**Snack Squad: Food Delivery App Documentation**

**Table of Contents**

1. **Overview**
2. **User Roles & Permissions**
3. **Core Features**
4. **System Architecture**
5. **API Endpoints**
6. **Data Models**
7. **Mobile App Flow**
8. **Security Considerations**

**Presented by :**

**R . jai akash (6031BC98A6ED5F4A13142A8F3DC9E772)**

**V.S VIGNESH(5E527CO742A34A2977B4DDF555DB27C4)**

**E.GOBIKRISHNAN(5EBB6E0887E384CB24B4D0AB97850BB2)**

**R.SRIHARI(EFA5FF9EA7AA4C7901084899610BD8E34)**

**M.NARENDRAN(6E609400400A15BBD0841A7247B8EC5C)**

**Introduction**

Welcome to **Snack Squad**, a food delivery application designed to help users easily browse restaurants, order food, and track their deliveries. This document provides an overview of the Snack Squad app’s features, user flows, and technical specifications.

1. **Overview**

**Overview of Snack Squad**

**Snack Squad** is an innovative and user-centric **food delivery platform** designed to offer customers a seamless, efficient, and personalized food ordering experience. It connects users with local restaurants and delivery drivers, making it easy to explore menus, place orders, track deliveries, and enjoy a wide variety of meals—all from the convenience of a mobile app.

Whether you're craving a late-night snack, ordering lunch for the office, or planning a weekend dinner, **Snack Squad** provides a reliable, fast, and intuitive platform to meet all your food delivery needs.

**Key Features**

* **Restaurant Discovery**: Browse through a wide range of local restaurants, cafes, and eateries by category, cuisine, or proximity.
* **Personalized Recommendations**: Get AI-driven suggestions based on your taste preferences, past orders, and dietary restrictions.
* **Menu Customization**: Customize your meals with various options, including toppings, portion sizes, and special requests.
* **Seamless Ordering Process**: Effortlessly add items to your cart, choose delivery options, and securely pay using multiple payment methods.
* **Real-Time Order Tracking**: Track your order from the kitchen to your doorstep, with live updates on the delivery status and the driver’s location.
* **Contactless Delivery**: Choose from various delivery options, including contactless drop-offs and location-based delivery preferences.
* **Loyalty & Rewards**: Earn points or credits with each order and redeem them for discounts, free delivery, or exclusive offers.

**Platform Components**

1. **Customer App**: The main interface where users discover restaurants, customize their orders, make payments, and track deliveries. Features a clean and intuitive UI for browsing, order management, and customer support.
2. **Restaurant Dashboard**: A web-based portal for restaurant owners and managers to manage their menu items, track incoming orders, and view performance analytics. Restaurants can also customize their delivery windows and communicate directly with customers.
3. **Driver App**: The app used by delivery drivers to receive order assignments, navigate to pickup and drop-off locations, and communicate with customers. Real-time location tracking helps ensure timely deliveries and provides updates to customers.
4. **Backend API & Infrastructure**: The server-side infrastructure that handles authentication, order management, payment processing, customer support, and real-time order tracking.

**Target Audience**

* **Customers**: Individuals looking for a fast, easy, and reliable food delivery service. This includes urban dwellers, office workers, families, students, and anyone looking to order food from local restaurants.
* **Restaurant Owners**: Restaurants that want to expand their customer base and streamline the order fulfillment process. Snack Squad provides a platform for restaurants to reach a wider audience.
* **Delivery Drivers**: Individuals seeking flexible, on-demand work delivering food to customers.

**Technology Stack**

* **Frontend**: Mobile apps built for iOS and Android using technologies such as **React Native** for cross-platform development, ensuring a consistent experience across devices.
* **Backend**: A microservices-based architecture, leveraging technologies like **Node.js** or **Django** for API development, **PostgreSQL** or **MongoDB** for database management, and **Redis** for caching and real-time notifications.
* **Payment Gateway**: Integrated with third-party services like **Stripe**, **PayPal**, or **Apple Pay** for secure and seamless payment processing.
* **Real-Time Tracking**: **WebSockets** or **Firebase** for live updates on delivery status and driver location.
* **Cloud Infrastructure**: Hosted on **AWS** or **Google Cloud** to scale with increasing demand, providing reliability and redundancy.
* **Security**: Comprehensive security measures including **TLS encryption**, **OAuth 2.0** for authentication, and **PCI-DSS** compliance for secure payment handling.

**Business Model**

**Snack Squad** operates on a commission-based revenue model, earning a percentage from each transaction made through the platform. The company generates revenue in the following ways:

1. **Commission from Restaurants**: A percentage of each order placed through the platform.
2. **Delivery Fees**: A flat or dynamic delivery fee charged to customers, depending on factors like order size, distance, and delivery time.
3. **Promotions and Advertising**: Restaurants can opt to pay for premium placement in the app or advertise special promotions to users.
4. **Subscription Plans**: Offering users premium subscription plans with benefits such as free delivery, exclusive discounts, or priority support.

**Vision and Future Direction**

The goal of **Snack Squad** is to become the leading food delivery platform by offering a superior user experience, expanding the variety of restaurants and cuisines, and continuously evolving the platform with innovative features. Future growth will focus on:

* **AI-driven personalization** to provide increasingly tailored meal and restaurant recommendations.
* **Sustainability** initiatives like eco-friendly packaging and carbon offset programs to make food delivery more environmentally conscious.
* **Expanding globally** with localized services, supporting multiple languages and currencies, and offering region-specific food options.
* **Exploring autonomous delivery methods** such as drones or robots to improve delivery efficiency and reduce costs.

Snack Squad is a food delivery service that allows customers to:

* Browse restaurant menus
* Customize and order meals
* Track delivery status in real-time
* Rate restaurants and food items

The application is designed to be user-friendly with an intuitive interface, optimized for both Android and iOS devices. The service integrates with restaurants to offer delivery, offering a wide range of cuisine options.

**2. User Roles & Permissions**

**2.1. Customer**

* **Permissions**: Browse restaurants, place orders, track deliveries, rate restaurants/food.
* **Features**:
  + Account creation & login.
  + Search and filter restaurants by type, rating, and delivery time.
  + Real-time order tracking.
  + Review system for restaurants/food.

**2.2. Restaurant Owner**

* **Permissions**: Manage menu, view orders, update delivery status, respond to customer reviews.
* **Features**:
  + Dashboard to manage restaurant info (name, hours, contact details).
  + Manage menu items (add, edit, delete).
  + View and manage incoming orders.
  + Update delivery status.

**2.3. Delivery Driver**

* **Permissions**: View delivery tasks, mark deliveries as complete.
* **Features**:
  + See real-time orders assigned to them.
  + Navigate to restaurant & customer locations via integrated map.
  + Mark deliveries as completed.

**2.4. Admin**

* **Permissions**: Full access to app features, user management, and analytics.
* **Features**:
  + Dashboard with metrics (total orders, revenue, user stats).
  + Manage users, restaurants, and delivery drivers.

## View and resolve issues like complaints or disputes.

## User Roles and Permissions

Snack Squad supports multiple user roles, each with specific permissions and access to different parts of the application. This allows for tailored functionality based on the role the user plays in the system, such as a customer, restaurant owner, delivery driver, or admin.

Below is a breakdown of each user role and the permissions associated with them.

**1. Customer**

A **Customer** is an individual who uses the Snack Squad app to browse restaurants, place orders, and receive deliveries.

**Permissions:**

* **Account Management:**
  + Create a new account (sign up).
  + Log in using email, social media accounts, or other authentication methods.
  + Edit personal details (name, address, contact info).
  + Reset password.
* **Restaurant Interaction:**
  + Browse restaurant listings based on location, cuisine, or rating.
  + Filter restaurants by various criteria (e.g., price, rating, delivery time).
  + View restaurant menus, including prices and available items.
  + Customize orders (e.g., special requests, item substitutions, etc.).
* **Order Management:**
  + Add items to the cart and proceed to checkout.
  + Choose delivery or pickup options.
  + Make payments using supported methods (credit card, digital wallets, cash).
  + Track the status of current orders (e.g., preparing, dispatched, out for delivery).
  + Cancel orders before they are prepared (if allowed).
* **Reviews and Feedback:**
  + Leave ratings and reviews for restaurants after an order is completed.
  + View past order history, including ratings and feedback given.
* **Notifications:**
  + Receive real-time notifications for order status updates (e.g., "Order Confirmed", "Out for Delivery").
  + Receive promotional offers or discounts (if applicable).

**2. Restaurant Owner**

A **Restaurant Owner** is a business owner or manager who controls the restaurant’s profile, menu, and order management within the Snack Squad system.

**Permissions:**

* **Account and Profile Management:**
  + Create and manage the restaurant’s profile (restaurant name, location, contact details, working hours).
  + Upload and update restaurant logo, banner, and other branding elements.
* **Menu and Item Management:**
  + Add, update, or remove menu items (including pricing, description, and images).
  + Set item availability based on stock or hours of operation.
  + Create special menu items or promotions (e.g., limited-time offers).
* **Order Management:**
  + View incoming orders and update the status (e.g., "Preparing", "Ready for Pickup", "Delivered").
  + Accept or reject orders based on availability or capacity.
  + Monitor and manage order timelines (e.g., estimated preparation time).
* **Customer Interaction:**
  + View customer order history and reviews.
  + Respond to customer reviews (if needed).
  + Offer special promotions, discounts, or coupons to customers.
* **Analytics and Reports:**
  + Access sales data, order statistics, and customer reviews for performance monitoring.
  + Generate reports on revenue, order volume, and customer satisfaction.
* **Payment Management:**
  + View and manage payments, including transaction history and payouts.

