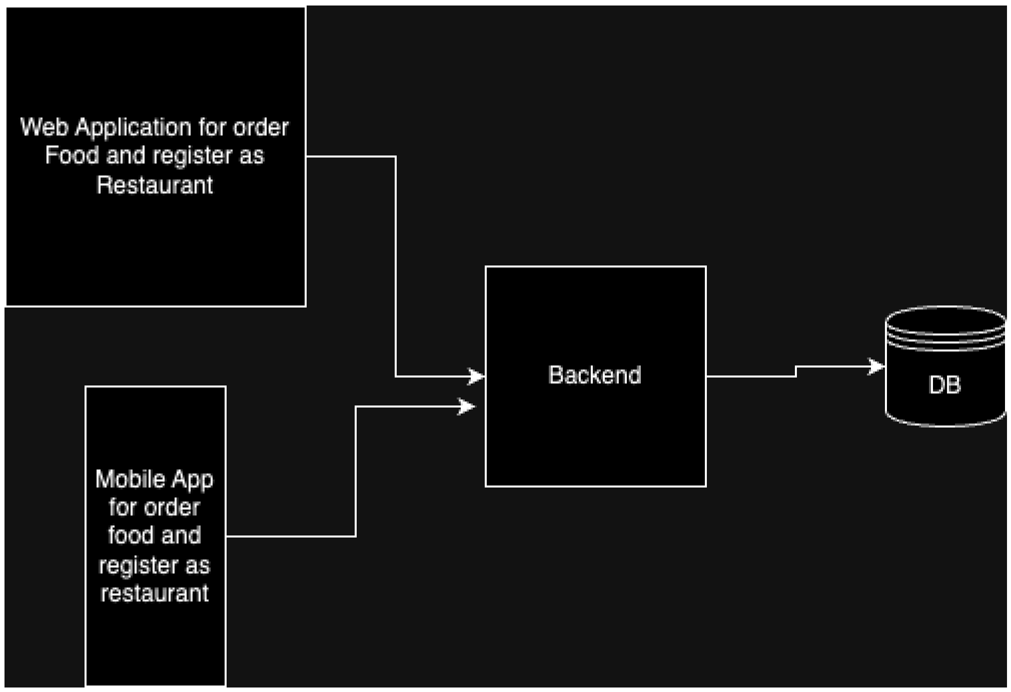
Tech Stack

- Frontend: React.js, Redux
- Backend: Node.js with Express.js
- Database: MongoDB for flexibility and scalability
- Hosting: AWS (Amazon Web Services) for reliability, scalability, and security

Overview

For customer - foodie.com

For restaurant - partern.foodie.com


Web Application for order Food and register as Restaurant
Mobile App for order food and register as restaurant
Backend
DB

