MyKitchen

Software Design Document

CS 4850 Senior Project

Spring 2024

Professor Perry

2/9/24

Indy-4 Kitchen

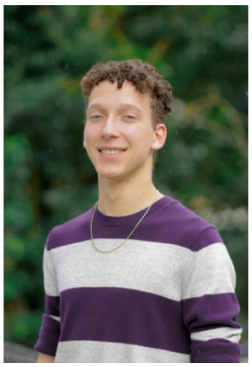
Merrick	Nick	Minchul
		

Table of Contents

1. Introduction

1.1 Overview

This Software Design Document (SDD) outlines the architectural and design details of the MyKitchen platform, a pioneering online marketplace that connects home cooks with local food enthusiasts. The document provides a comprehensive overview of the system's architecture, user interface design, data management, security measures, and interaction flows. It serves as a blueprint for developers, designers, and project stakeholders, ensuring a unified understanding and coherent approach to building a robust, user-friendly platform that transforms home kitchens into ghost kitchens.

1.2 Project Scope

The MyKitchen project aims to develop a web-based platform using React for the frontend and PHP for the backend, with data stored in a MySQL database. Hosted on AWS for scalability and reliability, the platform will feature user authentication, profile management, kitchen profiles, dish listings, and a Stripe integration for secure payment processing. The scope encompasses creating an intuitive user experience that encourages culinary discovery and supports small-scale kitchen operations, promoting a community-centric food ecosystem.