**3. Delivery Driver (Courier)**

A **Delivery Driver** is an individual who picks up orders from restaurants and delivers them to customers.

**Permissions:**

* **Account and Profile Management:**
  + Create and manage a driver profile (name, contact information, vehicle details).
  + Update availability status (e.g., "Available", "On Delivery", "Offline").
* **Order Delivery Management:**
  + Receive order delivery assignments from the system.
  + View order details including customer name, address, and delivery instructions.
  + Track route to restaurant and then to the customer's location.
  + Update order status as "Picked Up", "In Transit", and "Delivered" once completed.
* **Notifications:**
  + Receive push notifications for new delivery requests, order status updates, and changes in delivery instructions.
  + Get alerts for any delivery issues (e.g., wrong address, delays).
* **Earnings and Payment Tracking:**
  + View earnings for completed deliveries.
  + Track daily, weekly, or monthly payouts and payment history.
* **Customer Interaction:**
  + Contact the customer via the app for delivery clarification, issues, or to confirm delivery details.

**4. Admin**

An **Admin** is a system administrator responsible for managing the overall operations of the app, including user roles, system settings, and performance monitoring.

**Permissions:**

* **User Management:**
  + Create, update, or deactivate customer, restaurant, and driver accounts.
  + Assign and modify user roles (Customer, Restaurant Owner, Delivery Driver).
  + Monitor account activity for security and compliance.
* **System Configuration:**
  + Configure system settings (e.g., payment gateways, tax rates, notification settings).
  + Manage promotional campaigns or discount settings.
* **Analytics and Reporting:**
  + Access detailed reports on overall system performance, including order volume, sales, customer satisfaction, and more.
  + Generate reports for financial accounting and operational metrics.
* **Monitoring & Troubleshooting:**
  + Monitor order flow, restaurant activity, and delivery tracking in real-time.
  + Investigate and resolve system issues, bugs, or security concerns.
* **Access Control and Security:**
  + Implement role-based access control (RBAC) policies to ensure the appropriate permissions for different user roles.
  + Review system logs for auditing purposes and security alerts.
* **Review and Feedback Management:**
  + View, approve, or moderate customer reviews and ratings.
  + Handle customer complaints and feedback escalation.

**Summary of Permissions**

| **Role** | **Account Management** | **Browse Restaurants** | **Order Management** | **Delivery Management** | **Analytics & Reporting** | **Reviews & Feedback** | **Notifications** | **Payment Management** | **System Configuration** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
| **Customer** | Create, Edit, Reset | Browse, Filter | Add to Cart, Track, | N/A | View Order History | Rate & Review | Order Updates | N/A | N/A |
|  |  |  | Cancel |  |  |  |  |  |  |
| **Restaurant Owner** | Create, Edit | View Menu | Manage Orders, Update Status | N/A | Sales, Order Stats | Respond to Reviews | Order Updates | View Transactions | Manage Menu, Discounts |
| **Delivery Driver** | Create, Edit | N/A | N/A | Accept/Complete Deliveries | View Earnings | N/A | Delivery Updates | N/A | N/A |
| **Admin** | Manage Users | N/A | View All Orders | View Delivery Status | System Performance | Moderate Reviews | System Alerts | Manage Payouts | System Settings |

Each user role has been designed to limit access and actions to only those features that are relevant to the user’s responsibilities within the system, ensuring security and operational efficiency.

4o mini

**3. Core Features**

**1.1 Account Creation & Login**

* **Sign Up**: Customers can create a new account by providing their email address, phone number, or linking their social media accounts.
* **Login**: Secure login with email/password, social logins (Google, Facebook, etc.), or biometric authentication (e.g., Face ID, Touch ID).
* **Password Recovery**: Customers can reset forgotten passwords via email or SMS.

**1.2 Browse and Search for Restaurants**

* **Location-based Search**: Customers can search for restaurants near them using their current location (via GPS) or by entering a specific address.
* **Search Filters**: Filters allow users to narrow down results based on cuisine, price range, dietary preferences (vegan, gluten-free), or ratings.
* **Restaurant Details**: Each restaurant has a detailed profile, including hours of operation, menu, ratings, delivery options, and customer reviews.

**1.3 View Menus and Customize Orders**

* **Menu Display**: Customers can view the restaurant’s full menu with prices, images, and descriptions.
* **Item Customization**: Ability to customize dishes (e.g., add extra toppings, request no onions, change portion size).
* **Add to Cart**: Users can add multiple items to their cart and easily modify quantities or remove items.

**1.4 Order Management**

* **Checkout Process**: Once items are selected, customers can review their cart and proceed to checkout.
  + **Address Input**: Customers can input or select a saved delivery address.
  + **Payment Options**: Supports multiple payment methods including credit/debit cards, digital wallets (Apple Pay, Google Pay), and cash on delivery.
  + **Order Summary**: Displays total cost, taxes, discounts (if any), and expected delivery time.
* **Order Confirmation**: After completing the checkout, customers receive a confirmation with order details and an estimated delivery time.
* **Order Tracking**: Customers can track their orders in real-time (e.g., "Order Confirmed", "Order Picked Up", "Out for Delivery").
* **Cancel Order**: Depending on restaurant policies, customers can cancel their orders within a limited time frame.

**1.5 Reviews and Ratings**

* **Rate Orders**: After receiving an order, customers can rate their experience with the restaurant and delivery driver (1 to 5 stars).
* **Write Reviews**: Customers can leave detailed feedback on their order experience, including food quality, delivery time, and customer service.
* **View Past Orders**: Customers can view their order history, including reviews and ratings given to past orders.

**1.6 Promotions and Discounts**

* **Coupon Codes**: Customers can apply promo codes for discounts on orders (e.g., first-time customer discounts, seasonal promotions).
* **Special Offers**: Restaurants may offer time-limited deals or combo offers directly on the menu.

**2. Restaurant Features**

**2.1 Restaurant Profile Management**

* **Create/Update Profile**: Restaurant owners can create or update their restaurant profile with basic information such as name, location, phone number, logo, and photos.
* **Operating Hours**: Set working hours and blackout dates (e.g., holidays, maintenance).
* **Payment Details**: Manage payout settings and payment methods for receiving earnings from completed orders.

**2.2 Menu Management**

* **Add/Update Menu Items**: Restaurants can add new items, update existing dishes, or remove unavailable items from the menu.
* **Pricing Management**: Set and update prices for individual menu items.
* **Menu Customization**: Customize dish descriptions, add ingredients, or indicate allergens.
* **Category Management**: Organize the menu into categories (e.g., Appetizers, Main Courses, Desserts).
* **Availability Control**: Set item availability (e.g., out of stock, limited availability) and schedule daily specials.

**2.3 Order Management**

* **View and Manage Orders**: Restaurant owners can view incoming orders in real-time, accept or reject them based on kitchen capacity or stock.
* **Update Order Status**: Owners can update order statuses (e.g., Preparing, Ready for Pickup, Delivered).
* **Modify Orders**: Depending on restaurant policy, owners may allow modifications to orders (e.g., special requests, ingredient substitutions).
* **Order History**: View and manage past orders, including detailed customer feedback and reviews.

**2.4 Customer Interaction**

* **Manage Reviews**: Restaurants can view, respond to, or flag inappropriate reviews from customers.
* **Promotions and Discounts**: Create custom promotions and discount codes to attract more customers, such as happy hour discounts or buy-one-get-one-free offers.

**2.5 Analytics and Reporting**

* **Order Analytics**: Track key metrics like order volume, average order value, popular menu items, and peak times.
* **Customer Feedback**: View detailed reports on customer ratings, reviews, and satisfaction levels.
* **Sales Reports**: Generate daily, weekly, or monthly sales reports, including revenue, number of orders, and payment status.

**3. Delivery Driver Features**

**3.1 Driver Account Management**

* **Create and Manage Profile**: Delivery drivers can create a profile with their contact details and vehicle information (e.g., bike, car).
* **Availability Status**: Drivers can toggle their availability status (e.g., Available, On Delivery, Offline).
* **Manage Contact Info**: Update personal information, change phone number, and manage payment preferences.

**3.2 Receive Delivery Assignments**

* **Order Notifications**: Drivers receive push notifications for new delivery assignments within their area.
* **View Delivery Details**: Drivers can view order details, including restaurant location, customer address, special delivery instructions, and the estimated delivery time.
* **Pick-Up Confirmation**: Once the driver arrives at the restaurant, they can confirm that the order has been picked up and is on its way.

**3.3 Track Deliveries and Update Status**

* **Route Navigation**: Use integrated GPS navigation to get directions to both the restaurant and customer’s location.
* **Update Delivery Status**: Drivers update the delivery status (e.g., "Picked Up", "In Transit", "Delivered").
* **Delivery Confirmation**: Upon delivery, the driver confirms that the order has been successfully delivered, and the customer can rate the driver’s service.