Database models and thier relationships

we can two separate login page one for regular user and one for Restaurant Login

1. Regular user login
2. Restaurant login

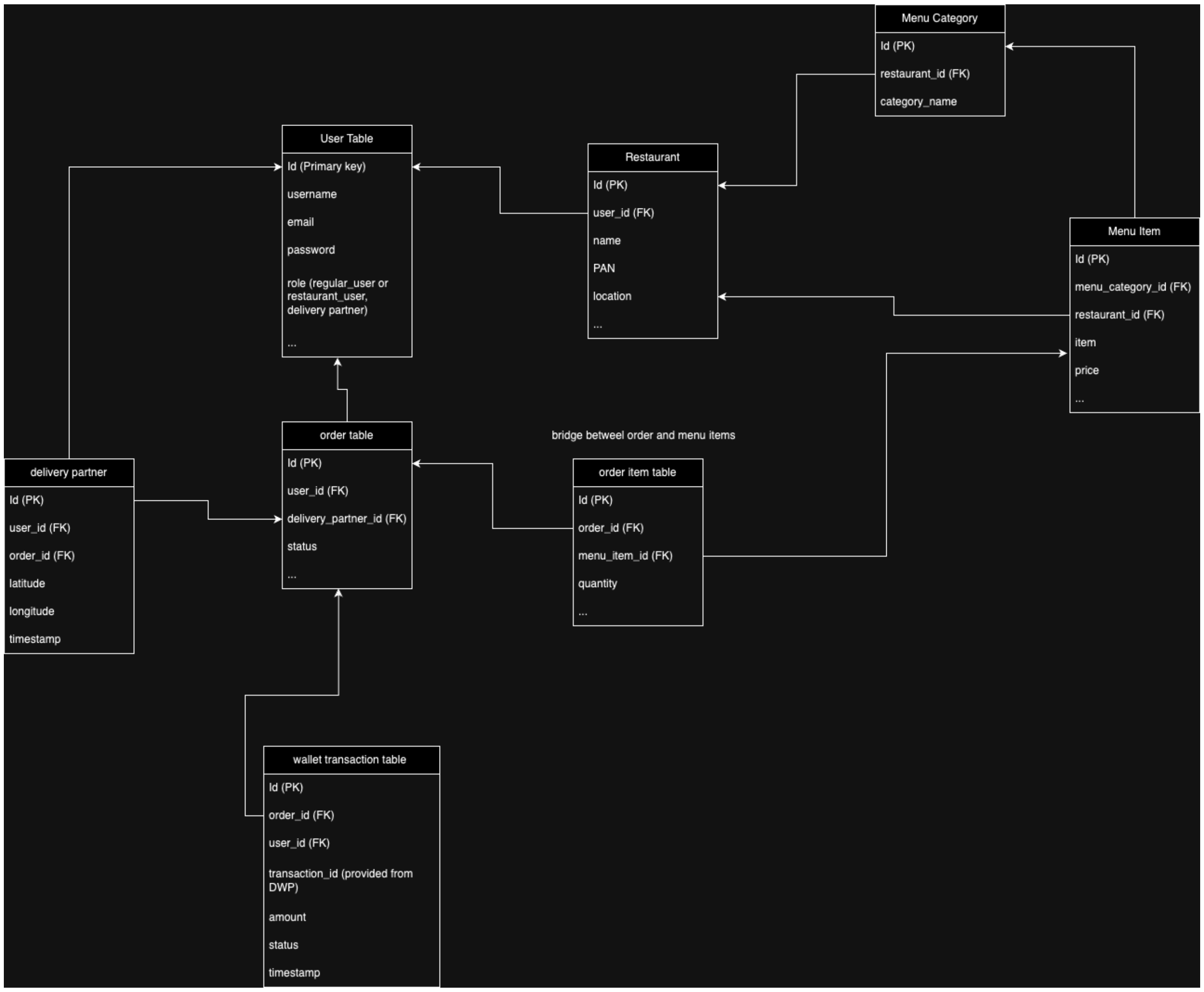
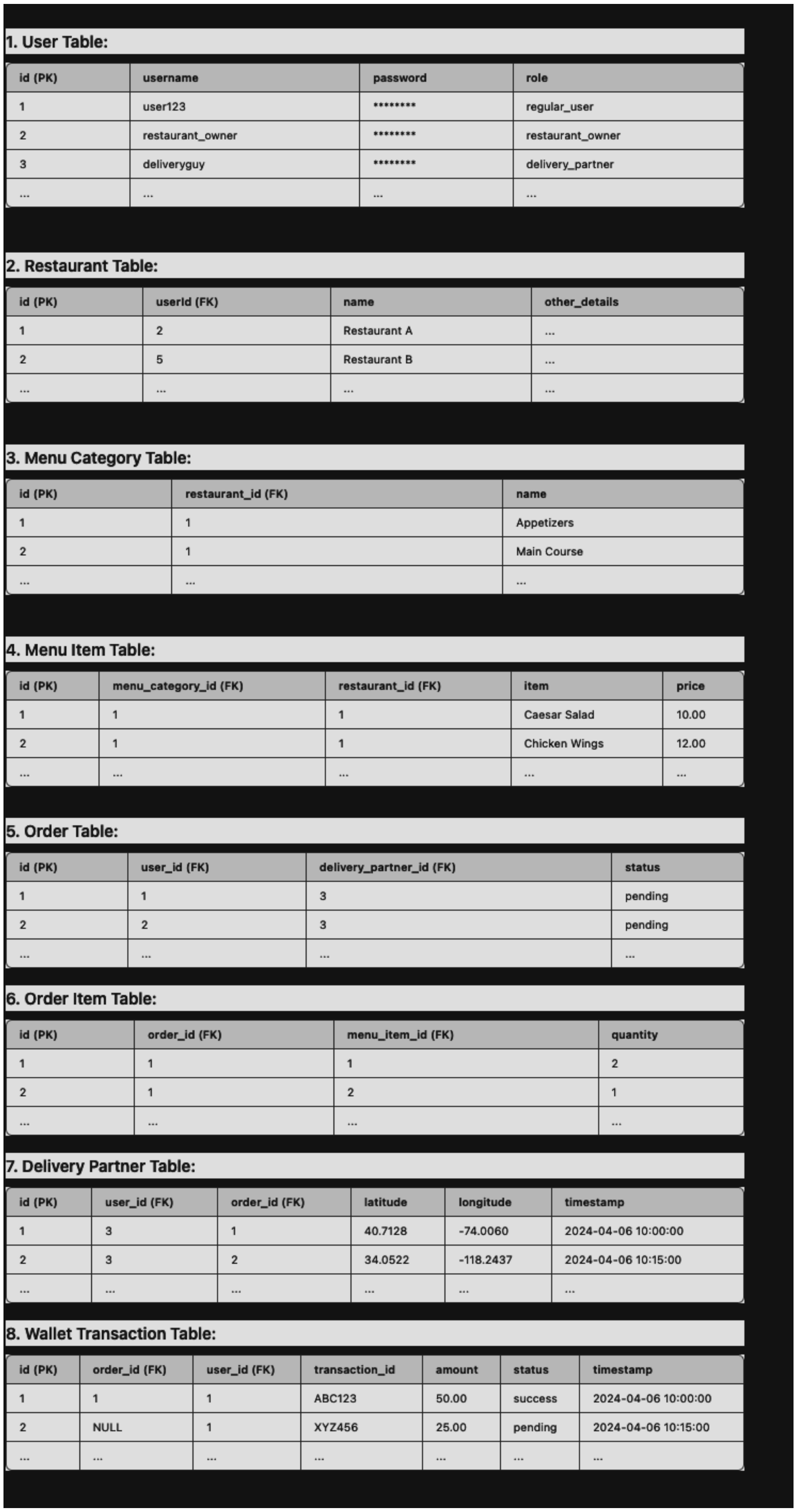


