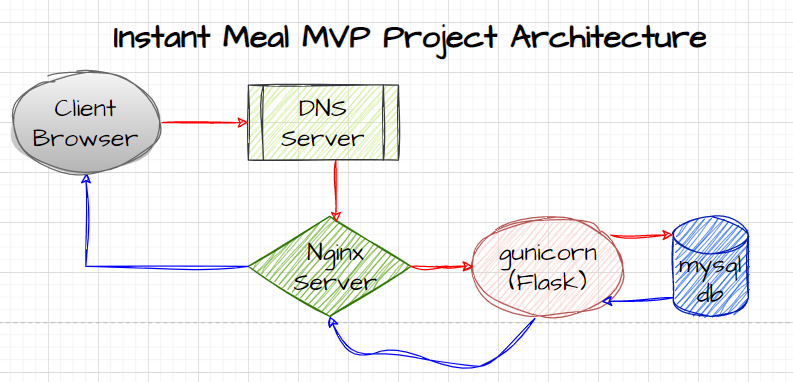
# Architecture



Here's a brief description of each component and how they are relevant to a food delivery application:

## 1. Client Browser:

- Description: The client browser is the user's interface to access and interact with the food delivery application. It renders the application's user interface, allowing users to browse restaurants, view menus, place orders, and track deliveries.

- Relevance: The client browser is the primary touchpoint for users, making it a crucial component for delivering a seamless and user-friendly experience. The browser interacts with the application's frontend to display information and enable user interactions.

## 2. DNS Server:

- Description: The Domain Name System (DNS) server translates human-readable domain names (like www.fooddeliveryapp.com) into IP addresses that computers use to identify each other on the internet.

- Relevance: The DNS server is essential for the food delivery application's accessibility. It ensures that users can reach the application by translating the domain name into the corresponding IP address. Reliable DNS resolution is crucial for users to access the application without issues.

## 3. NGINX Server:

- Description: NGINX is a web server that can also act as a reverse proxy server, load balancer, and HTTP cache. It handles incoming web requests and distributes them to the appropriate backend servers.

- Relevance: NGINX is relevant for the food delivery application's infrastructure. It can be used to enhance the application's performance, scalability, and security. NGINX can act as a reverse proxy to handle client requests and distribute them to Gunicorn servers running the Flask application.

## 4. Gunicorn (Flask):

- Description: Gunicorn (Green Unicorn) is a WSGI (Web Server Gateway Interface) server that serves Flask applications. It manages incoming requests, executes the application code, and returns the responses to the client.

- Relevance: Gunicorn, in conjunction with Flask, is the backend server responsible for processing business logic, handling orders, and managing user data. It ensures that the application can handle concurrent requests efficiently and scales appropriately to meet demand.

## 5. MySQL:

- Description: MySQL is a relational database management system (RDBMS) that stores and manages the application's data, including user profiles, restaurant information, order history, and more.

- Relevance: MySQL is crucial for the food delivery application's data persistence. It stores and retrieves data related to users, orders, menus, and other essential information. The database plays a key role in maintaining data integrity, consistency, and reliability for the application.

In summary, these components work together to provide a reliable, scalable, and efficient infrastructure for the food delivery application. The client browser ensures a user-friendly interface, DNS resolves domain names, NGINX manages web requests, Gunicorn with Flask handles application logic, and MySQL stores and retrieves data to support the overall functionality of the application.

# APIs and Methods

## List and description of the API routes we will be creating for our web client to communicate with our web server

1. User Authentication:

- `/api/user/register` (POST): Create a new user account.

- `/api/user/login` (POST): Authenticate and generate a user token.

- `/api/user/logout` (POST): Invalidate the user token and log out.

2. Restaurant Listings:

- `/api/restaurants` (GET): Get a list of available restaurants.

- `/api/restaurants/{id}` (GET): Get details of a specific restaurant.

3. Menu Operations:

- `/api/menu/{restaurantId}` (GET): Get the menu of a specific restaurant.

- `/api/menu/{restaurantId}/item` (POST): Add a new item to the menu.

- `/api/menu/{restaurantId}/item/{itemId}` (PUT): Update details of a menu item.

- `/api/menu/{restaurantId}/item/{itemId}` (DELETE): Remove a menu item.

4. Order Management:

- `/api/orders` (GET): Get a list of orders for the current user.

- `/api/orders/{orderId}` (GET): Get details of a specific order.

- `/api/orders` (POST): Place a new order.

- `/api/orders/{orderId}/cancel` (PUT): Cancel a pending order.

5. User Profile:

- `/api/user/profile` (GET): Get user profile information.

- `/api/user/profile` (PUT): Update user profile information.

- `/api/user/orders` (GET): Get a list of orders placed by the user.

6. Cart Operations:

- `/api/cart` (GET): Get the current contents of the user's shopping cart.

- `/api/cart/add` (POST): Add an item to the user's shopping cart.

- `/api/cart/remove` (POST): Remove an item from the user's shopping cart.

7. Payment and Checkout:

- `/api/checkout` (POST): Initiate the checkout process.

- `/api/payment/process` (POST): Process the payment for the order.

8. Address Management:

- `/api/user/addresses` (GET): Get a list of user addresses.

- `/api/user/addresses` (POST): Add a new address for the user.

- `/api/user/addresses/{addressId}` (PUT): Update details of a user address.

- `/api/user/addresses/{addressId}` (DELETE): Remove a user address.

## List and description of API endpoints or function/methods that we will be creating to allow other clients to use

1. User Authentication:

- `/api/auth/register` (POST): Register a new user.

- `/api/auth/login` (POST): Authenticate and generate a token for the user.

2. Restaurant Information:

- `/api/restaurants` (GET): Get a list of all available restaurants.