**3.4 Payment and Earnings Tracking**

* **Earnings Dashboard**: Track earnings for completed deliveries, including tips, bonuses, and overall revenue.
* **Payment Preferences**: Manage payment settings for receiving payouts (e.g., bank transfer, digital wallet).
* **View Past Deliveries**: Drivers can access a history of completed deliveries and earnings.

**4. Admin Features**

**4.1 User Management**

* **Manage User Roles**: Admins can create, update, and deactivate user accounts, assigning roles to customers, restaurant owners, and drivers.
* **Monitor Activity**: Track user activity (e.g., order history, login attempts) and investigate any suspicious or unauthorized activity.

**4.2 System Configuration**

* **Payment Configuration**: Configure supported payment gateways (e.g., Stripe, PayPal) and set tax rates or commission fees.
* **Promo Code Management**: Create, activate, and deactivate promotional codes for customer discounts.
* **Notifications Management**: Customize and manage the notification system for real-time updates (order status, promotional offers).

**4.3 Reports and Analytics**

* **System Analytics**: Access detailed reports on system-wide performance, including order volume, user growth, sales data, and customer satisfaction.
* **Financial Reports**: Generate financial reports for auditing and accounting purposes, such as restaurant earnings and delivery driver payouts.
* **User Feedback**: View overall customer ratings and reviews to ensure quality control.

**4.4 Security and Monitoring**

* **Activity Logs**: Admins can access logs of all user activities within the system for auditing and compliance purposes.
* **Monitor Orders**: Track order flow across the platform, including potential delays or issues.
* **Moderation Tools**: Approve or remove inappropriate content (e.g., offensive reviews, inappropriate comments).

**Summary of Core Features**

**1.1 Account Creation & Login**

* **Sign Up**: Customers can create a new account by providing their email address, phone number, or linking their social media accounts.
* **Login**: Secure login with email/password, social logins (Google, Facebook, etc.), or biometric authentication (e.g., Face ID, Touch ID).
* **Password Recovery**: Customers can reset forgotten passwords via email or SMS.

**1.2 Browse and Search for Restaurants**

* **Location-based Search**: Customers can search for restaurants near them using their current location (via GPS) or by entering a specific address.
* **Search Filters**: Filters allow users to narrow down results based on cuisine, price range, dietary preferences (vegan, gluten-free), or ratings.
* **Restaurant Details**: Each restaurant has a detailed profile, including hours of operation, menu, ratings, delivery options, and customer reviews.

**1.3 View Menus and Customize Orders**

* **Menu Display**: Customers can view the restaurant’s full menu with prices, images, and descriptions.
* **Item Customization**: Ability to customize dishes (e.g., add extra toppings, request no onions, change portion size).
* **Add to Cart**: Users can add multiple items to their cart and easily modify quantities or remove items.

**1.4 Order Management**

* **Checkout Process**: Once items are selected, customers can review their cart and proceed to checkout.
  + **Address Input**: Customers can input or select a saved delivery address.
  + **Payment Options**: Supports multiple payment methods including credit/debit cards, digital wallets (Apple Pay, Google Pay), and cash on delivery.
  + **Order Summary**: Displays total cost, taxes, discounts (if any), and expected delivery time.
* **Order Confirmation**: After completing the checkout, customers receive a confirmation with order details and an estimated delivery time.
* **Order Tracking**: Customers can track their orders in real-time (e.g., "Order Confirmed", "Order Picked Up", "Out for Delivery").
* **Cancel Order**: Depending on restaurant policies, customers can cancel their orders within a limited time frame.

**1.5 Reviews and Ratings**

* **Rate Orders**: After receiving an order, customers can rate their experience with the restaurant and delivery driver (1 to 5 stars).
* **Write Reviews**: Customers can leave detailed feedback on their order experience, including food quality, delivery time, and customer service.
* **View Past Orders**: Customers can view their order history, including reviews and ratings given to past orders.

**1.6 Promotions and Discounts**

* **Coupon Codes**: Customers can apply promo codes for discounts on orders (e.g., first-time customer discounts, seasonal promotions).
* **Special Offers**: Restaurants may offer time-limited deals or combo offers directly on the menu.

**2. Restaurant Features**

**2.1 Restaurant Profile Management**

* **Create/Update Profile**: Restaurant owners can create or update their restaurant profile with basic information such as name, location, phone number, logo, and photos.
* **Operating Hours**: Set working hours and blackout dates (e.g., holidays, maintenance).
* **Payment Details**: Manage payout settings and payment methods for receiving earnings from completed orders.

**2.2 Menu Management**

* **Add/Update Menu Items**: Restaurants can add new items, update existing dishes, or remove unavailable items from the menu.
* **Pricing Management**: Set and update prices for individual menu items.
* **Menu Customization**: Customize dish descriptions, add ingredients, or indicate allergens.
* **Category Management**: Organize the menu into categories (e.g., Appetizers, Main Courses, Desserts).
* **Availability Control**: Set item availability (e.g., out of stock, limited availability) and schedule daily specials.

**2.3 Order Management**

* **View and Manage Orders**: Restaurant owners can view incoming orders in real-time, accept or reject them based on kitchen capacity or stock.
* **Update Order Status**: Owners can update order statuses (e.g., Preparing, Ready for Pickup, Delivered).
* **Modify Orders**: Depending on restaurant policy, owners may allow modifications to orders (e.g., special requests, ingredient substitutions).
* **Order History**: View and manage past orders, including detailed customer feedback and reviews.

**2.4 Customer Interaction**

* **Manage Reviews**: Restaurants can view, respond to, or flag inappropriate reviews from customers.
* **Promotions and Discounts**: Create custom promotions and discount codes to attract more customers, such as happy hour discounts or buy-one-get-one-free offers.

**2.5 Analytics and Reporting**

* **Order Analytics**: Track key metrics like order volume, average order value, popular menu items, and peak times.
* **Customer Feedback**: View detailed reports on customer ratings, reviews, and satisfaction levels.
* **Sales Reports**: Generate daily, weekly, or monthly sales reports, including revenue, number of orders, and payment status.

**3. Delivery Driver Features**

**3.1 Driver Account Management**

* **Create and Manage Profile**: Delivery drivers can create a profile with their contact details and vehicle information (e.g., bike, car).
* **Availability Status**: Drivers can toggle their availability status (e.g., Available, On Delivery, Offline).
* **Manage Contact Info**: Update personal information, change phone number, and manage payment preferences.

**3.2 Receive Delivery Assignments**

* **Order Notifications**: Drivers receive push notifications for new delivery assignments within their area.
* **View Delivery Details**: Drivers can view order details, including restaurant location, customer address, special delivery instructions, and the estimated delivery time.
* **Pick-Up Confirmation**: Once the driver arrives at the restaurant, they can confirm that the order has been picked up and is on its way.

**3.3 Track Deliveries and Update Status**

* **Route Navigation**: Use integrated GPS navigation to get directions to both the restaurant and customer’s location.
* **Update Delivery Status**: Drivers update the delivery status (e.g., "Picked Up", "In Transit", "Delivered").
* **Delivery Confirmation**: Upon delivery, the driver confirms that the order has been successfully delivered, and the customer can rate the driver’s service.

**3.4 Payment and Earnings Tracking**

* **Earnings Dashboard**: Track earnings for completed deliveries, including tips, bonuses, and overall revenue.
* **Payment Preferences**: Manage payment settings for receiving payouts (e.g., bank transfer, digital wallet).
* **View Past Deliveries**: Drivers can access a history of completed deliveries and earnings.

**4. Admin Features**

**4.1 User Management**

* **Manage User Roles**: Admins can create, update, and deactivate user accounts, assigning roles to customers, restaurant owners, and drivers.
* **Monitor Activity**: Track user activity (e.g., order history, login attempts) and investigate any suspicious or unauthorized activity.

**4.2 System Configuration**

* **Payment Configuration**: Configure supported payment gateways (e.g., Stripe, PayPal) and set tax rates or commission fees.
* **Promo Code Management**: Create, activate, and deactivate promotional codes for customer discounts.
* **Notifications Management**: Customize and manage the notification system for real-time updates (order status, promotional offers).

**4.3 Reports and Analytics**

* **System Analytics**: Access detailed reports on system-wide performance, including order volume, user growth, sales data, and customer satisfaction.
* **Financial Reports**: Generate financial reports for auditing and accounting purposes, such as restaurant earnings and delivery driver payouts.
* **User Feedback**: View overall customer ratings and reviews to ensure quality control.

**4.4 Security and Monitoring**

* **Activity Logs**: Admins can access logs of all user activities within the system for auditing and compliance purposes.
* **Monitor Orders**: Track order flow across the platform, including potential delays or issues.
* **Moderation Tools**: Approve or remove inappropriate content (e.g., offensive reviews, inappropriate comments).

**Summary of Core Features**

* **Customer Features**: Account creation, restaurant browsing, menu viewing, order management, real-time tracking, reviews and ratings, payment options, and promotions.
* **Restaurant Features**: Menu management, order processing, customer interaction, promotions, sales analytics, and review management.
* **Delivery Driver Features**: Delivery assignment notifications, route navigation, order status updates, and payment tracking.
* **Admin Features**: User management, system configuration, reporting and analytics, activity monitoring, and security management.