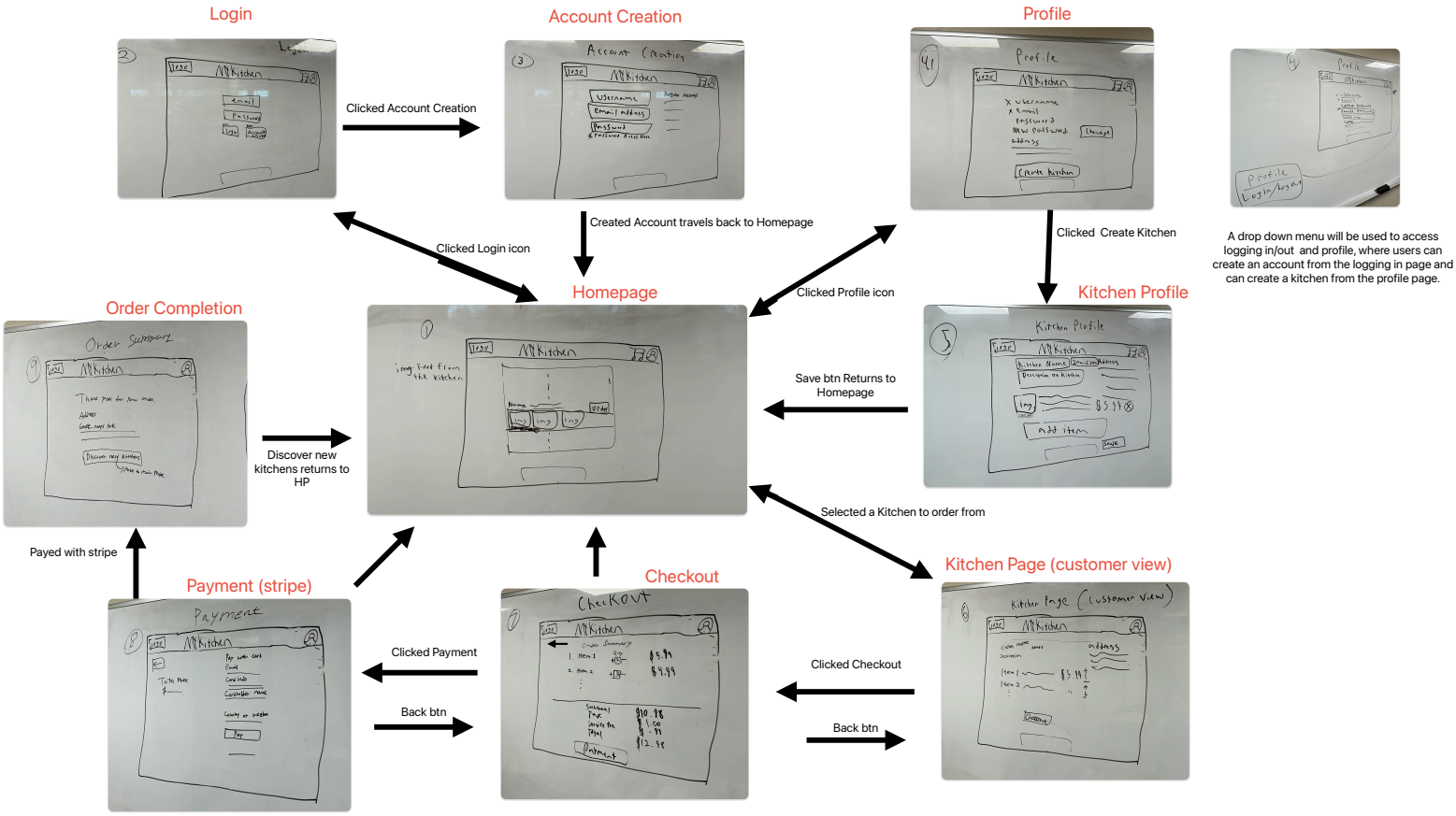
1.3 Definitions and Acronyms

UI: User Interface - The space where interactions between humans and the platform occur.
AWS: Amazon Web Services - The cloud computing service used for hosting the platform.
API: Application Programming Interface - A set of rules that allows different software entities to communicate with each other.
React: A JavaScript library for building user interfaces, utilized for frontend development.
PHP: Hypertext Preprocessor - A server-side scripting language used for backend development.
MySQL: An open-source relational database management system, used for storing user data, kitchen profiles, and transaction records.
Stripe: An online payment processing platform, integrated into MyKitchen for handling transactions.

