Akasa Air

IT'S YOUR SKY

Interview Task – Full Stack Developer

NAME – VISHNU NAIR
REG NO – 2347131
5 MCA A
EMAIL -
vishnu.muraleedharan@mca.christuniversity.in
DEPLOYED LINK -
https://vishnunairakasaairfd.vercel.app/

https://vishnunair-akasaair-fd.netlify.app/

v

Akasa Air
IT'S YOUR SKY

CONTENTS

1. **DESIGN OF THE FOOD ORDERING PLATFORM**
2. **IMPLEMENTATION OF THE FOOD ORDERING PLATFORM**
3. **DEPLOYMENT**
4. **REQUIREMENT DETAILS ANALYSIS**
5. **API TESTING(https://www.postman.com/vishnu-nair01/workspace/vishnu-nair-food-order-api/collection/33918390-a587283b-e7c9-4779-a686-f0ab536371b3?action=share&creator=33918390)**
6. **REFERENNCE**
7. **HOW TO RUN**
8. **IMAGE**

1. DESIGN OF THE FOOD ORDERING PLATFORM

The design is divided into frontend and backend, with a clear separation of concerns between the user interface and the server-side logic.

FRONTEND (REACT)

- **Technology Stack:** React (JavaScript framework), React Router for navigation, Axios for API requests, and Tailwind CSS for styling.
- **Key Components:**
  - Navbar: Displays navigation options (Login, Register, Browse Inventory, Cart).
  - Login/Register Pages: Allow users to sign in and sign up with email and password.
  - Inventory Page: Displays items grouped by categories (e.g.,Veg,Non-Veg). Users can browse items and add them to their cart.
  - Cart Page: Shows items added to the user's cart. Users can modify item quantities and proceed to checkout.
  - Orders Page: Displays the user's order history and order statuses.
- **State Management:** Context API is used for managing authentication (AuthContext) and cart data (CartContext).

BACKEND (SPRING BOOT & MONGODB)

- **Technology Stack:** Spring Boot for RESTful APIs, MongoDB for data persistence, JWT for authentication, BCrypt for password hashing.
- **Key Modules:**

- User Authentication: Handles user registration and login. JWT is used to generate secure tokens, and BCrypt hashes user passwords for security.
- Inventory Management: Provides endpoints to browse, add, update, and check item stock in the inventory.
- Cart Management: Allows users to add items to their cart, view saved cart items, and update cart contents.
- Order Processing: Handles checkout, verifies stock availability, and manages order history. Orders are stored in MongoDB, and users can view order statuses.

- **Security:** Spring Security is configured to secure all routes except authentication endpoints. JWT is used to validate user sessions.

2. IMPLEMENTATION OF THE FOOD ORDERING PLATFORM

Frontend Implementation

- **Component Structure**:
  - **App.jsx**: The main entry point that includes routing and layout components like Navbar.
  - **Login.jsx and Register.jsx:** Forms to handle user authentication, with Axios used to send login and registration requests.
  - **Inventory.jsx:** Fetches the available items from the backend and displays them with category filters.
  - **Cart.jsx:** Allows users to view and manage their cart items. Axios is used to sync cart data with the backend.
  - **Orders.js:** Displays the user's past orders, fetched from the backend.

Key Libraries:

- React Router: For page navigation (e.g., /login, /inventory, /cart, /orders).
- Axios: For making HTTP requests to the backend (e.g., login, add to cart, checkout).
- Tailwind CSS: For responsive UI design.

Backend Implementation

- **Controller Layer**: Handles HTTP requests and sends responses. Example:
  - **AuthController**: Manages login and registration.
  - **Cart Controller**: Manages cart-related operations (add, remove, update).
  - **OrderController**: Handles checkout.
  - **InventoryController**: Manages the item inventory (list, update stock).
- **Service Layer**: Business logic is handled here.
  - **AuthService**: Validates user data and communicates with the UserRepository to store user info securely.
  - **CartService**: Manages cart operations and ensures persistence.
  - **OrderService**: Handles order creation, stock validation, and retrieval of order history.

- **Repository Layer**: Interfaces with MongoDB.
    - **UserRepository**: Stores user data (email, hashed password, etc.).
    - **ItemRepository**: Stores items and handles stock updates.
    - **OrderRepository**: Stores and retrieves order history.
- **Security**: Configured with Spring Security and JWT. Unauthorized users are restricted from accessing protected routes.

3.DEPLOYMENT

- The backend is configured to be deployed on a platform https://render-web.onrender.com/ and frontend is deployed in https://www.netlify.com/ and https://vercel.com/

- Overall deployment link
- **https://vishnunair-akasaair-fd.netlify.app/**
- **https://vishnunairakasaairfd.vercel.app/**

  click on view detail to manage inventory
- **https://vishnunairakasaairfd.vercel.app/inventory**

4. REQUIREMENT DETAILS ANALYSIS

USER REGISTRATION AND AUTHENTICATION

- **Registration**: Implemented in the backend with a POST request to /api/auth/register. The user provides an email, password, and full name, which are validated and saved in the MongoDB database using UserRepository.
- **Authentication**: When a user logs in (POST /api/auth/login), the system verifies the credentials and generates a JWT token using JwtUtil. This token is then used to authenticate requests, allowing secure access to protected routes.