**3.1. User Authentication**

* **Sign-up/Login**: Users can sign up with email, phone number, or social media accounts (Google, Facebook).
* **Forgot Password**: Users can reset their passwords via email or SMS.

**3.2. Restaurant Discovery**

* **Search**: Users can search for restaurants by cuisine type, location, and rating.
* **Filters**: Narrow down results by price range, rating, dietary restrictions (e.g., vegetarian, vegan, gluten-free).
* **Menu View**: Browse restaurant menus, view item descriptions, and prices.

**3.3. Ordering System**

* **Order Customization**: Users can add special instructions, customize items (e.g., no onions, extra cheese), and add sides/desserts.
* **Order Confirmation**: Once the order is placed, users receive a confirmation and estimated delivery time.

**3.4. Delivery Tracking**

* **Real-time Updates**: Track the status of the order (order placed, food being prepared, food on its way, delivered).
* **GPS Integration**: Customers can view the live location of the delivery driver on a map.

**3.5. Payment Integration**

* **Payment Methods**: Credit/debit cards, mobile wallets (Google Pay, Apple Pay), and cash on delivery.
* **Payment Processing**: Secure payment gateways like Stripe or PayPal are integrated for seamless transactions.

**4. System Architecture**

**High-Level Architecture Overview**

The system architecture follows a **client-server** model, where clients (mobile apps and web portals) interact with backend services, which in turn interact with databases and external services.

**Key Components:**

* **Frontend (Mobile & Web Clients)**: User-facing applications (iOS/Android) and the web admin portal.
* **Backend (API Server)**: RESTful API services responsible for handling business logic, order processing, and managing interactions between different user roles.
* **Database Layer**: Stores user data, order data, restaurant information, and transaction history.
* **Third-Party Services**: External integrations for payment processing, location services, SMS/email notifications, and delivery routing.
* **Notification System**: Push notifications for order updates, promotions, and alerts.
* **Admin Dashboard**: A web interface for system administrators to monitor the entire platform, manage users, and generate reports.

**2. Frontend**

The frontend consists of mobile apps (iOS and Android) and a web portal for admins. Each part is designed to ensure an intuitive, user-friendly experience.

* **Mobile Apps (iOS/Android)**:
  + Built using **React Native** (cross-platform) or **Swift** (for iOS) and **Kotlin** (for Android).
  + Responsible for managing user interactions, displaying restaurant data, processing orders, and enabling real-time order tracking.
  + Integrated with the backend via REST APIs for order management, user authentication, and notifications.
  + **Push notifications**: Managed using **Firebase Cloud Messaging (FCM)** for order status updates and promotions.
* **Web Admin Portal**:
  + Built with **React.js** or **Vue.js** to provide a responsive and dynamic user interface for system administrators.
  + Admins can access data on orders, user activity, restaurant performance, and other analytics.

**3. Backend (API Server)**

The backend is designed as a **microservices architecture**, with each service handling a specific aspect of the platform’s functionality. This allows for scalability and independent management of services.

**3.1 Microservices**

Each core functionality of Snack Squad (ordering, payments, notifications, etc.) is handled by independent services that communicate via API calls.

* **User Service**: Handles user registration, authentication (using **JWT** tokens), and profile management.
  + Technologies: **Node.js** with **Express.js** or **NestJS**.
* **Restaurant Service**: Manages restaurant profiles, menus, orders, and order status.
  + Technologies: **Node.js** with **Express.js** or **NestJS**.
* **Order Management Service**: Processes customer orders, handles order status transitions, and coordinates with the restaurant and delivery services.
  + Technologies: **Node.js** or **Go** for high concurrency.
* **Payment Service**: Integrates with external payment gateways (e.g., **Stripe**, **PayPal**) to process payments and handle transactions.
  + Technologies: **Node.js** or **Python**.
* **Delivery Service**: Manages delivery assignments, driver statuses, and route optimizations.
  + Technologies: **Node.js** or **Python**.
* **Notification Service**: Sends notifications to users, restaurants, and drivers (via **Push Notifications**, **Email**, or **SMS**).
  + Technologies: **Firebase Cloud Messaging (FCM)** for mobile notifications, **Twilio** for SMS, and **SendGrid** for emails.
* **Analytics Service**: Collects and analyzes data on orders, customer behavior, and system performance.
  + Technologies: **Apache Kafka** for event streaming, **Elasticsearch** for search and analytics.

**3.2 API Gateway**

* **API Gateway**: A single entry point for all client requests. It handles routing to appropriate microservices, authentication, load balancing, rate limiting, and caching.
  + Technologies: **Kong** or **NGINX** for reverse proxy and API management.
  + Acts as a **Security Layer**: Implements security features like rate-limiting, request validation, and JWT authentication.

**4. Database Layer**

**4.1 Relational Database (PostgreSQL)**

* **PostgreSQL** is used as the primary database for storing structured data:
  + **User Data**: Customer profiles, restaurant profiles, driver information, and admin details.
  + **Order Data**: Order history, order status, payment transactions, and delivery status.
  + **Restaurant Data**: Menus, restaurant profiles, reviews, and ratings.

**Database Schema Overview**:

* **Users** (Customers, Restaurant Owners, Delivery Drivers)
* **Restaurants** (Profiles, Menus, Orders)
* **Orders** (Order Details, Status, Payment)
* **Reviews & Ratings**

**4.2 NoSQL Database (MongoDB)**

* **MongoDB** is used for storing unstructured or semi-structured data, such as:
  + **Real-time Order Status**: For efficient tracking of live order updates and driver status.
  + **Push Notification Logs**: To store notification history for auditing purposes.

**4.3 Caching Layer (Redis)**

* **Redis** is used to cache frequently accessed data (e.g., restaurant menus, user profiles) to improve performance and reduce database load.

**5. External Integrations**

**5.1 Payment Gateway Integration**

* **Stripe** or **PayPal** for secure payment processing, supporting credit cards, debit cards, and digital wallets (Apple Pay, Google Pay).
* Payment services handle fraud detection, refunds, and payment reconciliation.

**5.2 Location and Maps Integration**

* **Google Maps API** or **Mapbox** for:
  + Location-based restaurant search.
  + Real-time delivery tracking using GPS.
  + Route optimization for delivery drivers.

**5.3 Third-Party Notification Services**

* **Twilio** for SMS notifications (e.g., order confirmations, delivery status).
* **SendGrid** for email notifications (e.g., receipts, promotional emails).

**6. Real-Time Communication**

**6.1 WebSockets**

* **WebSockets** are used for **real-time communication** between the mobile apps, restaurants, and delivery drivers.
  + **Order Status Updates**: Customers can receive live updates on their order status (e.g., Preparing, Out for Delivery).
  + **Driver Location Updates**: Drivers can share their real-time location with the customer for tracking.

**6.2 Push Notifications**

* **Firebase Cloud Messaging (FCM)**: Used for sending push notifications to customers and delivery drivers (e.g., order updates, promotions).
* **Twilio** or **SendGrid**: For sending SMS and email notifications when required.

**7. Admin Portal**

The **Admin Dashboard** is a **web-based** portal used by administrators to manage the system, monitor performance, and generate reports.

* **Features**:
  + **User Management**: Add, edit, or deactivate customer, restaurant, and driver accounts.
  + **Analytics & Reporting**: View system-wide metrics, such as orders processed, sales data, customer satisfaction, and restaurant performance.
  + **System Monitoring**: Monitor the status of API services, payment processing, and delivery tracking.
  + **Security**: Admins can audit logs, reset passwords, and investigate suspicious activity.

**8. Security and Compliance**

**8.1 Authentication and Authorization**

* **JWT (JSON Web Tokens)**: Used for secure user authentication and session management.
* **OAuth2**: Supports third-party logins (Google, Facebook, etc.).

**8.2 Data Encryption**

* **TLS/SSL**: All data exchanged between clients and servers is encrypted using **TLS** for secure communication.
* **AES Encryption**: Sensitive data, such as passwords and payment details, is stored using **AES-256** encryption.

**8.3 Rate Limiting & DDoS Protection**

* **API Rate Limiting**: To prevent abuse, API requests are rate-limited based on user roles and request type.
* **Cloudflare or AWS Shield**: For DDoS protection and website security.

**9. Deployment & Scalability**

**9.1 Containerization and Orchestration**

* **Docker**: Used for containerizing backend services for easy deployment and scalability.
* **Kubernetes**: For orchestrating containers, ensuring high availability, auto-scaling, and efficient resource management.

**9.2 Cloud Infrastructure**

* **AWS, Google Cloud, or Azure**: Hosting services for compute (EC2, Kubernetes), storage (S3), and databases (RDS, DynamoDB).
* **Content Delivery Network (CDN)**: **Cloudflare** or **AWS CloudFront** for faster content delivery and load balancing.

**4.1. Frontend**

The app is designed using modern mobile frameworks to ensure smooth user experience and responsiveness.

* **Mobile App**:
  + **Platform**: React Native or Flutter.
  + **Languages**: JavaScript (React Native), Dart (Flutter).
  + **UI Framework**: Material UI or custom design based on brand guidelines.
  + **State Management**: Redux or Provider (for Flutter).