Table Representation



1. User Table:

id (PK)	username	password	role
1	user123	********	regular_user
2	restaurant_owner	********	restaurant_owner
3	deliveryguy	********	delivery_partner
...	...	...	...

2. Restaurant Table:

id (PK)	userId (FK)	name	other_details
1	2	Restaurant A	...
2	5	Restaurant B	...
...	...	...	...

3. Menu Category Table:

id (PK)	restaurant_id (FK)	name
1	1	Appetizers
2	1	Main Course
...	...	...

4. Menu Item Table:

id (PK)	menu_category_id (FK)	restaurant_id (FK)	item	price
1	1	1	Caesar Salad	10.00
2	1	1	Chicken Wings	12.00
...	...	...	...	...

5. Order Table:

id (PK)	user_id (FK)	delivery_partner_id (FK)	status
1	1	3	pending
2	2	3	pending
...	...	...	...

6. Order Item Table:

id (PK)	order_id (FK)	menu_item_id (FK)	quantity
1	1	1	2
2	1	2	1
...	...	...	...

7. Delivery Partner Table:

id (PK)	user_id (FK)	order_id (FK)	latitude	longitude	timestamp
1	3	1	40.7128	-74.0060	2024-04-06 10:00:00
2	3	2	34.0522	-118.2437	2024-04-06 10:15:00
...	...	...	...	...	...

8. Wallet Transaction Table:

id (PK)	order_id (FK)	user_id (FK)	transaction_id	amount	status	timestamp
1	1	1	ABC123	50.00	success	2024-04-06 10:00:00
2	NULL	1	XYZ456	25.00	pending	2024-04-06 10:15:00
...	...	...	...	...	...	...

Example

- Deepak wants to order food from two different restaurants, Restaurant A and Restaurant B.
- Restaurant A offers 5 menu items, and Restaurant B offers 5 menu items as well.
- Deepak decides to order 3 items from Restaurant A and 2 items from Restaurant B.

