Documentation of Akasa Air Food App

by Ajay Sanjay Panaskar

Overview

This project is a full-stack food ordering platform built using Node.js (Express) for the backend, React for the frontend, and PostgreSQL as the database. It allows users to browse a selection of food items, manage a cart, place orders, and make payments through the Razorpay payment gateway. The system is secure, utilizing JWT and bcrypt for authentication and password management.

System Design

1. Application Architecture
    - Frontend: React with Context API
    - Backend: NodeJs, ExpressJs
    - Database: Postgres
    - Authentication: Using JWT

2. Features

- User Registration and Authentication
- Users can register using an email and password.
- User authentication is handled with sessions; once logged in, the session is persisted across pages.

- Browsing Food Items
    - Items are categorized into Pizza, Burgers, Desserts, Breads, etc.
    - Users can browse and filter items by these categories.

- Shopping Cart
- Users can add multiple items to their cart with specified quantities.
- Items in the cart are persisted, and users can see their cart history across sessions.

- Users can adjust quantities of items in their cart.

- Checkout
- Users can view a summary of their selected items before proceeding to checkout.
- Upon successful checkout, the stock is updated, and users receive an order confirmation.

- Order Management
- Users can view their order history.
- Order delivery status is tracked and visible.

- Payment Integration
- Users can securely process payments through the Razorpay payment gateway.
- Payment confirmations are sent to users after successful transactions.

- Wishlist
- Users can save items to a wishlist for future purchases.
- The wishlist allows users to easily access items they’re interested in without cluttering their cart.