BROWSE ITEM INVENTORY

- **Browse by Category**: The InventoryController allows users to get a list of all items (GET /api/inventory/all). Each Item includes a category field, which can be used to filter items by categories such as Veg, Non-Veg, etc.
- **Creative Display**: On the front end, the Inventory page would fetch items and display them grouped by category. You can include item images, prices, and availability, leveraging the imageUrl and other fields of the Item model.

SELECTION BASKET/CART

1. **Add to Cart**: Users can add items to their cart via a POST request to /api/cart/add/{userEmail}, which associates the cart items with the user email. The backend ensures multiple quantities of the same item are allowed.
2. **Stock Check on Checkout**: The CartService checks stock availability before confirming a purchase. If an item is out of stock, the user is notified.

3. **Persistent Cart**: The cart is saved in the database and associated with the user's email. This ensures that the cart persists across sessions and devices
4. **Edge Case Handling**: If any errors occur during cart management (e.g., invalid items), appropriate responses are returned.
5. **Multi-Device Login**: The cart is tied to the user's email, allowing it to be accessible across multiple devices.

CHECKOUT

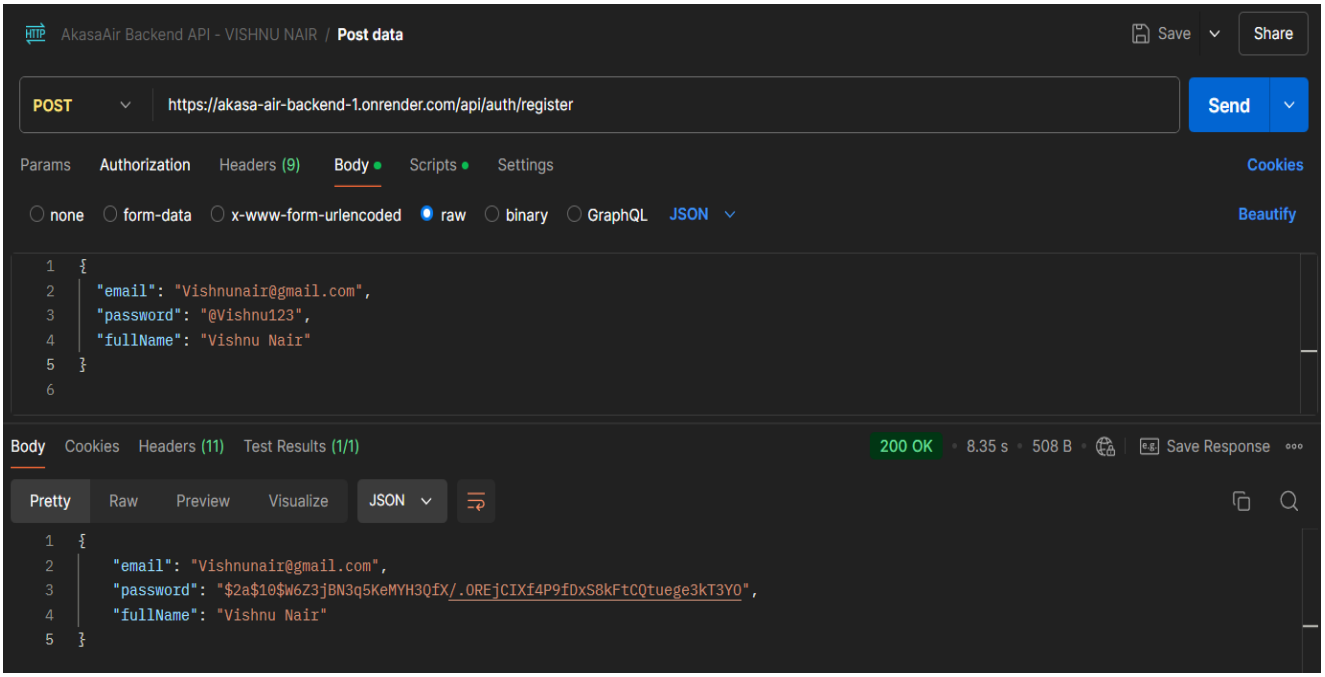
- **View Total Breakdown**: The checkout page will display a summary of all items, including their prices and the total cost.
- **Transaction Success**: After successfully completing the order, the system generates an Order with an order ID, storing the transaction details in MongoDB.
- **Handling Unavailable Items**: If any item in the cart is unavailable, the backend notifies the user and prevents them from completing the purchase.
- **Order Deduction**: The updateStock method in the ItemService ensures that stock is deducted only after the order is successfully checked out.
- **Order History**: Users can view their order history by making a GET request to /api/order/user/{email}.