**4.2. Backend**

* **Platform**: Node.js with Express (for RESTful APIs).
* **Database**:
  + **Primary Database**: MongoDB (NoSQL, for flexible data models like users, restaurants, orders).
  + **Relational Database**: PostgreSQL (for payment records, ratings).
* **Authentication**: Firebase Authentication or JWT (JSON Web Tokens) for secure login.
* **Real-time Communication**: Socket.io for real-time order updates and chat functionality.

**4.3. APIs**

The backend exposes a series of RESTful APIs for communication between the frontend (mobile app) and backend.

**Common Endpoints**

1. **User Authentication**
   * POST /api/auth/register
   * POST /api/auth/login
   * POST /api/auth/forgot-password
2. **Restaurant API**
   * GET /api/restaurants - List all restaurants.
   * GET /api/restaurants/{id} - Get restaurant details.
   * POST /api/restaurants/{id}/menu - Add/edit/remove menu items.
3. **Order API**
   * POST /api/orders - Place a new order.
   * GET /api/orders/{id} - Get order details.
   * PATCH /api/orders/{id} - Update order status (restaurant, driver, or customer).
   * GET /api/delivery/{orderId} - Get current delivery status and location.

**5. Data Models**

**5.1. User Model**

**User Models**

**1.1 User**

The User model represents both customers, restaurant owners, and delivery drivers, as they all share certain common attributes like authentication details, contact information, and role-specific data.

**Attributes**:

* user\_id: Unique identifier for the user (Primary Key).
* email: User’s email address (unique).
* phone\_number: Contact phone number (optional).
* password\_hash: Hashed password (for security purposes).
* first\_name: User’s first name.
* last\_name: User’s last name.
* role: Role of the user (e.g., customer, restaurant\_owner, delivery\_driver, admin).
* profile\_picture: URL of the user's profile picture (optional).
* status: Account status (e.g., active, inactive, suspended).
* created\_at: Date and time the account was created.
* updated\_at: Date and time the account was last updated.

**1.2 Customer**

A specific type of User, the Customer model extends the User model and contains additional attributes related to a customer’s preferences and history.

**Attributes**:

* user\_id: Foreign Key (references User).
* preferred\_address: Default delivery address (or list of addresses).
* payment\_methods: List of saved payment methods (credit card, wallet, etc.).
* order\_history: List of completed orders.
* wishlist: List of restaurants or items that the user frequently orders or has marked as favorites.
* notifications\_enabled: Boolean flag indicating whether the user wants to receive notifications.
* average\_rating: Average rating the user has given to restaurants and drivers.

**1.3 Restaurant Owner**

A RestaurantOwner model, extending the User model, stores restaurant-specific details.

**Attributes**:

* user\_id: Foreign Key (references User).
* restaurant\_name: Name of the restaurant.
* restaurant\_phone: Restaurant's contact phone number.
* restaurant\_address: Restaurant's physical address.
* restaurant\_logo: URL of the restaurant’s logo.
* operating\_hours: Daily opening and closing times.
* restaurant\_type: Type of cuisine or food served (e.g., "Italian", "Fast Food").
* status: Current operational status of the restaurant (e.g., open, closed, suspended).
* menu: List of menu items (reference to MenuItem model).
* average\_rating: Average rating given by customers.

**1.4 Delivery Driver**

A DeliveryDriver model extends the User model and stores details related to the driver’s profile, vehicle, and delivery history.

**Attributes**:

* user\_id: Foreign Key (references User).
* vehicle\_type: Type of vehicle used by the driver (e.g., car, bike, scooter).
* vehicle\_number: Vehicle registration number.
* rating: Average rating given to the driver.
* current\_location: Geolocation (latitude, longitude) of the driver for real-time tracking.
* status: Delivery status of the driver (e.g., available, on\_delivery, offline).
* order\_history: List of orders delivered by the driver.
* earnings: Total earnings accumulated by the driver.

**2. Restaurant Models**

**2.1 Restaurant**

The Restaurant model represents the restaurant entity and is associated with many other data models, such as the MenuItem, Order, and Review models.

**Attributes**:

* restaurant\_id: Unique identifier for the restaurant (Primary Key).
* restaurant\_name: Name of the restaurant.
* restaurant\_address: Physical location of the restaurant.
* contact\_number: Phone number for the restaurant.
* logo\_url: URL of the restaurant’s logo.
* category: Cuisine category (e.g., Fast Food, Fine Dining, Vegan).
* average\_rating: Average rating of the restaurant based on customer reviews.
* delivery\_available: Boolean indicating if the restaurant offers delivery.
* pickup\_available: Boolean indicating if the restaurant offers pickup.
* menu: List of MenuItem references.

**2.2 MenuItem**

The MenuItem model represents individual food items on a restaurant’s menu.

**Attributes**:

* item\_id: Unique identifier for the menu item (Primary Key).
* restaurant\_id: Foreign Key (references Restaurant).
* name: Name of the menu item.
* description: Detailed description of the item.
* price: Price of the item.
* category: Category of the item (e.g., Appetizer, Main Course, Dessert).
* available: Boolean indicating if the item is available for ordering.
* image\_url: URL for an image of the item.
* tags: List of tags (e.g., vegan, gluten-free, spicy).

**3. Order Models**

**3.1 Order**

The Order model tracks the status of orders placed by customers and processed by restaurants.

**Attributes**:

* order\_id: Unique identifier for the order (Primary Key).
* customer\_id: Foreign Key (references Customer).
* restaurant\_id: Foreign Key (references Restaurant).
* delivery\_driver\_id: Foreign Key (references DeliveryDriver, nullable if pickup).
* order\_status: Status of the order (e.g., pending, preparing, out\_for\_delivery, delivered).
* order\_total: Total amount of the order (including taxes and tips).
* payment\_status: Status of the payment (e.g., paid, pending, failed).
* payment\_method: Payment method used (e.g., credit\_card, wallet).
* order\_items: List of items ordered (references OrderItem).
* delivery\_address: Delivery address for the order.
* created\_at: Timestamp of when the order was placed.
* updated\_at: Timestamp of the last update to the order.

**3.2 OrderItem**

The OrderItem model stores details about individual items in an order.

**Attributes**:

* order\_item\_id: Unique identifier for the order item (Primary Key).
* order\_id: Foreign Key (references Order).
* menu\_item\_id: Foreign Key (references MenuItem).
* quantity: Number of items ordered.
* customizations: Customizations applied to the menu item (e.g., extra toppings, special requests).
* price: Price of the item at the time of ordering (in case of promotions or discounts).

**3.3 OrderHistory**

The OrderHistory model logs changes in the status of an order.

**Attributes**:

* history\_id: Unique identifier for the status update (Primary Key).
* order\_id: Foreign Key (references Order).
* status: The status of the order (e.g., pending, preparing, out\_for\_delivery, delivered).
* timestamp: Timestamp of when the status was updated.
* notes: Additional notes related to the status change (e.g., delay, issue with the order).

**4. Payment Models**

**4.1 Payment**

The Payment model tracks the payment details associated with an order.

**Attributes**:

* payment\_id: Unique identifier for the payment (Primary Key).
* order\_id: Foreign Key (references Order).
* amount: Total amount of the payment (including taxes, delivery fees, and tips).
* payment\_method: Payment method used (e.g., credit\_card, wallet, cash).
* payment\_status: Status of the payment (e.g., success, failed, pending).
* payment\_date: Timestamp when the payment was made.
* payment\_reference: Payment gateway reference or transaction ID.

**4.2 Refund**

The Refund model handles any refunds issued for orders.

**Attributes**:

* refund\_id: Unique identifier for the refund (Primary Key).
* payment\_id: Foreign Key (references Payment).
* refund\_amount: Amount refunded.
* refund\_reason: Reason for the refund (e.g., order canceled, payment error).
* refund\_status: Status of the refund (e.g., processed, pending, failed).
* refund\_date: Timestamp when the refund was issued.

**5. Review Models**

**5.1 Review**

The Review model represents customer feedback on orders, restaurants, and delivery drivers.

**Attributes**:

* review\_id: Unique identifier for the review (Primary Key).
* order\_id: Foreign Key (references Order).
* customer\_id: Foreign Key (references Customer).
* restaurant\_id: Foreign Key (references Restaurant).
* delivery\_driver\_id: Foreign Key (references DeliveryDriver, nullable).
* rating: Numeric rating (1 to 5 stars).
* review\_text: Optional text feedback from the customer.
* created\_at: Timestamp when the review was created.

**6. Notification Models**

json

Copy code

{

"userId": "123456",

"name": "John Doe",

"email": "john.doe@example.com",

"phone": "+1234567890",

"role": "customer",

"address": "123 Main St, City, Country",

"orders": [ "orderId1", "orderId2" ]

}

**5.2. Restaurant Model**

json

Copy code

{

"restaurantId": "abc123",

"name": "Pizza Place",

"address": "456 Elm St, City, Country",

"cuisine": "Italian",

"rating": 4.5,

"menu": [ "itemId1", "itemId2" ]

}

**5.3. Order Model**

json

Copy code

{

"orderId": "order123",

"userId": "123456",

"restaurantId": "abc123",

"items": [ "itemId1", "itemId2" ],

"status": "in-progress",

"deliveryTime": "2024-11-14T12:30:00Z",

"totalPrice": 35.99

}

**5.4. Rating & Review Model**

json

Copy code

{

"ratingId": "rating789",

"userId": "123456",

"restaurantId": "abc123",

"foodId": "itemId1",

"rating": 5,

"review": "Amazing pizza, would order again!"

