Restaurant Ordering System — Combined Docs

This document compiles the key resources to run and understand the project: architecture, dataset/seed info, README (quick start), the Postman API collection summary, and seed script highlights.

---

1) Architecture & Design

(From ARCHITECTURE.md)

```
# Architecture & Design

## Overview
This document describes the high-level architecture of the Restaurant Ordering System. The application
follows a standard modern full-stack pattern using Next.js for the frontend and API routes, and MongoDB
(via Mongoose) for persistence.

## Components

- Frontend: Next.js (App Router) + Tailwind CSS
- Backend: Next.js API routes (server code inside `app/api/*`)
- Database: MongoDB with Mongoose models
- Authentication: NextAuth.js (credentials provider, JWT sessions)
- Seed data: `scripts/seed.ts` to initialize demo users, countries, restaurants, menu items, and payment
methods

## High-level Flow
1. User visits the site (SSR or CSR via Next.js pages).
2. Login via credentials provider; server validates user and issues JWT session.
3. Auth middleware enforces RBAC and country-based filtering.
4. API routes handle CRUD operations for restaurants, orders, and payment methods.
5. Orders are created and stored in MongoDB; checkout updates order status and records payment method.

## Data Model (ERD summary)
- Country 1:N Restaurants
- Country 1:N Users
- Restaurant 1:N MenuItem
- User 1:N Orders
- Order contains many MenuItems (items array)
- PaymentMethod 1:N Orders

## Security & Permissions
- Passwords hashed with `bcryptjs`.
- Server-side RBAC enforced on API routes; Admin/Manager/Member roles with permission objects.
- Country-based data isolation: Managers and Members only see their assigned country; Admin sees all.

## Deployment Notes
- Environment variables required: `MONGODB_URI`, `NEXTAUTH_SECRET`, `NEXTAUTH_URL`.
- Build: `npm run build` then `npm start` for production.
- Recommended hosting: Vercel for frontend + API routes; MongoDB Atlas for DB.
```

````
## Scaling & Performance
- Add indexes on frequently queried fields (user.email, restaurant.country, order.status).
- Use connection pooling and horizontal scaling for the API layer.
- Use CDN and image optimization for restaurant images (next/image + external source domain whitelist).

## Observability
- Add logging (structured) and basic metrics (request rate, error rates).
- Centralized error reporting for critical endpoints (Sentry or similar).

## Future improvements
- Move heavy business logic to dedicated serverless functions or microservices.
- Add caching for restaurant listings and menu data.
- Add payment provider integration (Stripe) for real checkout.
````

---

2) Datasets & Seed Data

(From DATASETS.md)

~~~
# Datasets & Seed Data

This file documents the seed data used for the demo application and how to re-seed.

## How to seed

Run:

```bash
npm run seed
~~~

This will populate the database with countries, roles, users, restaurants, menu items, and payment methods.

Sample seed summary

- Countries:
  - India
  - USA
- Roles:
  - Admin (full permissions)
  - Manager (country-scoped management)
  - Member (basic customer)
- Demo Users (examples):
  - nick.fury@shield.com — Admin — password123
  - carol.danvers@avengers.com — Manager (USA) — password123
  - steve.rogers@avengers.com — Manager (India) — password123
  - thanos@titan.com — Member (India) — password123
  - thor@asgard.com — Member (India) — password123

   - travis@usa.com — Member (USA) — password123

- Restaurants (examples):

   - Spice Garden (India) — Indian cuisine
   - Taj Mahal (India) — Indian cuisine
   - Curry House (India) — Indian cuisine
   - American Grill (USA) — American cuisine
   - Liberty Diner (USA) — American cuisine
   - Burger Paradise (USA) — Burgers

- Menu items: 4–8 items per restaurant (name, description, price)

- Payment Methods:

   - Card
   - NetBanking
   - UPI

Test scenarios

1. Member in India creates an order and attempts checkout (fails if no permission).
2. Manager in India creates an order and checks out for a user in India.
3. Admin views all orders and manages payment methods.

Notes

- The scripts/seed.ts file contains full seed objects; edit it to customize data.
- If seeding fails, ensure MONGODB_URI is set and the DB is reachable.