Order Table Representation:

id (PK)	user_id (FK)	delivery_partner_id (FK)	status
1	123	324355	pending

Order Item Table Representation:

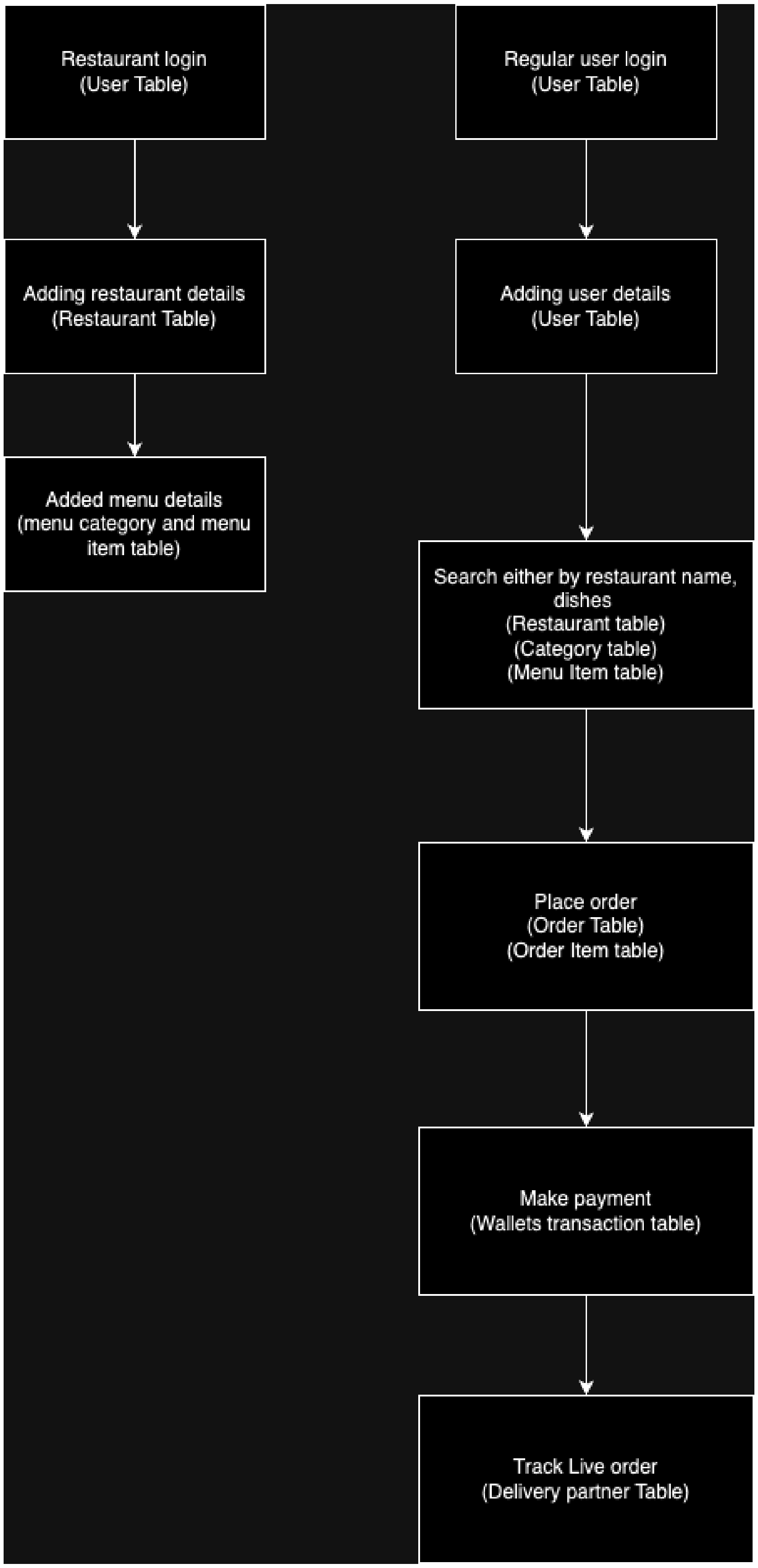
id (PK)	order_id (FK)	menu_item_id (FK)	quantity
1	1	1	2
2	1	2	1
3	1	6	1
4	1	8	1

In this representation:

- Deepak places an order with id 1 in the order table.
- In the order item table, each row represents an item ordered by Deepak.
- The order item table includes entries for each item ordered, along with the quantity.
- The order_id in the order item table (1 in this case) links all items to the same order placed by Deepak.