}

**6. Mobile App Flow**

**1 App Initialization**

* Upon launch, the app displays a **splash screen** featuring the **Snack Squad logo** and tagline.
* **Loading phase**: The app checks for any necessary updates and initializes the user session (e.g., checking for authenticated users or fetching saved preferences).

**1.2 Authentication Screen**

* **Login** or **Sign Up** options:
  + **Login**: Users can log in using their email/password or through social logins (Google, Facebook).
  + **Sign Up**: New users can create an account by providing their basic details (email, phone number, password).
  + **Forgot Password**: Option to reset password via email or SMS.

**2. Home Screen**

**2.1 User Greeting & Personalized Content**

* Once logged in, the app greets the user by name (e.g., “Welcome, John!”).
* The **Home Screen** displays personalized content based on the user's past orders, preferences, and location:
  + **Location-based Restaurant Suggestions**: Restaurants near the user’s location (via GPS).
  + **Promotions**: Special deals and offers, such as “Free delivery on your first order” or “10% off your next order.”
  + **Popular Categories**: Categories such as Pizza, Burgers, Vegan, etc.
  + **Quick Reorder**: Option to reorder items from previous orders.

**2.2 Search Bar**

* A search bar is prominently displayed at the top, allowing users to search for restaurants by name, cuisine, or specific dishes.
* **Filter options**:
  + **Cuisine Type**: Choose from categories like Italian, Indian, Mexican, etc.
  + **Price Range**: Filters to show restaurants within the desired budget.
  + **Rating**: Filter by customer reviews (e.g., 4 stars and above).
  + **Dietary Preferences**: Filters like Vegan, Gluten-Free, Halal, etc.

**3. Restaurant and Menu Browsing**

**3.1 Restaurant Listing**

* The user can scroll through a list of nearby restaurants or use the **search/filter options** to find specific restaurants.
* Each restaurant entry displays:
  + Restaurant name, image, rating, and delivery time.
  + A “View Menu” button to browse the restaurant’s offerings.

**3.2 Restaurant Details**

* Once a restaurant is selected, the app shows the **restaurant profile**, including:
  + **Restaurant Info**: Name, logo, description, location, hours of operation, contact details.
  + **Menu Categories**: Organized by Appetizers, Main Courses, Desserts, etc.
  + **Menu Items**: Each item has:
    - Name, description, price, and an image.
    - A customizable option (e.g., add extra toppings, remove an ingredient).
    - An “Add to Cart” button.

**3.3 Menu Customization**

* Users can customize menu items (e.g., extra toppings, special requests, or choosing portion size).
* When an item is customized, it’s displayed with the user’s selections and price adjustments.

**4. Cart & Checkout Flow**

**4.1 Cart Screen**

* The user can view all the items added to the cart with:
  + **Item details**: Name, quantity, price, and any customizations.
  + **Total Price**: Includes subtotal, taxes, and delivery fees.
  + **Edit Cart**: Ability to change item quantity or remove items.
  + **Delivery Address**: Option to enter or select a saved address.
* **Continue to Checkout**: Button that leads to the checkout page.

**4.2 Checkout Screen**

* The user confirms the following details:
  + **Delivery Address**: The user can either use their default saved address or enter a new one.
  + **Payment Method**: The user selects a payment method (e.g., Credit Card, Wallet, Cash on Delivery).
    - New users are prompted to enter payment details if they haven't added one previously.
    - Users can view payment options such as **Apple Pay**, **Google Pay**, or **Credit/Debit Card**.
  + **Order Summary**: A final confirmation of the order details, including:
    - List of items ordered, quantities, prices, and total cost.
    - Estimated delivery time (based on restaurant and driver availability).
* **Apply Promo Code**: Option to enter a promotional code if applicable (e.g., 10%OFF, FREESHIP).
* **Confirm Order**: Button to finalize and place the order.

**5. Order Confirmation & Tracking**

**5.1 Order Confirmation Screen**

* After successfully placing an order, the user sees an **Order Confirmation** screen with:
  + **Order ID**.
  + **Restaurant Info**: Restaurant name and contact information.
  + **Order Summary**: Items ordered, total cost, and payment method.
  + **Estimated Delivery Time**: The expected delivery time window.
* **Track My Order**: The user is provided with a “Track My Order” button that takes them to the real-time order tracking screen.

**5.2 Real-Time Order Tracking**

* The user can track the status of the order in real-time:
  + **Order Status**: Updates include Order Received, Preparing, Out for Delivery, and Delivered.
  + **Delivery Driver Location**: A map showing the driver’s location and route in real-time.
  + **Push Notifications**: The user will receive notifications for status changes (e.g., when the order is out for delivery or when the driver is nearby).

**6. Order Completion & Feedback**

**6.1 Order Delivery Confirmation**

* Once the order is delivered, the user is prompted to confirm receipt of the order.
* **Delivery Confirmation**: Button to confirm the order has been delivered successfully.

**6.2 Rating and Review**

* After delivery, the user is encouraged to rate the restaurant and the delivery experience:
  + **Restaurant Rating**: 1-5 stars.
  + **Driver Rating**: 1-5 stars (separate from restaurant rating).
  + **Review Text**: Optional feedback on food quality, delivery time, and customer service.
* **Submit Review**: Button to submit the review.

**7. User Profile & Settings**

**7.1 Profile Management**

* Accessed via the **Profile Icon** in the bottom navigation bar:
  + **View/Edit Profile**: Update personal details, email, phone number, and password.
  + **Saved Addresses**: Add, edit, or delete saved delivery addresses.
  + **Payment Methods**: Add or manage saved payment options.

**7.2 Order History**

* A list of previous orders, with options to **Reorder** items or view past reviews.

**7.3 Settings**

* Access the settings menu to manage:
  + **Push Notifications**: Toggle notifications for order updates, promotions, and offers.
  + **Language & Currency**: Change app language or currency.
  + **Help & Support**: Contact customer support, report issues, or view FAQs.

**7.4 Log Out**

* Option to log out of the account.

**8. Push Notifications & Alerts**

* **Order Updates**: Real-time notifications about the order status (e.g., "Your order is out for delivery").
* **Promotions**: Notifications about special offers, new restaurant listings, and discounts.
* **Driver Updates**: Alerts for the user when their delivery driver is near or when the order has been delivered.

**9. App Flow Summary**

1. **Launch & Authentication**: User logs in or signs up.
2. **Browse Restaurants & Menu**: Search for restaurants, view menus, and customize orders.
3. **Cart & Checkout**: Review order, confirm delivery address, and select payment method.
4. **Order Tracking**: Track order status and delivery progress in real-time.
5. **Order Completion**: Confirm delivery, rate the restaurant and driver, and provide feedback.
6. **Profile & Settings**: Manage profile, addresses, payment methods, and notifications.

**6.1. Onboarding**

1. **Sign-up/Login**: The app will ask users to sign up or log in on the first launch.
2. **Location Access**: Request permission to access the device’s location for restaurant discovery.

**6.2. Main Navigation**

1. **Home Screen**: Display a list of nearby restaurants, or search bar for user queries.
2. **Restaurant Detail**: Show restaurant info, menu, and order button.
3. **Cart**: View and edit selected items, proceed to checkout.
4. **Order Confirmation & Tracking**: After checkout, show order confirmation with delivery tracking.

**6.3. User Profile**

* Update personal details, payment methods, address, and view past orders.

1. **Security Considerations**

**Data Encryption**

**1.1 Transport Layer Security (TLS)**

* **Use HTTPS** for all communication between the app and backend services. This ensures that all data transmitted (such as personal details, payment information, and location data) is encrypted using **TLS** (Transport Layer Security).
* Enforce **TLS 1.2 or higher** to ensure strong encryption standards are used during communication.

**1.2 Encryption at Rest**

* All sensitive data (e.g., passwords, payment details) stored in the database should be encrypted at rest.
* Use **AES-256** encryption or similar industry-standard encryption algorithms to protect sensitive user data stored in databases or on disk.

**1.3 End-to-End Encryption (E2EE)**

* For real-time communications (such as chat between customers and restaurants or drivers), **End-to-End Encryption** (E2EE) should be used to ensure that data cannot be intercepted or read by unauthorized parties.

**2. Authentication and Authorization**

**2.1 Secure User Authentication**

* **Password Hashing**: Always store passwords as securely hashed values (e.g., using **bcrypt**, **PBKDF2**, or **Argon2**). Never store plain-text passwords.
* **Multi-Factor Authentication (MFA)**: Implement **two-factor authentication (2FA)** for users, especially for administrative and high-privilege accounts, to provide an additional layer of security.
* **OAuth 2.0**: Use **OAuth 2.0** for third-party logins (e.g., Google, Facebook) and secure token-based authentication for user sessions.
* **JWT (JSON Web Tokens)**: For secure user authentication and stateless sessions, use **JWT** to generate short-lived access tokens and store them securely on the client-side (e.g., using **Secure HttpOnly Cookies**).