2.0 User Interface

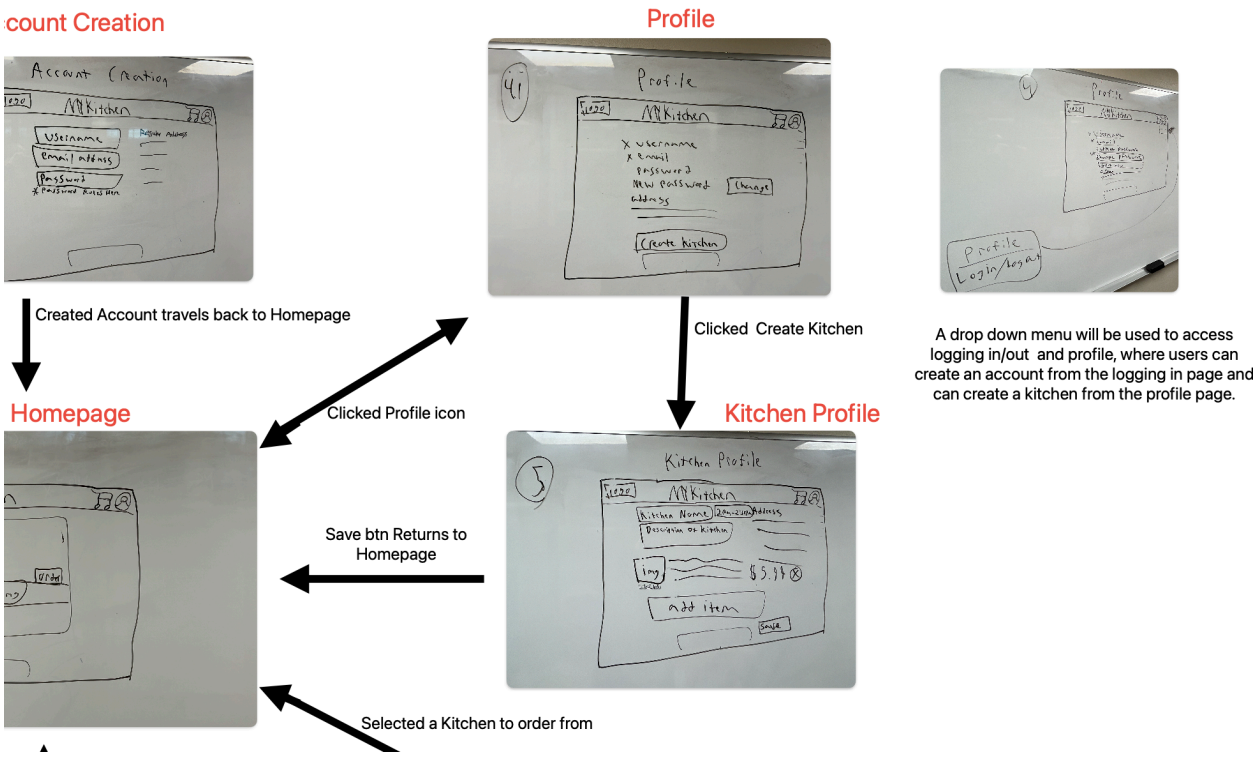
2.1 High Level UI architecture

Below is high level UI diagram to show the flow that the website should follow. Showcasing is a design that revolves around the homepage of the website that will attract users in discovering ghost kitchens around them. What follows is a list of cycles (use cases) that the users will likely go through.



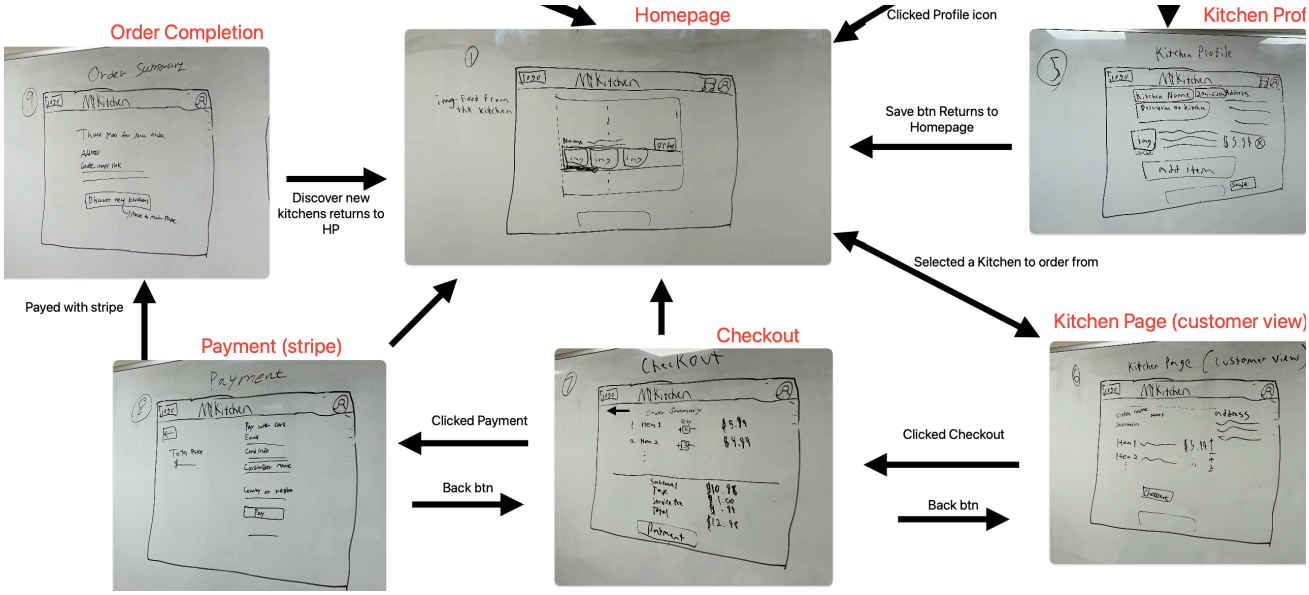
2.1.1 Account Flow

- **Objective**: Enable users to create a new account or log into an existing one to access personalized features such as profile editing, kitchen creation, and dish shopping.
- **Process**:
  1. **Initiation**: Users initiate the process by selecting the 'Login' button on the homepage.
  2. **Authentication**:
     - Existing users are directed to the login page to enter credentials.
     - New users are guided to the account creation page via a dedicated 'Create Account' button.
  3. **Account Creation**:
     - The account creation form captures essential user information. Upon submission via the 'Submit' button, users are redirected to the homepage, now authenticated.
- **Design Considerations**:
  1. Visual cues for login and account creation should be prominent on the homepage.
  2. Form fields must include validation with clear feedback for user errors.
  3. Navigation post-authentication should offer direct access to user profiles and other personalized areas.