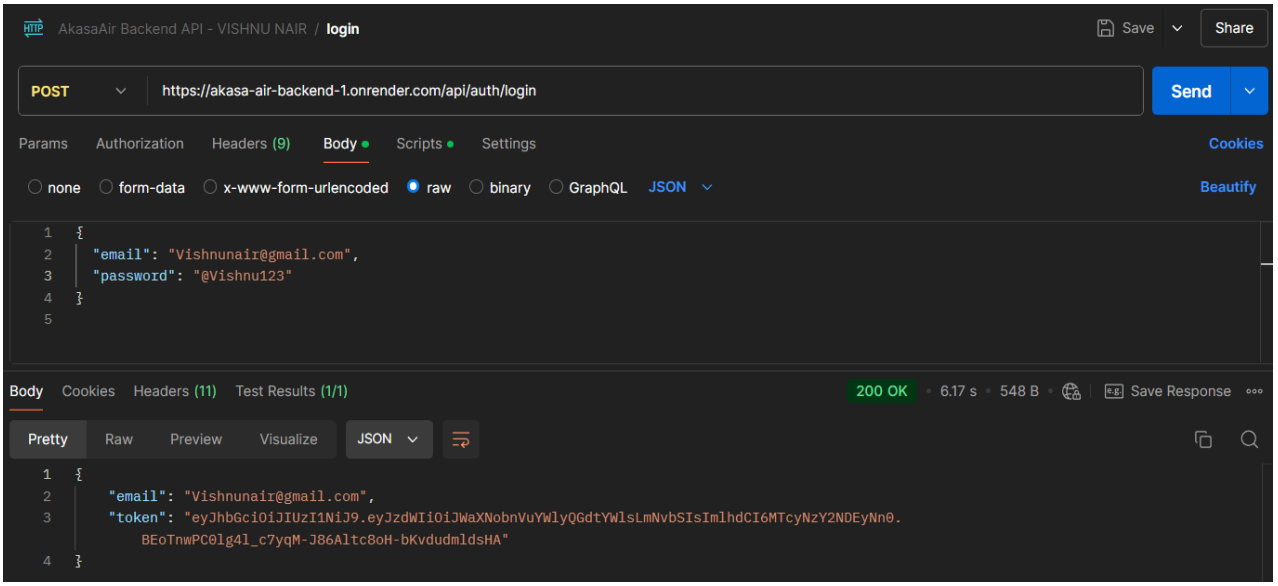
5. API TESTING

AUTHENTICATION API (AUTHCONTROLLER)

- **POST /api/auth/register**
  - **URL:** https://akasa-air-backend-1.onrender.com/api/auth/register
  - **Description:** Registers a new user by providing their email, password, and full name.
- **POST /api/auth/login**
  - **URL:** https://akasa-air-backend-1.onrender.com/api/auth/login
  - **Description:** Logs in a user and returns a JWT token for authenticated sessions.
- **GET /api/auth/check-email**
  - **URL:** https://akasa-air-backend-1.onrender.com/api/auth/check-email?email=user@example.com

- **Description:** Checks if the provided email is already registered in the system.

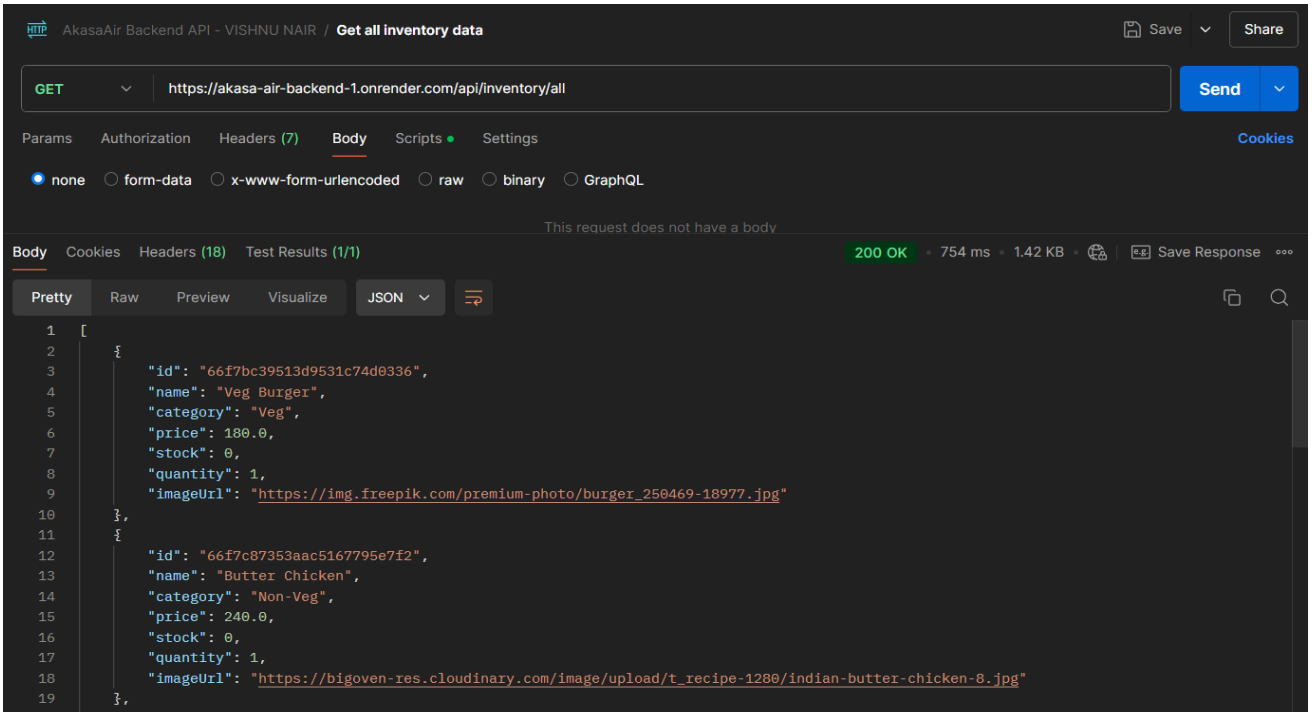




INVENTORY API (INVENTORYCONTROLLER)

- **GET /API/INVENTORY/ALL**
  - **URL:** https://akasa-air-backend-1.onrender.com/api/inventory/all
  - **Description:** Retrieves a list of all items available in the inventory.
- **POST /API/INVENTORY/ADD**
  - **URL:** https://akasa-air-backend-1.onrender.com/api/inventory/add
  - **Description:** Adds a new item to the inventory, including details like name, category, price, and stock.