**2.2 Role-Based Access Control (RBAC)**

* Implement **role-based access control** to restrict access to sensitive functionality based on user roles (e.g., customer, restaurant\_owner, delivery\_driver, admin).
* Enforce permissions to ensure that users can only access resources or perform actions that are appropriate for their roles.

**2.3 Session Management**

* Ensure that session tokens (JWT or cookies) are short-lived and have an expiration time (e.g., 15 minutes for access tokens).
* Provide mechanisms for users to log out or expire their session tokens, especially if their account is compromised or they log out from multiple devices.
* Use **refresh tokens** to allow users to obtain new access tokens without requiring re-authentication.

**3. Data Protection and Privacy**

**3.1 User Data Minimization**

* Collect only the necessary data required to perform the service (e.g., name, delivery address, payment details).
* Provide clear and concise **privacy policies** that inform users about what data is collected, how it’s used, and how long it’s stored.

**3.2 Anonymization and Pseudonymization**

* For privacy reasons, ensure that **sensitive data** (e.g., payment details, addresses) is anonymized or pseudonymized when not needed for immediate processing.
* Avoid storing sensitive payment data directly in your database. Use secure, compliant third-party payment processors (e.g., **Stripe**, **PayPal**) to handle payment transactions.

**3.3 GDPR and CCPA Compliance**

* Ensure that the platform complies with **GDPR** (General Data Protection Regulation) for EU users and **CCPA** (California Consumer Privacy Act) for users in California, ensuring that users have control over their personal data, including the ability to:
  + Access, update, or delete their personal data.
  + Consent to data collection and processing.
  + Opt-out of marketing or data-sharing practices.

**4. Payment Security**

**4.1 PCI-DSS Compliance**

* **PCI-DSS** (Payment Card Industry Data Security Standard) compliance is mandatory for handling credit card information. Since **Snack Squad** may process payments, ensure that all payment transactions comply with PCI-DSS standards, especially when handling cardholder data.
* Use a **third-party payment processor** (e.g., **Stripe**, **PayPal**) to handle sensitive payment data. Do not store or process card details directly on your systems to minimize the risk of data breaches.

**4.2 Tokenization**

* Use **tokenization** to replace sensitive payment information (like credit card numbers) with a non-sensitive, randomly generated string of characters (the token). The token can be used to make payments without exposing sensitive data.

**4.3 Fraud Detection**

* Implement **real-time fraud detection** and **transaction monitoring** systems that detect anomalous behavior, such as unusually large transactions, multiple failed payment attempts, or unusual login locations.
* **Machine learning algorithms** can be applied to predict and prevent fraud.

**5. API Security**

**5.1 API Authentication & Authorization**

* Secure all APIs with proper **authentication and authorization** mechanisms. Use **OAuth 2.0** or **API keys** for service-to-service communication, and **JWTs** for user authentication in the mobile app.
* Ensure **endpoints** are secured, especially those that deal with sensitive data (e.g., payment processing, order creation, user management).

**5.2 Rate Limiting**

* **Rate limiting** is essential to prevent abuse of your API, especially for **login** and **checkout** endpoints. Implement a rate-limiting mechanism (e.g., limit the number of login attempts to prevent brute force attacks).

**5.3 Input Validation and Sanitization**

* Always validate and sanitize user inputs, especially those that interact with databases, to prevent **SQL Injection** and **Cross-Site Scripting (XSS)** attacks.
* Use **parameterized queries** or **ORMs** (Object-Relational Mappers) to avoid SQL injection vulnerabilities.

**5.4 CORS (Cross-Origin Resource Sharing)**

* Configure **CORS** policies to ensure that only trusted domains can make requests to your API.
* Avoid using overly permissive CORS configurations to prevent cross-site request forgery (CSRF) attacks.

**6. Mobile App Security**

**6.1 Secure Storage**

* Avoid storing sensitive data such as passwords or payment details on the device. If local storage is necessary, use secure storage solutions such as **Keychain** (iOS) or **Keystore** (Android).
* For session management, store tokens securely (e.g., **Secure Storage** on mobile devices, or **Secure HttpOnly Cookies** for web apps).

**6.2 Code Obfuscation**

* Use **code obfuscation** techniques for mobile apps to prevent reverse engineering and protect proprietary code and logic.
* For Android, tools like **ProGuard** or **R8** can be used to obfuscate and optimize the app's code.

**6.3 App Integrity**

* Implement **integrity checks** to ensure that the app has not been tampered with (e.g., checking the app signature before processing sensitive data).
* Consider integrating with **Google SafetyNet** (Android) or **Apple App Attest** (iOS) to verify that the app is running on a genuine, untampered device.

**7. User Behavior Monitoring**

**7.1 Anomaly Detection**

* Monitor user behavior for suspicious activity such as login attempts from unusual IP addresses, multiple failed login attempts, or changes in account details that could indicate a potential account takeover.
* Use tools like **AWS GuardDuty**, **Cloudflare**, or **Google Cloud Security** for real-time monitoring and alerts.

**7.2 Audit Logs**

* Maintain detailed **audit logs** for all critical actions performed on the platform, such as account creation, login, order placement, payment processing, and admin actions. These logs should be encrypted and stored securely for potential forensic investigations.

**8. Security Testing and Auditing**

**8.1 Penetration Testing**

* Regularly conduct **penetration testing** and security audits to identify and fix potential vulnerabilities in the app, APIs, and backend services.
* Test for common vulnerabilities such as **SQL injection**, **XSS**, **CSRF**, and **server misconfigurations**.

**8.2 Vulnerability Scanning**

* Use automated **vulnerability scanning** tools (e.g., **OWASP ZAP**, **Burp Suite**) to scan your systems and codebase for known security vulnerabilities.

**Data Encryption**: All sensitive user information, including payment details, should be encrypted using SSL/TLS.

**Token-based Authentication**: Use JWT tokens for secure user authentication.

* **Role-based Access Control**: Implement proper role-based authorization to restrict access to sensitive operations.
* **Payment Security**: Use trusted third-party payment processors (e.g., Stripe, PayPal) for secure transactions.

1. **Future Enhancements**

**Advanced Personalization and AI Integration**

**1.1 AI-Powered Recommendations**

* **Personalized Restaurant & Menu Suggestions**: Leverage machine learning algorithms to provide personalized restaurant recommendations based on user preferences, past orders, dietary restrictions, and even time of day.
* **Dynamic Pricing Models**: Introduce AI-driven pricing models that offer personalized discounts or promotional offers based on the user's ordering history, loyalty, or location.

**1.2 Smart Search and Voice Recognition**

* **Voice Search**: Integrate voice-activated search using **Google Assistant** or **Siri** to allow users to find restaurants, dishes, or make orders hands-free.
* **Advanced Search Filters**: AI-driven filters that understand user intent (e.g., “low-calorie pizza options” or “best vegan restaurants near me”).

**1.3 Automated Meal Planning**

* **Dietary Assistant**: Use AI to analyze a user’s dietary preferences, restrictions, and past order history to automatically suggest weekly meal plans or healthy food choices.

**2. Enhanced Delivery Experience**

**2.1 Real-Time Delivery Tracking with AR**

* **Augmented Reality (AR)**: Implement **AR** to allow users to track their order in real time with interactive maps. Users can see the location of the delivery driver and estimated arrival in an immersive 3D view.
* **Driver Location Alerts**: Implement proximity-based notifications to alert users when their delivery driver is within a set distance from their location, enhancing delivery precision.

**2.2 Drone Delivery Integration**

* **Autonomous Drone Delivery**: In urban areas, pilot the integration of **drone delivery** for faster and more eco-friendly deliveries. Drones could be used for small, lightweight orders (e.g., a single meal or drink).
* **Real-Time Drone Tracking**: Users could track their drone delivery in real-time via the app and get accurate delivery time estimates.

**2.3 Contactless Delivery Options**

* **Drop-off Locations**: Enable users to specify **contactless delivery preferences**, such as drop-off at a specific location (e.g., doorstep, porch, or a safe location in a building).
* **QR Code Scanning**: Allow users to scan a QR code on the delivery package for added security or to confirm delivery receipt, especially useful for non-contact interactions.

**3. Sustainability and Eco-Friendly Initiatives**

**3.1 Sustainable Packaging**

* Partner with restaurants to encourage the use of **eco-friendly packaging** (e.g., biodegradable, recyclable, or reusable materials). Provide users with the option to choose sustainable packaging at checkout.

**3.2 Carbon Offset Programs**

* Implement a **carbon offset program** where users can contribute to offsetting the carbon footprint of their delivery. For example, the app could offer an option to pay a small extra fee for planting trees or funding clean energy projects, which aligns with sustainable delivery practices.

**3.3 Food Waste Reduction Features**

* Introduce a **food waste tracking** feature where users can choose to donate leftover food to charities or community organizations through the app, or suggest options for smaller portion sizes to prevent food waste.

**4. Expanded Payment Options**

**4.1 Cryptocurrency Payments**

* **Cryptocurrency Integration**: Allow users to pay for orders using popular cryptocurrencies (e.g., **Bitcoin**, **Ethereum**, **Stablecoins**) for users interested in digital currency options.
* **Blockchain for Transparency**: Use **blockchain technology** to securely and transparently handle payments, especially for customers who prefer decentralized, trustless transactions.