2.1.2 Profile Flow

- **Objective**: Provide a seamless process for users to access and edit their profile, including personal information and kitchen details.
- **Process**:
    1. **Profile Access**: Authenticated users access their profile through a clearly marked icon, leading to the profile edit page.
    2. **Information Editing**: Users can update personal details, such as address and password.
    3. **Kitchen Creation**: A 'Create Kitchen' option within the profile page allows users to establish a kitchen profile, where they can add and describe dishes, set prices, and upload images.
- **Design Considerations**:
    1. The kitchen profile creation interface should mirror the user profile for consistency.
    2. Adding dishes must be intuitive, with the ability to dynamically add multiple entries.
    3. Upon completion, a clear path back to the homepage should be provided.



2.1.3 Kitchen Purchase Flow

- **Objective**: Facilitate a straightforward and secure process for users to browse kitchens, select dishes, and complete purchases.
- **Process**:
  1. **Kitchen Selection**: From the homepage, users can browse and select a kitchen, leading to a customer-view kitchen page.
  2. **Ordering**: Users add or remove dishes to/from their cart, with clear indications of selections and totals.
  3. **Checkout**:
     - A 'Checkout' button leads to a summary page displaying selected items, quantities, and prices, including subtotals and total cost.
     - The 'Pay' button redirects users to a Stripe-powered payment page for secure transaction completion.
  4. **Order Completion**: Post-payment, users are directed to an order completion page with a thank-you message, kitchen address, and pickup instructions.
- **Design Considerations**:
  1. The shopping cart and checkout process must be user-friendly, with easy navigation back to the kitchen page or cart adjustments.
  2. Integration with Stripe must ensure a secure and seamless payment experience, with clear total cost visibility.
  3. The order completion page should confirm the transaction success and provide practical pickup information.

3.0 System Architecture

3.1 System Architecture Overview

The MyKitchen platform operates with two clients, customers and kitchen owners, the goal of the website is to help customers find local kitchens and kitchen owners to get more customers. Underneath are diagrams that expose how the separate subsystems of the platform will function.

3.2 Database

The MyKitchen platform will utilize a MySQL database to manage and store all essential data entities including orders, products (dishes), user addresses, kitchen addresses, kitchens, and user information. The schema is designed to ensure data integrity and efficient access patterns that support the platform's functionality.

3.2.1 Entities and Relationships

- **Users**: This entity stores user information. Each record represents a registered user of the platform, including authentication details and user-specific data.
- **User Addresses**: Associated with the User entity, this entity stores addresses for each user. There is a one-to-one relationship between User Addresses and Users, ensuring each user has a unique address associated with their profile.
- **Kitchens**: Represents the kitchens operated by users on the platform. This entity includes information about the kitchen such as name, description, and operational details.
- **Kitchen Addresses**: Similar to User Addresses, each Kitchen Address is linked to a specific Kitchen entity. This one-to-one relationship ensures each kitchen has a unique address, facilitating accurate location-based services.
- **Products (Dishes)**: This entity catalogues the various dishes offered by kitchens. It includes details such as dish name, ingredients, price, and the kitchen offering the dish.
- **Orders**: The Orders entity captures transactions on the platform, linking users to the dishes they purchase. Each order is associated with one or more Products, reflecting the items included in the purchase.

3.2.2 Data Integrity and Constraints

- **Foreign Keys**: To maintain referential integrity, foreign keys are used to link related entities. For example, each User Address is linked to its corresponding User via a foreign key, and each Product (Dish) in an Order is linked back to the Orders entity.

- **Unique Constraints**: Entities such as Users and Kitchens have unique constraints on identifiers and potentially on other attributes like email addresses or kitchen names to prevent duplicates.