- **PUT /API/INVENTORY/UPDATE**
  - **URL:** https://akasa-air-backend-1.onrender.com/api/inventory/update
  - **Description:** Updates the details of an existing item in the inventory.
- **DELETE /API/INVENTORY/DELETE/{ID}**
  - **URL:** https://akasa-air-backend-1.onrender.com/api/inventory/delete/{id}
  - **Description:** Deletes a specific item from the inventory using its ID.
- **GET /API/INVENTORY/{ID}**
  - **URL:** https://akasa-air-backend-1.onrender.com/api/inventory/{id}
  - **Description:** Retrieves details of a specific item from the inventory by its ID.
- **PUT /API/INVENTORY/UPDATE-STOCK**
  - **URL:** https://akasa-air-backend-1.onrender.com/api/inventory/update-stock
  - **Description:** Updates the stock level of an item after a checkout is completed.
- **GET /API/INVENTORY/CHECK-STOCK/{ID}/{QUANTITY}**
  - **URL:** https://akasa-air-backend-1.onrender.com/api/inventory/check-stock/{id}/{quantity}
  - **Description:** Checks if sufficient stock is available for a specific item and quantity before completing a checkout.



v

Akasa Air

IT'S YOUR SKY

AkasaAir Backend API - VISHNU NAIR / **Inventory Item Add**

Save | Share

POST https://akasa-air-backend-1.onrender.com/api/inventory/add Send

Params | Authorization | Headers (9) | Body | Scripts | Settings | Cookies

none | form-data | x-www-form-urlencoded | raw | binary | GraphQL | JSON | Beautify

```
{
  "name": "Egg Curry",
  "category": "Non-Veg",
  "price": 90,
  "stock": 1,
  "quantity": 1,
  "imageUrl": "https://images.services.kitchenstories.io/vmRyyK5xOTfkP0Jatqunl2h-11w=/1200x0/filters:quality(80)/images.kitchenstories.io/wagtailOriginalImages/R2899-photo-final-3x4.jpg"
}
```

Body | Cookies | Headers (18) | Test Results (1/1) — 200 OK • 713 ms • 847 B | Save Response

Pretty | Raw | Preview | Visualize | JSON

```
{
  "id": "66fa10f34dd8e93e17134625",
  "name": "Egg Curry",
  "category": "Non-Veg",
  "price": 90.0,
  "stock": 1,
  "quantity": 1,
  "imageUrl": "https://images.services.kitchenstories.io/vmRyyK5xOTfkP0Jatqunl2h-11w=/1200x0/filters:quality(80)/images.kitchenstories.io/wagtailOriginalImages/R2899-photo-final-3x4.jpg"
}
```

AkasaAir Backend API - VISHNU NAIR / **Update item data**

Get all inventory data
https://akasa-air-backend-1.onrender.com/api/inventory/all

Save | Share

PUT https://akasa-air-backend-1.onrender.com/api/inventory/update Send

Params | Authorization | Headers (9) | Body | Scripts | Settings | Cookies

none | form-data | x-www-form-urlencoded | raw | binary | GraphQL | JSON | Beautify

```
{
    "id": "66f7bc39513d9531c74d0336",
    "name": "Veg Burger",
    "category": "Veg",
    "price": 180.0,
    "stock": 20,
    "quantity": 1,
    "imageUrl": "https://img.freepik.com/premium-photo/burger_250469-18977.jpg"
}
```

Body | Cookies | Headers (18) | Test Results (1/1) — 200 OK • 773 ms • 774 B | Save Response

Pretty | Raw | Preview | Visualize | JSON

```
{
  "id": "66f7bc39513d9531c74d0336",
  "name": "Veg Burger",
  "category": "Veg",
  "price": 180.0,
  "stock": 20,
  "quantity": 1,
  "imageUrl": "https://img.freepik.com/premium-photo/burger_250469-18977.jpg"
}
```