Database Schema

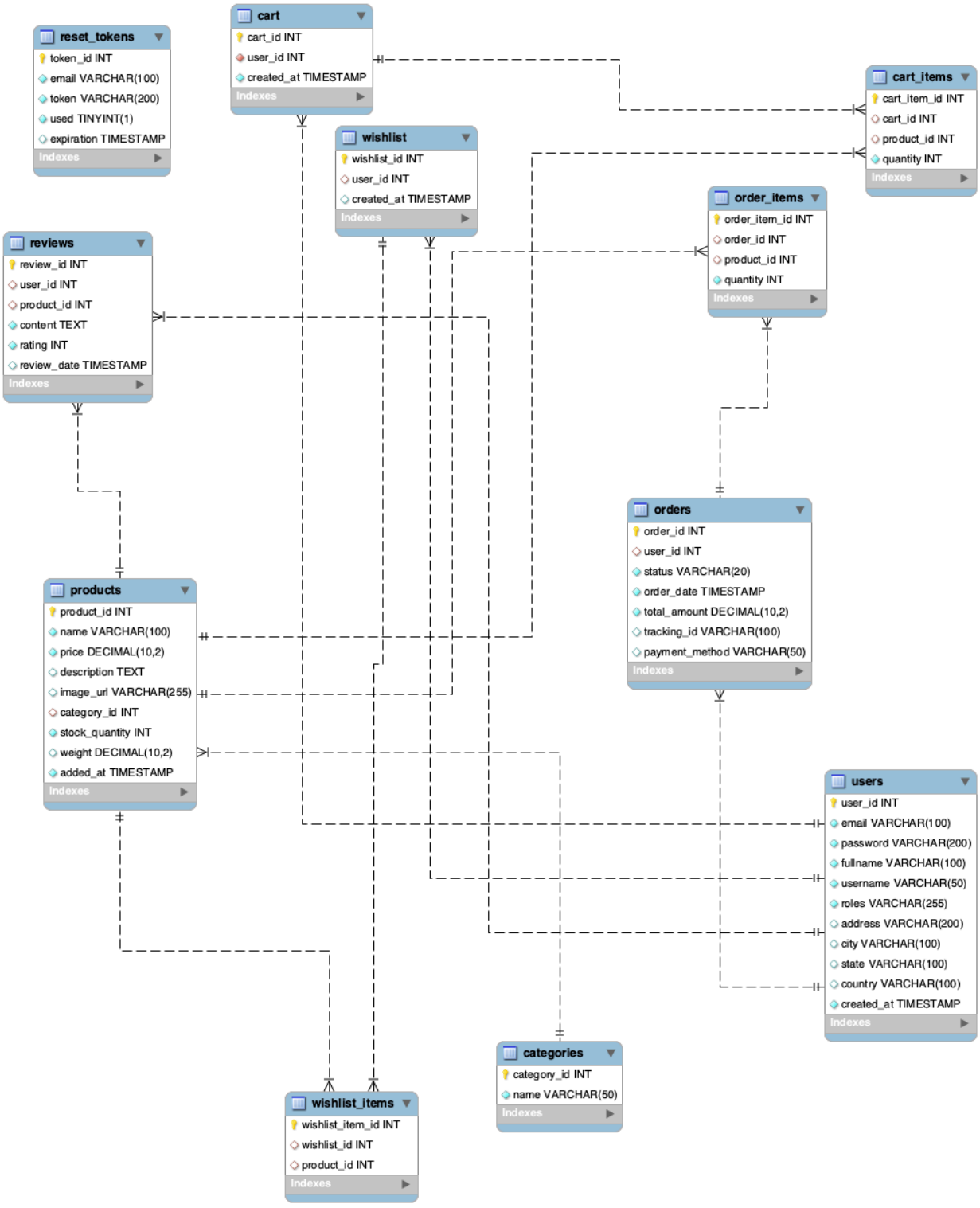
```
CREATE TABLE public.users

(

    user_id SERIAL PRIMARY KEY,

    email VARCHAR(100) UNIQUE NOT NULL,

    password VARCHAR(200) NOT NULL,

    fullname VARCHAR(100) NOT NULL,
```

```
    username VARCHAR(50) UNIQUE NOT NULL,
    roles VARCHAR(10)[] DEFAULT '{customer}'::VARCHAR[] NOT NULL,
    address VARCHAR(200),
    city VARCHAR(100),
    state VARCHAR(100),
    country VARCHAR(100),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP NOT NULL
);

CREATE TABLE public.categories
(
    category_id SERIAL PRIMARY KEY,
    name VARCHAR(50) NOT NULL UNIQUE
);

CREATE TABLE public.products
(
    product_id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    price REAL NOT NULL,
    description TEXT,
    image_url VARCHAR(255),
    category_id INTEGER REFERENCES public.categories (category_id) ON DELETE SET NULL,
    stock_quantity INTEGER NOT NULL DEFAULT 0,
    weight REAL,
    added_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP NOT NULL
);

CREATE TABLE public.cart
(
    cart_id SERIAL PRIMARY KEY,
```

```
    user_id INTEGER UNIQUE NOT NULL REFERENCES public.users (user_id) ON DELETE CASCADE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP NOT NULL
);



CREATE TABLE public.cart_items
(
    cart_item_id SERIAL PRIMARY KEY,
    cart_id INTEGER REFERENCES public.cart (cart_id) ON DELETE CASCADE,
    product_id INTEGER REFERENCES public.products (product_id) ON DELETE SET NULL,
    quantity INTEGER NOT NULL CHECK (quantity > 0),
    UNIQUE (cart_id, product_id)
);

CREATE TABLE public.orders
(
    order_id SERIAL PRIMARY KEY,
    user_id INTEGER REFERENCES public.users (user_id) ON DELETE CASCADE,
    status VARCHAR(20) NOT NULL,  -- e.g., 'Processing', 'Delivered', 'Cancelled'
    order_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP NOT NULL,
    total_amount REAL NOT NULL,
    tracking_id VARCHAR(100) UNIQUE,
    payment_method VARCHAR(50)
);



CREATE TABLE public.order_items
(
    order_item_id SERIAL PRIMARY KEY,
    order_id INTEGER REFERENCES public.orders (order_id) ON DELETE CASCADE,
    product_id INTEGER REFERENCES public.products (product_id) ON DELETE SET NULL,
```

```
    quantity INTEGER NOT NULL CHECK (quantity > 0)
);



CREATE TABLE public.wishlist
(
    wishlist_id SERIAL PRIMARY KEY,
    user_id INTEGER REFERENCES public.users (user_id) ON DELETE CASCADE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);



CREATE TABLE public.wishlist_items
(
    wishlist_item_id SERIAL PRIMARY KEY,
    wishlist_id INTEGER REFERENCES public.wishlist (wishlist_id) ON DELETE CASCADE,
    product_id INTEGER REFERENCES public.products (product_id) ON DELETE SET NULL,