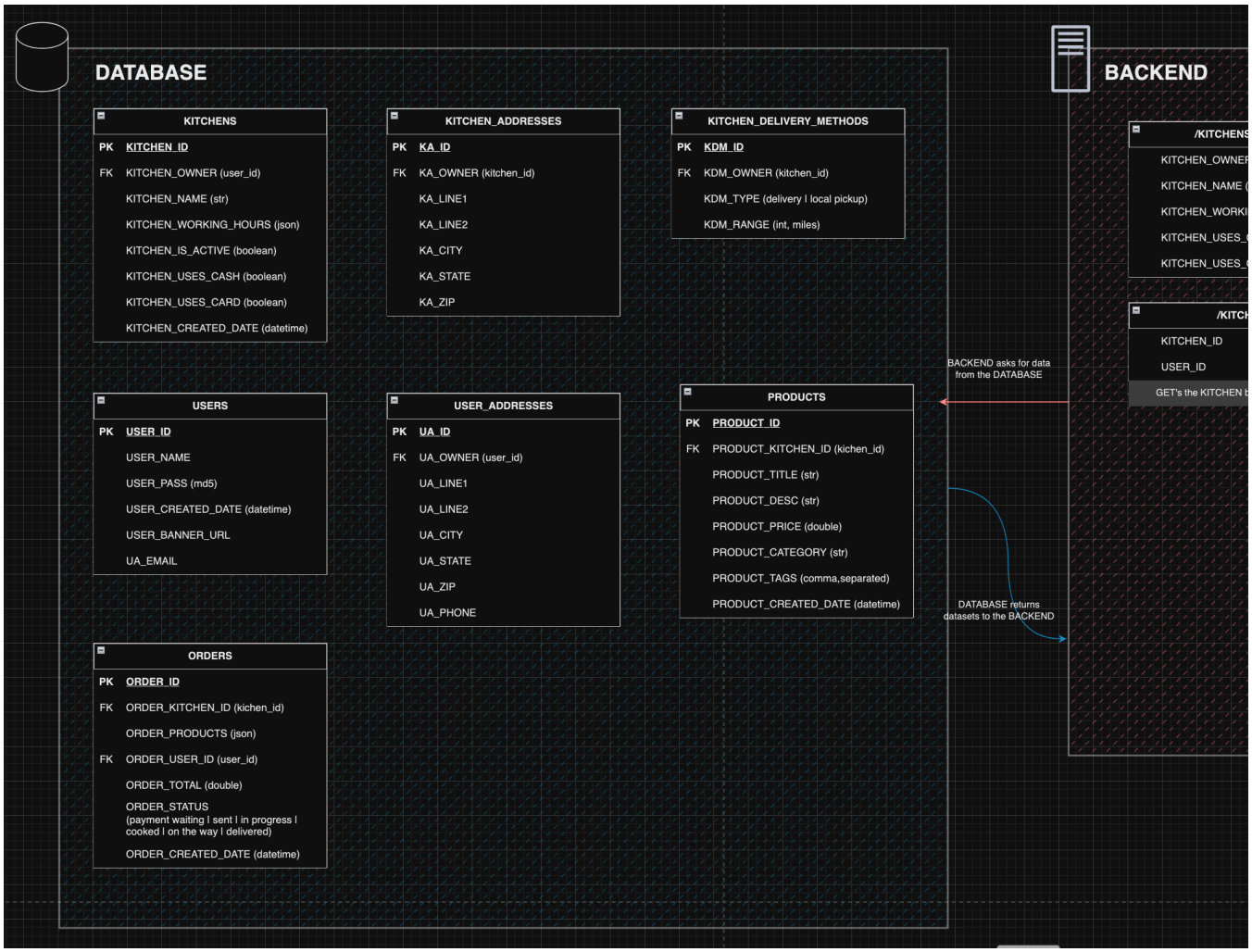
- **Indexing**: Critical attributes, particularly those involved in search operations (e.g., user names, kitchen names, dish names), will be indexed to enhance query performance.

3.3 Backend

The backend of the MyKitchen platform is designed with a focus on security, modularity, and efficient data management. It serves as the intermediary layer between the frontend interface and the MySQL database, ensuring that direct communication between the frontend and the database is restricted to maintain data integrity and security.

3.3.1 Communication Protocol

- **Backend-to-Database**: The backend exclusively handles all interactions with the database. It performs operations such as querying data, updating records, and managing transactions. These operations are facilitated through secure API calls to the database, leveraging prepared statements and parameterized queries to prevent SQL injection and other security vulnerabilities.



- **Frontend-to-Backend**: The frontend interacts with the backend through a well-defined set of API endpoints. These endpoints are designed to handle requests for data retrieval and submission, user authentication, profile updates, and other user-driven actions. The API ensures data is securely exchanged between the frontend and backend, employing authentication tokens and data validation to safeguard user information and actions.

3.3.2 Responsibilities

- **User Data Management**: The backend is responsible for updating and managing user-related information, including user profiles, kitchen profiles, addresses, and account settings. It ensures that any changes made by users through the frontend are accurately reflected in the database.

