**Design Document: Crave Cart - Online Food Ordering Platform**

**Table of Contents:**

1. Overview
2. Goals and Objectives
3. System Architecture
4. Design Components
5. Database Design
6. User Interface Design
7. APIs and Endpoints
8. Technologies Stack
9. Security Considerations

10.Conclusion

**Overview**

**Crave Cart** is a MERN stack-based online food ordering platform that enables users to browse menus, add items to their cart, place orders, and track them. The platform includes a user-friendly interface for customers and a robust admin panel for managing products, orders, and users. The system integrates secure payment functionality via Stripe and provides a seamless and responsive user experience.

**Goals and Objectives**

**Goals:**

* Provide a fast and reliable food ordering experience for users.
* Enable restaurant admins to manage menus, orders, and customers efficiently.
* Ensure data security and scalability.

**Objectives:**

1. Build a modular, maintainable codebase.
2. Deliver an intuitive UI/UX for both users and admins.
3. Implement role-based authentication and authorization.
4. Provide seamless payment processing and secure transactions.

**System Architecture**

The architecture follows a Model-View-Controller (MVC) design pattern with the following components:

1. **Frontend:**
   * Built using React.js to handle user interactions and dynamic content rendering.
   * State management using Context API for global state handling.
2. **Backend:**
   * Built with Node.js and Express.js to manage API requests and business logic.
3. **Database:**
   * MongoDB for data persistence and management.
4. **Third-Party Integrations:**
   * Stripe for secure payment processing.

**Design Components**

**User Panel:**

* **Features:** Browse menus, filter products, add items to the cart, checkout, track orders.
* **Tech Stack:** React.js, CSS, Context API.

**Admin Panel:**

* **Features:** Add/edit/delete products, manage orders, view customer details.
* **Tech Stack:** React.js, CSS, Context API.

**Authentication System:**

* **Implementation:**
  + User and admin roles with JWT-based authentication.
  + Password hashing using Bcrypt for secure storage.

**Database Design**

**Collections:**

1. **Users:**
   * name: User's full name
   * email: User's email (unique)
   * password: Hashed password
   * role: User role (customer/admin)
   * cartData: Data for cart
2. **Products:**
   * name: Product name
   * description: Short description
   * price: Product price
   * category: Product category
   * image: Image URL
3. **Orders:**
   * userId: Reference to Users collection
   * items: List of products with quantity
   * Amount: Total price
   * Address: Delivery Address
   * status: Order status (Pending, Delivered, Cancelled)
   * date: Timestamp
   * payment: Payment Status

**User Interface Design**

**User Panel:**

* **Homepage:**
  + Hero section showcasing popular dishes.
  + Menu categories with filters (e.g., Vegetarian, Non-Vegetarian).
* **Cart:**
  + Display of selected items, total cost, and "Proceed to Checkout" option.
* **Checkout:**
  + Payment gateway integration (Stripe).

**Admin Panel:**

* **Dashboard:**
  + Overview of orders, products, and users.
* **Product Management:**
  + Add/edit/delete products with image uploads.
* **Order Management:**
  + Update order status and view order details.

**APIs and Endpoints**

**User APIs:**

1. **Authentication:**
   * POST /api/v1/register: Register new users.
   * POST /api/v1/login: Log in users and issue JWT tokens.

**Order APIs:**

1. **Place Orders:**
   * POST /api/v1/order/place: Place a new order.
2. **Verify Orders:**
   * POST /api/v1/order/verify: Verify the payment status of an order.
3. **Update Status:**
   * POST /api/v1/order/status: Update the status of an existing order (e.g., Pending, Delivered).
4. **User Orders:**
   * POST /api/v1/order/userorders: Fetch orders placed by a specific user.
5. **List Orders:**
   * GET /api/v1/order/list: Retrieve all existing orders.

**Food APIs:**

1. **Add Food:**
   * POST /api/v1/food/add: Add a new food item to the menu (includes image upload).
2. **List Food:**
   * GET /api/v1/food/list: Fetch all available food items.
3. **Remove Food:**
   * POST /api/v1/food/remove: Remove a food item from the menu.

**Cart APIs:**

1. **Add to Cart:**
   * POST /api/v1/cart/add: Add an item to the user's cart.
2. **Remove from Cart:**
   * POST /api/v1/cart/remove: Remove an item from the user's cart.
3. **Get Cart:**
   * POST /api/v1/cart/get: Fetch all items currently in the user's cart.

**Technologies Used**

* **Frontend:** React.js, CSS, Context API.
* **Backend:** Node.js, Express.js.
* **Database:** MongoDB.
* **Authentication:** JSON Web Tokens (JWT), Bcrypt.
* **Payment Gateway:** Stripe.

**Security Considerations**

**Authentication**

* **JWT Tokens:**
  + Securely generated and signed with a secret key.
  + Stored in the client's localStorage.
* **Password Security:**
  + Passwords hashed using scrypt.js before storing in the database.
  + Plain passwords are never stored or logged.

**Authorization**

* **Protected Routes:**
  + Backend routes require valid JWT tokens.
  + Frontend routes use higher-order components to restrict access.

**Conclusion**

Crave Cart aims to deliver a reliable, user-friendly, and secure online food ordering experience. By leveraging modern technologies and a robust design architecture, the platform ensures scalability, efficiency, and convenience for both users and administrators. The project embodies best practices in software development, making it a reliable and adaptable solution for the food service industry. With its focus on user satisfaction and seamless operations, Crave Cart is poised to set a new benchmark in the online food delivery space.