~~~

---

3) Quick Start / README

(From `README.md`)

```markdown
# Restaurant Ordering System

A full-stack restaurant ordering system with role-based access control (RBAC), built with Next.js 14,
MongoDB, and TypeScript.

## Prerequisites

- Node.js 18+
- MongoDB (local or Atlas)

## Setup Instructions

1. **Install dependencies**
```bash
npm install
~~~

2. **Create environment file**

Create .env.local in the root directory:

```
MONGODB_URI=mongodb://localhost:27017/restaurant-ordering
NEXTAUTH_URL=http://localhost:3000
NEXTAUTH_SECRET=your-secret-key-here
```

3. **Seed the database**

```
npm run seed
```

4. **Run the application**

```
npm run dev
```

5. **Open browser**

```
http://localhost:3000
```

Demo Credentials

Role	Email	Password
Admin	nick.fury@shield.com	password123
Manager	steve.rogers@avengers.com	password123
Member	thanos@titan.com	password123

```
---

## 4) API Collection (Postman)

(Summary from `postman-collection.json`)

- File path: `postman-collection.json` (included in repo root)
- Usage: Import into Postman or run with Newman.
- Key collections:
  - Authentication: login endpoints for seeded users and `GET /api/auth/session`.
  - Restaurants: `GET /api/restaurants`, `GET /api/restaurants/:id`.
  - Orders: `GET /api/orders`, `POST /api/orders`, `GET /api/orders/:id`, `POST /api/orders/:id/checkout`, `POST /api/orders/:id/cancel`.
  - Payment Methods: `GET /api/payment-methods`, `POST /api/payment-methods` (Credit Card / UPI), `PATCH /api/payment-methods/:id`, `DELETE /api/payment-methods/:id`.

The full JSON collection is available at `postman-collection.json` (attached in the repository). Import it into Postman and set `baseUrl` to `http://localhost:3000`.

---

## 5) Seed Script Highlights

(From `scripts/seed.ts`)

- The seed script creates countries (India, America), roles (Admin, Manager, Member), users (Nick Fury, Captain Marvel, Captain America, Thanos, Thor, Travis), restaurants and menu items, and payment methods. It prints seeded user credentials on completion. Example printed credentials:
```

Admin:
Email: nick.fury@shield.com
Password: password123
Country: America

Managers:
Email: carol.danvers@avengers.com (India)
Email: steve.rogers@avengers.com (America)
Password: password123

Members:
Email: thanos@titan.com (India)
Email: thor@asgard.com (India)
Email: travis@example.com (America)
Password: password123

~~~
---

## 6) How to get a PDF from this file (local commands)

Try one of these on your machine (pick whichever tool you have):

- With `pandoc` (recommended if installed):

```bash
pandoc EXPORT_DOCS/COMPILATION.md -o EXPORT_DOCS/Restaurant-Ordering-System-Docs.pdf
~~~

- Or use npx markdown-pdf (will install a converter temporarily):

```
npx -y markdown-pdf EXPORT_DOCS/COMPILATION.md -o EXPORT_DOCS/Restaurant-Ordering-System-Docs.pdf
```

- Or convert via Chrome headless (if you have Node + Puppeteer script), or manually paste the Markdown into an editor and export to PDF.

---

7) Where are the original files

- ARCHITECTURE.md — architecture & design
- DATASETS.md — dataset/seed description
- README.md — quick start
- postman-collection.json — full Postman collection
- scripts/seed.ts — seed script

All files are in the repository root.

If you want, I can now try to convert this combined Markdown to PDF on this machine and place the resulting PDF in EXPORT_DOCS/ — would you like me to attempt the conversion now? If so, I will try pandoc and then fall back to npx markdown-pdf and report back with the PDF path (or exact error if conversion fails).