- **Content Delivery**: It dynamically generates and updates content on user pages, such as kitchen listings, user profiles, and product (dish) information, ensuring users have access to the latest data.

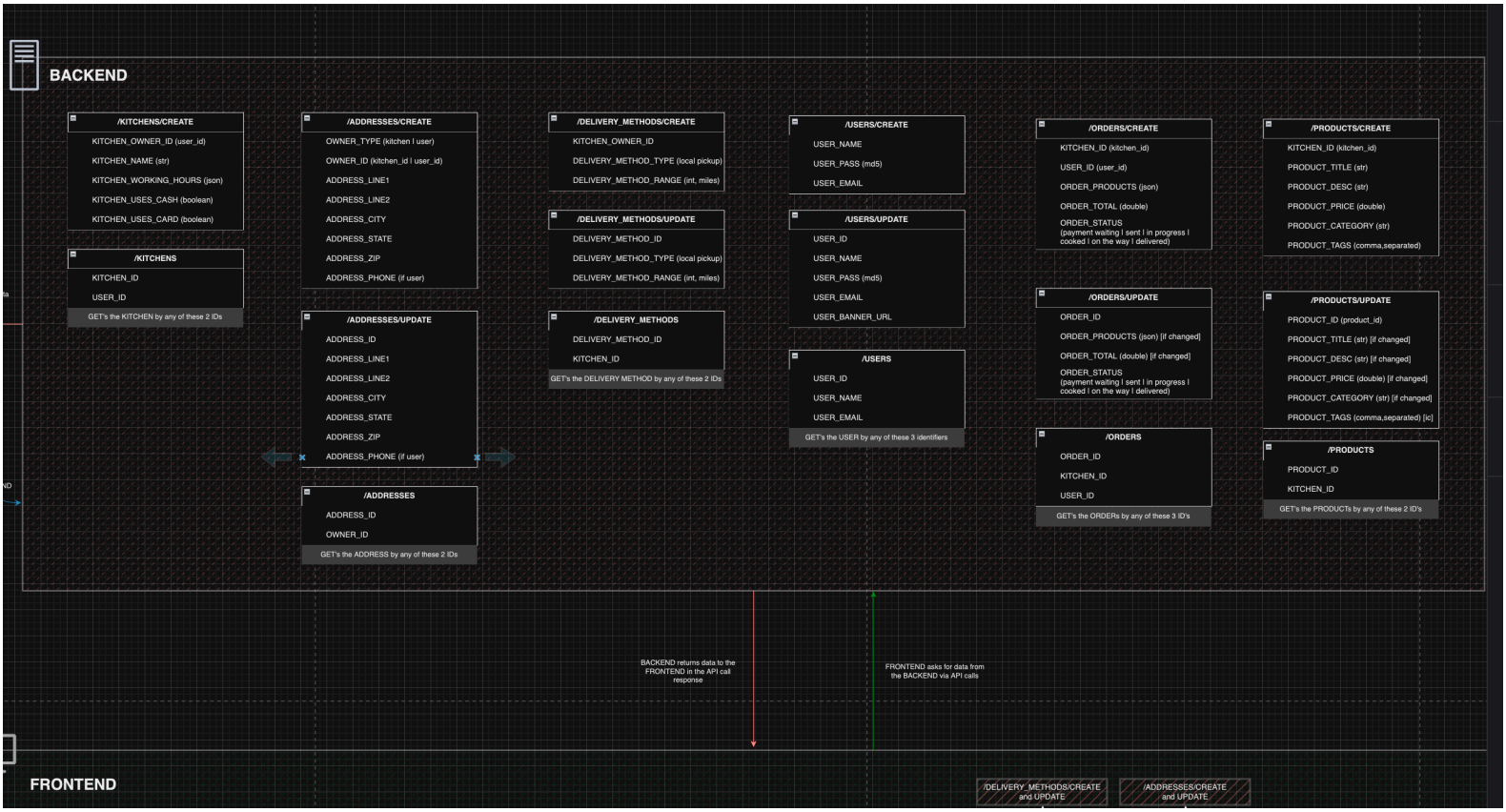
3.3.3 Hosting

- **AWS EC2 Hosting**: The backend infrastructure will be hosted on Amazon Web Services (AWS) EC2 instances. This choice provides scalable computing capacity in the AWS cloud, allowing for flexibility in configuring, managing, and scaling server instances as per the platform's requirements. AWS EC2 also offers robust security features, including network isolation, encryption, and secure access configurations, aligning with the platform's security objectives.