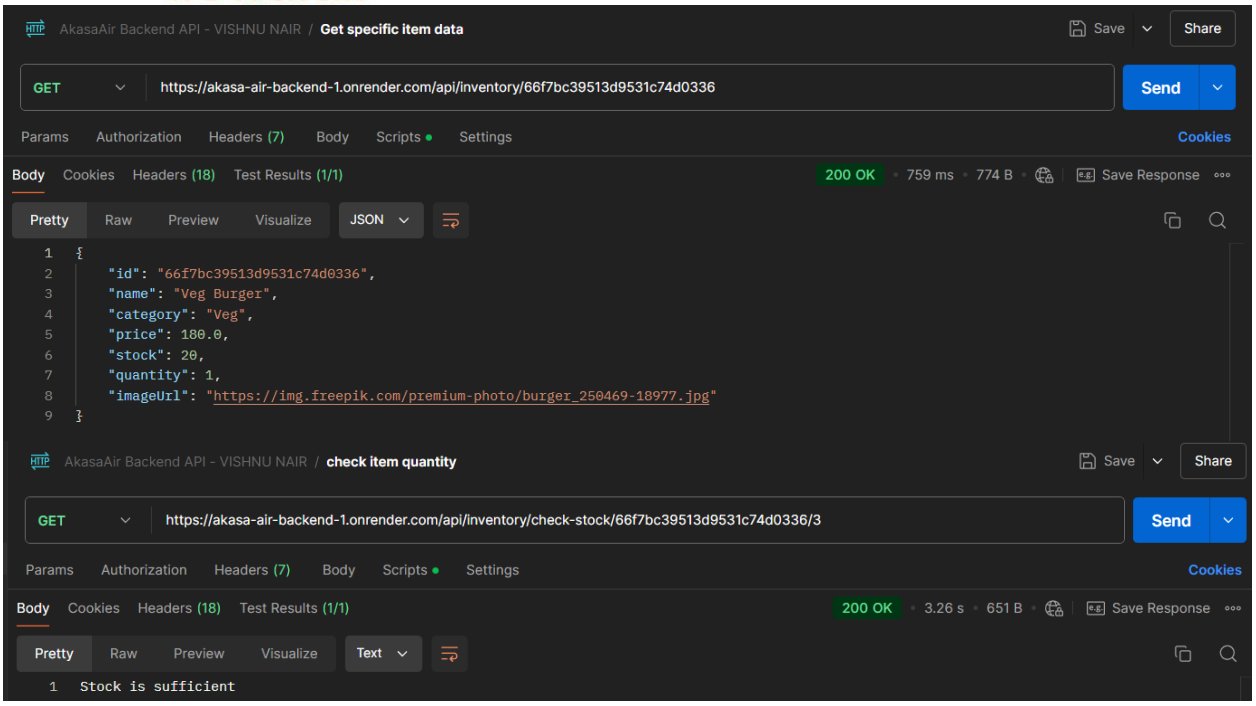
AkasaAir Backend API - VISHNU NAIR / **Delete item data**

Save | Share

DELETE https://akasa-air-backend-1.onrender.com/api/inventory/delete/66f7c93653aac5167795e7f4 Send

Params | Authorization | Headers (7) | Body | Scripts | Settings | Cookies

Body | Cookies | Headers (16) | Test Results (1/1) — 200 OK • 1922 ms • 565 B | Save Response



CART API (CARTCONTROLLER)

- **POST /API/CART/ADD/{USEREMAIL}**
  - **URL:** https://akasa-air-backend-1.onrender.com/api/cart/add/{userEmail}
  - **Description:** Adds specified items to a user's cart based on their email.
- **GET /API/CART/USER/{EMAIL}**
  - **URL:** https://akasa-air-backend-1.onrender.com/api/cart/user/{email}
  - **Description:** Retrieves the current cart contents for a specific user identified by their email.
- **PUT /API/CART/UPDATE**
  - **URL:** https://akasa-air-backend-1.onrender.com/api/cart/update
  - **Description:** Updates the contents of a user's cart with new items or quantities.
- **DELETE /API/CART/REMOVE/{EMAIL}/{ITEMID}**
  - **URL:** https://akasa-air-backend-1.onrender.com/api/cart/remove/{email}/{itemId}
  - **Description:** Removes a specified item from a user's cart based on their email and item ID.

Akasa Air
IT'S YOUR SKY

AkasaAir Backend API - VISHNU NAIR / **add item to cart**

POST https://akasa-air-backend-1.onrender.com/api/cart/add/vishnunair@gmail.com

Send

Body: raw, JSON

```
{
  "itemId": "66f7bc39513d9531c74d0336",
  "quantity": 2
}
```

200 OK · 1736 ms · 863 B

```
{
    "id": "66fa14da4dd8e93e17134626",
    "userEmail": "vishnunair@gmail.com",
    "items": [
        {
            "itemId": "66f7bc39513d9531c74d0336",
            "quantity": 2,
            "name": "Veg Burger",
            "price": 180.0,
            "imageUrl": "https://img.freepik.com/premium-photo/burger_250469-18977.jpg"
        }
    ]
}
```

AkasaAir Backend API - VISHNU NAIR / **Get specific item data Copy**

GET https://akasa-air-backend-1.onrender.com/api/cart/user/vishnunair@gmail.com

Send

This request does not have a body

200 OK · 1257 ms · 866 B

```
{
    "id": "66fa14da4dd8e93e17134626",
    "userEmail": "vishnunair@gmail.com",
    "items": [
        {
            "itemId": "66f7bc39513d9531c74d0336",
            "quantity": 2,
            "name": "Veg Burger",
            "price": 180.0,
            "imageUrl": "https://img.freepik.com/premium-photo/burger_250469-18977.jpg"
        }
    ]
}
```