**4.2 In-App Wallet & Loyalty Program**

* **Snack Squad Wallet**: Create a built-in **digital wallet** for users to store funds, apply cashback, or redeem loyalty points directly within the app. Users can top up their wallet via traditional payment methods or direct transfers.
* **Loyalty Rewards Program**: Develop a **rewards system** that grants points or credits for every order placed, which can be redeemed for discounts, free deliveries, or special offers. Advanced options could involve tiered rewards for frequent users (e.g., Gold, Platinum, VIP customers).

**4.3 Buy Now, Pay Later (BNPL)**

* Integrate **Buy Now, Pay Later (BNPL)** services, allowing users to pay for large orders in installments, either interest-free or with minimal fees. Services like **Klarna** or **Afterpay** could be integrated as payment options.

**5. Enhanced User Engagement and Social Features**

**5.1 Social Ordering and Sharing**

* **Group Orders**: Introduce a feature for **group orders**, where multiple users can collaborate on one order and split payments. This is ideal for office teams, families, or friends.
* **Social Sharing**: Allow users to share their food orders and experiences on social media directly from the app, complete with photos, ratings, and hashtags for viral marketing.
* **User-Generated Content (UGC)**: Enable customers to post their food experiences, restaurant reviews, or recipes directly within the app. This can foster a community-driven experience and increase app engagement.

**5.2 In-App Chat for Real-Time Communication**

* **Chat with Restaurants/Delivery Drivers**: Provide in-app messaging for customers to communicate with restaurants or delivery drivers in real time. This feature would be useful for order customizations, delivery inquiries, or resolving issues quickly.

**5.3 Gamification and Challenges**

* **Order Challenges**: Implement **gamification** elements such as “order streaks,” badges, and challenges (e.g., “Order from a new restaurant every week for a month”).
* **Referral Programs**: Enhance the **referral program** with rewards for users who bring in new customers, with tiered bonuses based on the number of successful referrals.

**6. Improved Driver and Restaurant Tools**

**6.1 Driver AI Assistance**

* **Route Optimization with AI**: Use **AI-powered route optimization** to help drivers find the quickest and most fuel-efficient routes, saving time and reducing costs.
* **Driver Safety Alerts**: Implement real-time **driver safety alerts**, such as warnings about traffic, bad weather, or hazardous road conditions.

**6.2 Restaurant Dashboard Enhancements**

* **AI-Driven Inventory Management**: Integrate AI-powered tools for restaurant owners to track inventory, predict demand based on order patterns, and automate restocking alerts.
* **Advanced Analytics**: Provide restaurant owners with advanced **analytics dashboards** showing insights into customer preferences, top-performing dishes, and financial performance to improve their operational strategies.

**6.3 Delivery Window Customization for Restaurants**

* Allow restaurants to set more **customizable delivery windows** (e.g., express delivery, scheduled deliveries) based on their operational capabilities. This can help avoid delivery delays and improve customer satisfaction.

**7. Global Expansion and Localization**

**7.1 Language and Currency Localization**

* **Multi-Language Support**: Expand the app’s language offerings to support a global customer base. Offer localization in major languages such as **Spanish**, **Mandarin**, **French**, and others.
* **Currency Conversion**: Implement **automatic currency conversion** for international users to view prices in their local currency. This will be particularly useful for global expansion.

**7.2 International Restaurant Partnerships**

* As the platform expands globally, partner with **international restaurants** to offer regional cuisines specific to local markets (e.g., sushi in Japan, pasta in Italy, or tacos in Mexico).

**7.3 Localized Delivery Infrastructure**

* Develop a localized delivery infrastructure, integrating with local delivery partners or offering tailored solutions for each region, considering diverse urban and rural dynamics.

**8. Advanced Security Features**

**8.1 Biometric Authentication**

* Integrate **biometric authentication** (fingerprint or face recognition) for quicker and more secure login to the app.

**8.2 Blockchain-Based Loyalty and Payments**

* Use **blockchain** to provide a decentralized and tamper-proof system for managing loyalty points, rewards, and payment transactions. This can add an extra layer of security and transparency.

**8.3 Zero Trust Security Model**

* Implement a **Zero Trust** security model for backend infrastructure, where trust is never assumed, and users and devices must continuously authenticate themselves for access.

**Security Considerations**

When designing, deploying, or maintaining systems—whether software applications, hardware devices, or network infrastructure—security considerations are critical to protect against unauthorized access, data breaches, and other cyber threats. This section outlines key security considerations that should be accounted for in any system design or operation.

**1. Data Protection**

* **Encryption**: Ensure that data is encrypted both in transit (using protocols like TLS/SSL) and at rest (using strong encryption algorithms). This prevents unauthorized access even if the data is intercepted or accessed by malicious actors.
* **Data Masking**: When displaying sensitive information, consider using techniques such as data masking or tokenization to obscure the actual data from unauthorized users.

**2. Authentication and Authorization**

* **Strong Authentication**: Implement multi-factor authentication (MFA) wherever possible. MFA requires users to provide more than one form of identification, such as a password and a one-time passcode (OTP) sent to a phone.
* **Role-Based Access Control (RBAC)**: Ensure that users only have access to resources they need to perform their jobs. Limit permissions to the minimum necessary and apply the principle of least privilege.
* **Session Management**: Ensure that user sessions are properly managed, with mechanisms for timeouts, automatic logouts, and protection against session fixation and hijacking attacks.

**3. Network Security**

* **Firewalls and Intrusion Detection Systems (IDS)**: Use firewalls to limit access to trusted networks and monitor incoming/outgoing traffic with IDS to detect potential intrusions.
* **Segmentation and Isolation**: Divide networks into separate segments to limit the impact of a potential breach. Isolating critical systems from general users can help mitigate threats.
* **VPNs and Secure Channels**: Implement Virtual Private Networks (VPNs) for remote access and secure communication channels for internal and external communications.

**4. Vulnerability Management**

* **Patch Management**: Keep all software, including operating systems, frameworks, and applications, up to date with security patches and updates. Vulnerabilities in outdated software can be a prime target for attackers.
* **Regular Security Audits**: Perform regular vulnerability assessments and penetration testing to identify potential weaknesses in the system.
* **Secure Software Development Lifecycle (SDLC)**: Incorporate security checks into the development process, including secure coding practices, code reviews, and automated security testing.

**5. Incident Response and Monitoring**

* **Logging and Monitoring**: Maintain comprehensive logging to track user activities and system events. Logs should be protected and monitored for signs of suspicious behavior.
* **Incident Response Plan**: Develop and regularly test an incident response plan to ensure that your team can quickly detect, contain, and mitigate any security breaches.
* **Alerting and Automation**: Set up automated alerts for any anomalies or suspicious activities, and ensure that these alerts are acted upon promptly by the appropriate personnel.

**6. Third-Party Risks**

* **Vendor Security**: Assess the security posture of third-party vendors and partners who have access to your systems, data, or infrastructure. Make sure they comply with relevant security standards and policies.
* **Supply Chain Security**: Be aware of the risks posed by the software or hardware supply chain. Attacks can target software updates or vulnerable components from trusted vendors, so it's important to validate third-party code and assets.

**7. Physical Security**

* **Access Control**: Ensure physical security for critical systems and data centers. This may include restricted access to servers, surveillance, and authentication measures like biometrics or smart cards.
* **Environmental Controls**: Protect against environmental risks (e.g., fire, flooding, power surges) that could impact the integrity of data and systems.

**8. Compliance and Regulatory Requirements**

* **Legal and Regulatory Compliance**: Ensure that your system complies with relevant laws and regulations, such as GDPR, HIPAA, PCI-DSS, and others, depending on your industry and geographic region.
* **Data Retention and Disposal**: Establish policies for data retention and secure disposal of data when it is no longer needed, following legal and industry best practices.

**9. User Education and Awareness**

* **Security Training**: Regularly train employees and users on security best practices, including how to recognize phishing attempts, create strong passwords, and follow safe browsing habits.
* **Social Engineering Defense**: Educate staff to be aware of social engineering attacks, such as spear-phishing or pretexting, which attempt to trick users into divulging confidential information.

**10. Backup and Disaster Recovery**

* **Data Backup**: Ensure regular backups of critical data and systems, and verify that backups are stored securely and can be easily restored in case of an incident.
* **Disaster Recovery Plan**: Develop and test a disaster recovery plan to ensure business continuity in the event of a major security breach, natural disaster, or system failure.

**11. Security in Cloud and Distributed Environments**

* **Cloud Security**: If using cloud services, ensure that the cloud provider has appropriate security measures in place. Utilize cloud-specific security tools like Identity and Access Management (IAM), encryption, and monitoring.
* **Distributed Systems**: For microservices and distributed architectures, implement security measures like service-to-service authentication, API rate limiting, and security gateways to ensure secure communication between components.

**Conclusion**

Snack Squad is a powerful and user-centric food delivery platform designed to provide a seamless and enjoyable experience for both customers and restaurant partners. With its rich feature set and secure architecture, Snack Squad aims to be a top choice for food delivery in any region.

For further inquiries or developer support, please refer to the Snack Squad Developer Portal.