3.3.4 API Design

- **API Endpoints**: The backend will expose a series of RESTful API endpoints to facilitate communication with the frontend. These endpoints will be designed to support various platform functionalities, including authentication, data retrieval, content management, and transaction processing.

- **Data Retrieval**: Upon receiving requests from the frontend, the backend will query the database for the required information and return it to the frontend in a structured format, typically JSON, ensuring seamless integration and dynamic content rendering on the frontend.


BACKEND
/KITCHENS/CREATE
KITCHEN_OWNER_ID (user_id)
KITCHEN_NAME (str)
KITCHEN_WORKING_HOURS (json)
KITCHEN_USES_CASH (boolean)
KITCHEN_USES_CARD (boolean)
/KITCHENS
KITCHEN_ID
USER_ID
GET's the KITCHEN by any of these 2 IDs
/ADDRESSES/CREATE
OWNER_TYPE (kitchen I user)
OWNER_ID (kitchen_id I user_id)
ADDRESS_LINE1
ADDRESS_LINE2
ADDRESS_CITY
ADDRESS_STATE
ADDRESS_ZIP
ADDRESS_PHONE (if user)
/ADDRESSES/UPDATE
ADDRESS_ID
ADDRESS_LINE1
ADDRESS_LINE2
ADDRESS_CITY
ADDRESS_STATE
ADDRESS_ZIP
ADDRESS_PHONE (if user)
/ADDRESSES
ADDRESS_ID
OWNER_ID
GET's the ADDRESS by any of these 2 IDs
/DELIVERY_METHODS/CREATE
KITCHEN_OWNER_ID
DELIVERY_METHOD_TYPE (local pickup)
DELIVERY_METHOD_RANGE (int, miles)
/DELIVERY_METHODS/UPDATE
DELIVERY_METHOD_ID
DELIVERY_METHOD_TYPE (local pickup)
DELIVERY_METHOD_RANGE (int, miles)
/DELIVERY_METHODS
DELIVERY_METHOD_ID
KITCHEN_ID
GET's the DELIVERY METHOD by any of these 2 IDs
/USERS/CREATE
USER_NAME
USER_PASS (md5)
USER_EMAIL
/USERS/UPDATE
USER_ID
USER_NAME
USER_PASS (md5)
USER_EMAIL
USER_BANNER_URL
/USERS
USER_ID
USER_NAME
USER_EMAIL
GET's the USER by any of these 3 identifiers
/ORDERS/CREATE
KITCHEN_ID (kitchen_id)
USER_ID (user_id)
ORDER_PRODUCTS (json)
ORDER_TOTAL (double)
ORDER_STATUS
(payment waiting I sent I in progress I cooked I on the way I delivered)
/ORDERS/UPDATE
ORDER_ID
ORDER_PRODUCTS (json) [if changed]
ORDER_TOTAL (double) [if changed]
ORDER_STATUS
(payment waiting I sent I in progress I cooked I on the way I delivered)
/ORDERS
ORDER_ID
KITCHEN_ID
USER_ID
GET's the ORDERs by any of these 3 ID's
/PRODUCTS/CREATE
KITCHEN_ID (kitchen_id)
PRODUCT_TITLE (str)
PRODUCT_DESC (str)
PRODUCT_PRICE (double)
PRODUCT_CATEGORY (str)
PRODUCT_TAGS (comma,separated)
/PRODUCTS/UPDATE
PRODUCT_ID (product_id)
PRODUCT_TITLE (str) [if changed]
PRODUCT_DESC (str) [if changed]
PRODUCT_PRICE (double) [if changed]
PRODUCT_CATEGORY (str) [if changed]
PRODUCT_TAGS (comma,separated) [ic]
/PRODUCTS
PRODUCT_ID
KITCHEN_ID
GET's the PRODUCTs by any of these 2 ID's
BACKEND returns data to the FRONTEND in the API call response
FRONTEND asks for data from the BACKEND via API calls
FRONTEND
/DELIVERY_METHODS/CREATE and UPDATE
/ADDRESSES/CREATE and UPDATE