This representation allows for a clear understanding of the items Deepak has ordered from different restaurants and enables efficient tracking and management of orders within the system.

Flow of application


Restaurant login
(User Table)
Adding restaurant details
(Restaurant Table)
Added menu details
(menu category and menu item table)
Regular user login
(User Table)
Adding user details
(User Table)
Search either by restaurant name, dishes
(Restaurant table)
(Category table)
(Menu Item table)
Place order
(Order Table)
(Order Item table)
Make payment
(Wallets transaction table)
Track Live order
(Delivery partner Table)

Points validation

1. **A restaurant can register itself on the platform with the menu and pricing:**
   - **Validation:** The restaurant table allows restaurants to register with their details, and the menu and menu item tables enable them to add menu items with prices.
2. **A foodie can search for a restaurant by name, dish, or cuisine:**
   - **Validation:** The user can search for restaurants by name using the restaurant table. They can also search for specific dishes or cuisines by querying the menu and menu item tables.
3. **The foodie can order dishes from different restaurants simultaneously:**
   - **Validation:** The order and order item tables allow users to place orders for multiple dishes from different restaurants.
4. **Relying on a digital wallet partner for credit/debit transactions:**
   - **Validation:** The wallet transaction table stores transactions processed by the digital wallet partner, allowing for credit/debit of the customer's wallet.
5. **Restaurant can update menu and inventory and accept orders they can serve:**
   - **Validation:** The menu and menu item tables enable restaurants to update their menu and inventory. The order table ensures that restaurants only accept orders they can serve.
6. **Freshly prepared food delivered by trusted delivery partners:**
   - **Validation:** The delivery partner table tracks the real-time location of delivery partners, ensuring timely delivery of freshly prepared food.
7. **Foodies can track delivery partner's real-time location:**
   - **Validation:** The delivery partner location table allows foodies to track the real-time location of delivery partners while awaiting their meal.

Microservices Architecture

User Management Service:

- Responsible for user authentication, registration, and profile management.
- Handles user roles and permissions, such as regular users, restaurant owners, and delivery partners.

Restaurant Management Service:

- Manages restaurant profiles, including registration, updating details, and menu management.
- Handles inventory management, pricing, and availability of dishes.
- Manages menu categories and items offered by restaurants.
- Provides search and filtering functionalities for users to browse menus and find dishes.

Order Management Service:

- Manages the ordering process, including placing, processing, and tracking orders.
- Handles order status updates, notifications, and real-time order tracking for users.

Delivery Management Service:

- Coordinates with delivery partners to manage delivery logistics and tracking.
- Handles assignment of delivery orders, tracking delivery status, and updating user notifications.

Payment Service:

- Integrates with digital wallet partners to handle payment transactions securely.
- Manages payment authorization, processing, and status updates for orders.

Notification Service:

- Sends notifications to users and restaurant owners for order updates, promotions, and other relevant events.
- Handles email, SMS, or push notifications based on user preferences.

Search Service:

- Provides search functionality for users to find restaurants based on name, cuisine, or dish.
- Implements search indexing and querying to deliver relevant search results efficiently.

Analytics Service:

- Collects and analyzes data related to user behavior, order trends, and restaurant performance.
- Generates reports and insights to help optimize operations, marketing strategies, and user experiences.

Feedback Service:

- Manages user feedback and reviews for restaurants, dishes, and delivery experiences.
- Provides mechanisms for users to submit feedback and ratings, and for restaurant owners to respond and address feedback.

These microservices align with the modular and scalable architecture of Foodie.com, allowing for independent development, deployment, and scaling of functionalities while maintaining loose coupling and separation of concerns. Each service focuses on a specific aspect of the system, promoting flexibility, resilience, and maintainability in the overall design.