AkasaAir Backend API - VISHNU NAIR / **update cart item**

PUT https://akasa-air-backend-1.onrender.com/api/cart/update

Send

Body: raw, JSON

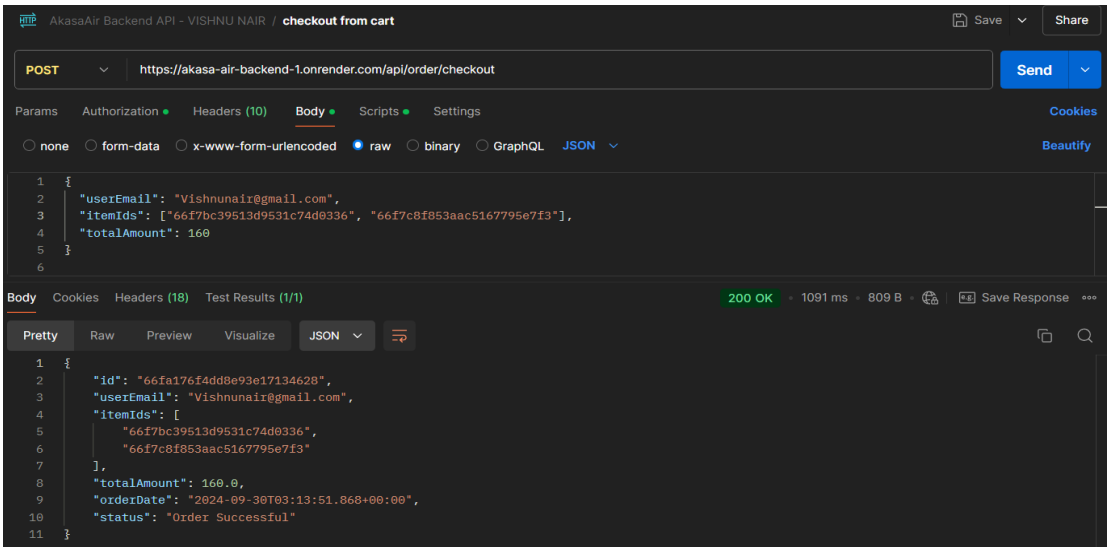
```
{
  "userEmail": "Vishnunair@gmail.com",
  "items": [
    {
      "itemId": "66f7c87353aac5167795e7f2",
      "quantity": 3
    },
    {
      "itemId": "66f7bc39513d9531c74d0336",  // Replace with another item ID
      "quantity": 4
    }
  ]
}
```

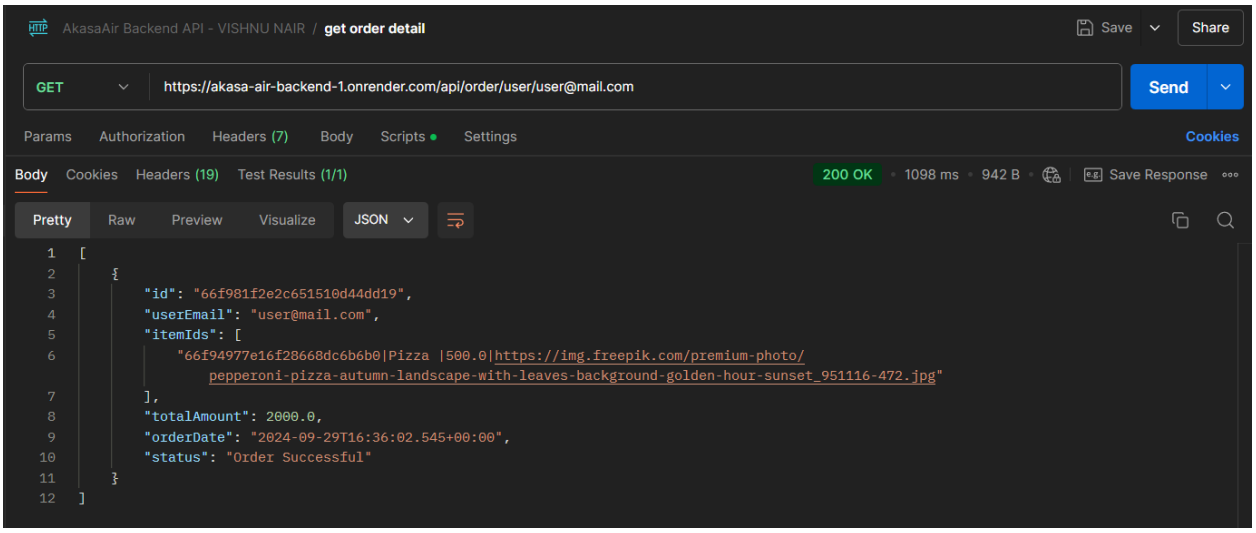
200 OK · 1068 ms · 830 B

ORDER API (ORDERCONTROLLER)

- **POST /API/ORDER/CHECKOUT**
  - **URL:** https://akasa-air-backend-1.onrender.com/api/order/checkout
  - **Description:** Creates a new order for a user by providing their email, item IDs, and total amount.

- **GET /API/ORDER/USER/{EMAIL}**
    - **URL:** https://akasa-air-backend-1.onrender.com/api/order/user/{email}
    - **Description:** Retrieves all orders associated with a specific user identified by their email.





6. REFERENCE

UI REFERENCE
https://www.behance.net/gallery/111586923/Resturant-website-Design

7. HOW TO RUN

Prerequisites

- Node.js and npm installed
- Java Development Kit (JDK) 11 or later
- MongoDB installed and running

Frontend Setup

1. Navigate to the frontend directory:

cd frontend

2. Install dependencies:

npm install

3. Start the development server:

npm start

The frontend will be available at http://localhost:3000.