- `/api/restaurants/{id}` (GET): Get details about a specific restaurant.

3. Menu Operations:

- `/api/menu/{restaurant\_id}` (GET): Get the menu for a specific restaurant.

- `/api/menu/{restaurant\_id}/items` (POST): Add a new item to the menu.

- `/api/menu/{restaurant\_id}/items/{item\_id}` (PUT): Update details of a menu item.

- `/api/menu/{restaurant\_id}/items/{item\_id}` (DELETE): Remove a menu item.

4. Order Management:

- `/api/orders` (GET): Get a list of orders for the authenticated user.

- `/api/orders/{order\_id}` (GET): Get details about a specific order.

- `/api/orders` (POST): Place a new order.

- `/api/orders/{order\_id}/cancel` (PUT): Cancel a specific order.

5. User Profile:

- `/api/profile` (GET): Get the profile information for the authenticated user.

- `/api/profile` (PUT): Update user profile information.

6. Cart Operations:

- `/api/cart` (GET): Get the current contents of the user's shopping cart.

- `/api/cart/add` (POST): Add items to the user's shopping cart.

- `/api/cart/remove` (PUT): Remove items from the user's shopping cart.

7. Payment:

- `/api/payment` (POST): Make a payment for an order.

8. Address Management:

- `/api/addresses` (GET): Get a list of saved delivery addresses.

- `/api/addresses` (POST): Add a new delivery address.

- `/api/addresses/{address\_id}` (PUT): Update details of a delivery address.

- `/api/addresses/{address\_id}` (DELETE): Remove a delivery address.

9. Notifications:

- `/api/notifications` (GET): Get notifications for the user.

10. Search and Filters:

- `/api/search` (GET): Search for restaurants or menu items.

- `/api/filters` (GET): Get filtering options for restaurants or menu items.

## List and description of 3rd party APIs that we will be using

1. Google Maps API:

- Description: Google Maps API allows you to integrate maps, location-based services, and geocoding into your application. This is crucial for a food delivery app as it helps in providing accurate delivery estimates, real-time tracking, and location-based search for restaurants.

2. Stripe or Braintree for Payment Processing:

- Description: Both Stripe and Braintree are widely used payment processing APIs that provide secure and easy-to-implement payment solutions. They support various payment methods, including credit cards, digital wallets, and more. Seamless and secure transactions are essential for any food delivery application.

3. Twilio for SMS Notifications:

- Description: Twilio provides APIs for adding communication features such as SMS notifications to your application. You can use Twilio to send order confirmation messages, delivery updates, and promotional offers via SMS. This enhances the user experience and keeps customers informed about their orders.

4. OpenWeatherMap API:

- Description: Weather can impact food delivery times and customer preferences. Integrating the OpenWeatherMap API allows your application to provide real-time weather updates. This information can be used to adjust delivery estimates and offer personalized food recommendations based on the weather.

5. Yelp Fusion API:

- Description: The Yelp Fusion API provides access to a vast database of restaurant information, reviews, ratings, and more. Integrating Yelp's data can enhance your app by offering users detailed information about restaurants, helping them make informed decisions based on reviews and ratings.

6. Push Notification Services (e.g., OneSignal, Firebase Cloud Messaging):

- Description: Implementing a push notification service is essential for keeping users engaged and informed about order status, promotions, and other relevant updates. OneSignal and Firebase Cloud Messaging are popular choices for sending push notifications to both iOS and Android devices.

7. Nutritionix API:

- Description: For health-conscious users, integrating the Nutritionix API can provide nutritional information for various food items on the menu. This adds value to your app by allowing users to make healthier food choices and track their nutritional intake.

# Data Model

# User Stories

## 1. Placing an Order

- As a user that is always busy in during office hours, I want to be able to browse through a list of nearby restaurants, view their menus, and add items to my cart.

- Acceptance Criteria:

- The application displays a list of nearby restaurants with relevant details (name, cuisine, ratings).

- Users can click on a restaurant to view its menu.

- Users can easily add items from the menu to their cart.

- The cart updates in real-time, showing the selected items and the total cost.

- Users can review their order and proceed to checkout.

## 2. Tracking Order Status

- As a user that is always busy in during office hours, after placing an order, I want to receive real-time updates on the status of my delivery.

- Acceptance Criteria:

- Users receive an order confirmation with estimated delivery time via push notification or SMS.

- The app provides real-time tracking of the delivery, showing the delivery person's location on a map.

- Users receive notifications when the order is out for delivery and when it has been successfully delivered.

- Users can view a history of their past orders with details on each order's status.

## 3. Customizing and Saving Preferences

- As a user that is always busy in during office hours, I want the ability to customize my food preferences, save favorite restaurants, and easily reorder previous meals.

- Acceptance Criteria:

- Users can create and manage a profile with personal details and food preferences.

- The app allows users to mark favorite restaurants for quick access.

- Users can easily reorder items from their order history.

- Customization options, such as dietary preferences and spice levels, are available during the ordering process.

## 4. Providing and Viewing Reviews

- As a user that is always busy in during office hours, I want to read and leave reviews for restaurants to make informed decisions about where to order from.

- Acceptance Criteria:

- Users can view restaurant ratings and reviews when browsing through the list of available restaurants.

- After completing an order, users are prompted to leave a review and provide a rating for the restaurant and the delivery service.

- Users can filter and sort restaurant listings based on ratings and reviews.

## 5. Handling Payment

- As a user that is always busy in during office hours, I want a secure and seamless payment process for my orders.

- Acceptance Criteria:

- Users can securely enter and save multiple payment methods (credit cards, digital wallets).

- The application supports popular payment gateways such as Stripe or Braintree.

- Users receive clear order summaries with detailed pricing before confirming payment.

- Payment confirmation and receipts are sent to users via email and/or within the app.

# Mockups