    UNIQUE (wishlist_id, product_id)
);

CREATE TABLE public.reset_tokens
(
    token_id SERIAL PRIMARY KEY,
    email VARCHAR(100) NOT NULL,
    token VARCHAR(200) NOT NULL,
    used BOOLEAN DEFAULT false NOT NULL,
    expiration TIMESTAMP
);

CREATE TABLE public.reviews
(
```

```
    review_id SERIAL PRIMARY KEY,
    user_id INTEGER REFERENCES public.users (user_id) ON DELETE SET NULL,
    product_id INTEGER REFERENCES public.products (product_id) ON DELETE SET NULL,
    content TEXT NOT NULL,
    rating INTEGER CHECK (rating >= 1 AND rating <= 5) NOT NULL,
    review_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```


reset_tokens
token_id INT
email VARCHAR(100)
token VARCHAR(200)
used TINYINT(1)
expiration TIMESTAMP
Indexes
cart
cart_id INT
user_id INT
created_at TIMESTAMP
Indexes
cart_items
cart_item_id INT
cart_id INT
product_id INT
quantity INT
Indexes
wishlist
wishlist_id INT
user_id INT
created_at TIMESTAMP
Indexes
order_items
order_item_id INT
order_id INT
product_id INT
quantity INT
Indexes
reviews
review_id INT
user_id INT
product_id INT
content TEXT
rating INT
review_date TIMESTAMP
Indexes
orders
order_id INT
user_id INT
status VARCHAR(20)
order_date TIMESTAMP
total_amount DECIMAL(10,2)
tracking_id VARCHAR(100)
payment_method VARCHAR(50)
Indexes
products
product_id INT
name VARCHAR(100)
price DECIMAL(10,2)
description TEXT
image_url VARCHAR(255)
category_id INT
stock_quantity INT
weight DECIMAL(10,2)
added_at TIMESTAMP
Indexes
users
user_id INT
email VARCHAR(100)
password VARCHAR(200)
fullname VARCHAR(100)
username VARCHAR(50)
roles VARCHAR(255)
address VARCHAR(200)
city VARCHAR(100)
state VARCHAR(100)
country VARCHAR(100)
created_at TIMESTAMP
Indexes
categories
category_id INT
name VARCHAR(50)
Indexes
wishlist_items
wishlist_item_id INT
wishlist_id INT
product_id INT
Indexes

API Routes(below are some basic API endpoints, the Postman API documentation link is also attached)

- Postman API documentation

  - https://documenter.getpostman.com/view/19272653/2sAXqtcMjw

- User APIs
- POST /api/register: Register a new user.
- POST /api/login: Authenticate a user.

- Item APIs
- GET /api/products: Fetch all items.
- GET /api/products/:category: Fetch items by category.

- Cart APIs
- POST /api/cart: Add items to the cart.
- GET /api/cart: Retrieve the user's cart.
- DELETE /api/cart/:itemId: Remove item from cart.

- Order APIs
- POST /api/checkout: Process the checkout.
- GET /api/orders: Fetch order history.

- Review APIs
- POST /api/reviews: Add a product review.
- GET /api/reviews/:productId: Get reviews for a product.

GitHub Repository link

- https://github.com/Ajay8309/Akasa-foodApp/
- The project working video is attached with the GitHub repository in Readme file