Backend Setup

1. Navigate to the backend directory:

cd backend

2. Build the project:

./mvnw clean install

3. Run the Spring Boot application:

./mvnw spring-boot:run

The backend will start on http://localhost:8080.

Environment Configuration

Create a .env file in the frontend directory with the following content:

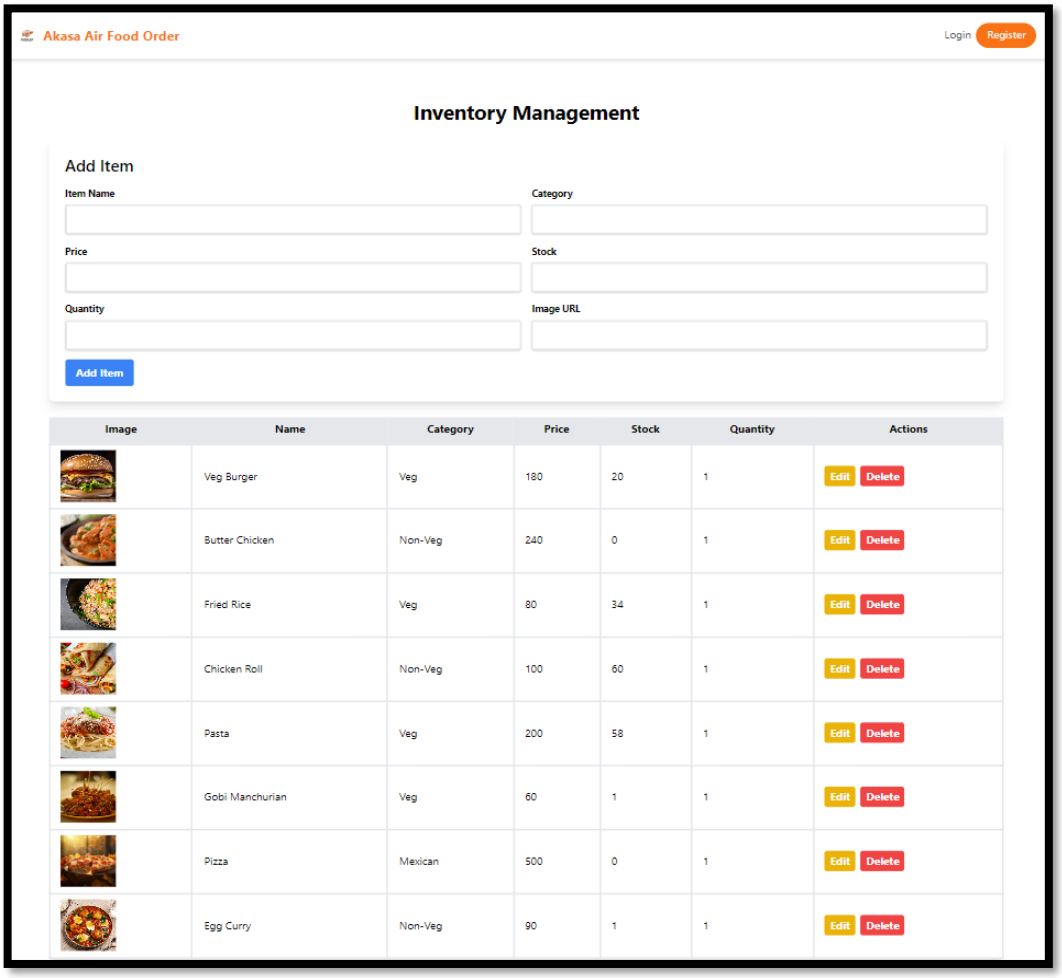
REACT_APP_API_URL=http://localhost:8080/api

Create an application.properties file in the src/main/resources directory of the backend with:

spring.data.mongodb.uri=mongodb://localhost:27017/food_ordering_db
jwt.secret=your_jwt_secret_key

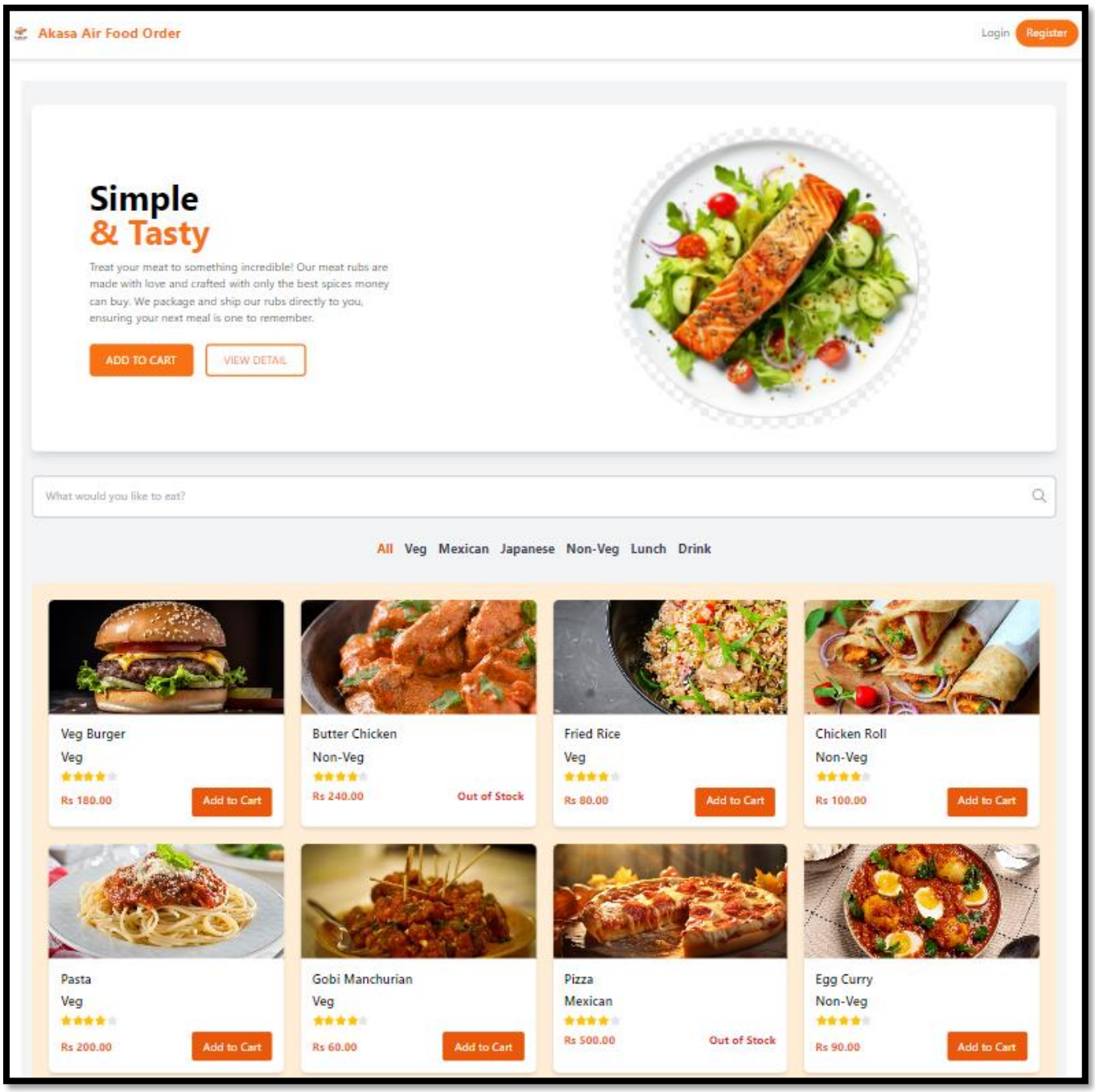
Replace your_jwt_secret_key with a secure random string.

8. IMAGES



v

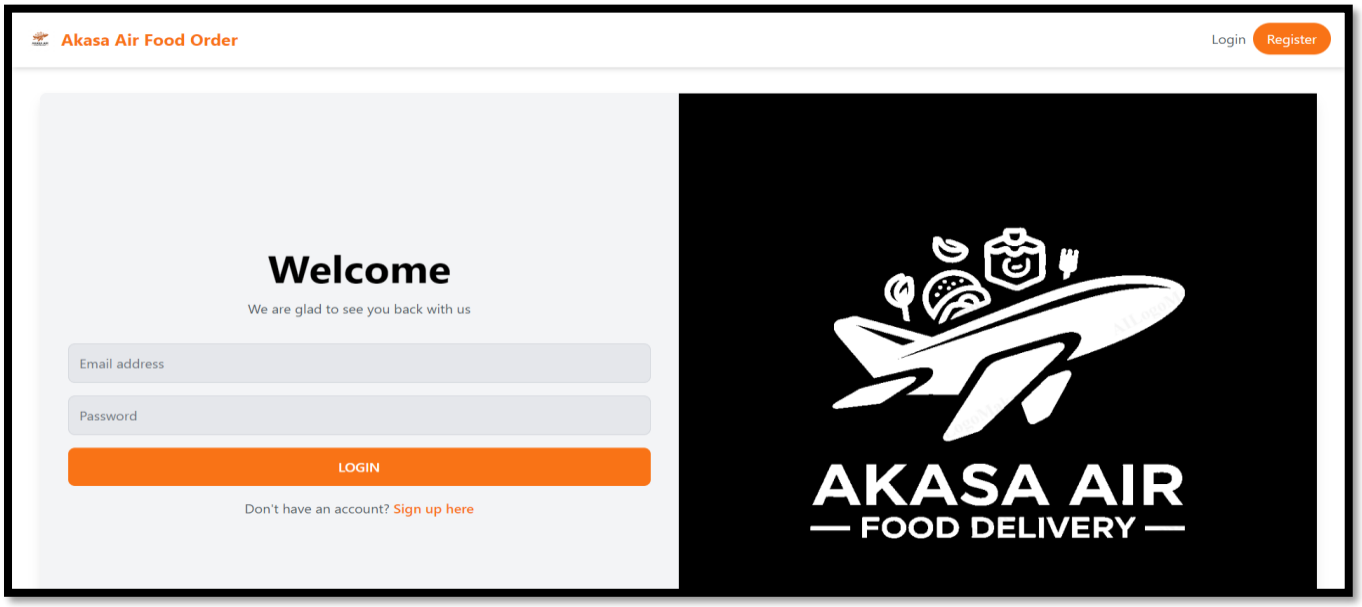
Akasa Air
IT'S YOUR SKY



v

Akasa Air

IT'S YOUR SKY



V

Akasa Air

IT'S YOUR SKY