Aroglin Spice Shop - FULL DOCUMENTATION

Aroglin Spice Farms - E-Commerce Platform

A full-stack e-commerce application for a local spice shop, built with Spring Boot (Backend) and Next.js (Frontend). Customers can browse, search, and purchase premium spices online, while admins manage inventory, orders, and customer relationships.

FEATURES

Customer Features:
- Email-based registration with OTP verification
- Secure login with session management
- Password change functionality
- Browse, filter, and search products
- View detailed spice information
- Related product suggestions
- Shopping cart with quantity update
- Place orders & track order status
- Razorpay & COD payment support
- Profile management

Admin Features:
- Secure admin login
- Password recovery with secret key
- Add/edit/delete spices
- Multiple variants & pack sizes
- Product image uploads
- Order management with status updates
- Customer list and profile view
- Admin profile update

TECH STACK

Backend:
- Spring Boot 3.4.5
- Java 21
- PostgreSQL
- Spring Security, BCrypt hashing
- Spring Session JDBC
- Razorpay API
- Gmail SMTP (Spring Mail)
- JPA / Hibernate
- Maven

Frontend:
- Next.js 15.3

- React 19
- Tailwind CSS 4
- Framer Motion
- Heroicons & React Icons
- Chart.js
- React Toastify
- Fetch API

AUTHENTICATION METHODS

Customer:
- Email OTP registration
- Secure login with session persistence
- Password validation
- Change password securely

Admin:
- Email/password login
- Forgot password via secret key
- Independent session handling

PREREQUISITES
- JDK 21+
- Node.js 18+
- PostgreSQL 12+
- Maven 3.6+
- Git

SETUP INSTRUCTIONS

Backend Setup:
1. Clone repository
2. Create PostgreSQL DB
3. Configure application.properties
4. Add environment variables in .env.properties
5. Run backend using:
mvn spring-boot:run
6. Create uploads folder

Frontend Setup:
1. Go to next-frontend
2. npm install
3. Add .env.local
4. npm run dev

ENVIRONMENT VARIABLES

Backend (.env.properties):
FRONTEND_URL=http://localhost:3000
BACKEND_URL=http://localhost:8080
EMAIL_USERNAME=your-email
EMAIL_PASSWORD=your-app-password
RAZORPAY_KEY_ID=...
RAZORPAY_KEY_SECRET=...

Frontend (.env.local):
NEXT_PUBLIC_BACKEND_URL=http://localhost:8080

DATABASE CONFIGURATION

DB setup:
CREATE DATABASE "spice-shop";

Update application.properties with credentials.

Tables auto-generated:
users, admins, spices, variants, packs, images, carts, cart_items, orders,
order_items, payments, addresses, spring_session, spring_session_attributes.

API ENDPOINTS

Customer Authentication:
POST /api/auth/send-otp
POST /api/auth/verify-otp
POST /api/auth/register
POST /api/auth/login
POST /api/auth/logout
GET /api/auth/check-session
POST /api/auth/change-password

Admin Authentication:
POST /api/admin/login
POST /api/admin/logout
GET /api/admin/profile
PUT /api/admin/update-profile/{id}
POST /api/admin/change-password
POST /api/admin/forgot-password/*

Products:
GET /api/spices
GET /api/spices/{id}
POST /api/spices
PUT /api/spices/{id}

DELETE /api/spices/{id}
PATCH /api/spices/{id}/availability

Cart:
GET /api/cart
POST /api/cart/items
PUT /api/cart/items/{itemId}
DELETE /api/cart/items/{itemId}

Orders:
POST /api/orders/place
GET /api/orders/history
GET /api/orders/{orderId}
GET /api/orders/all (admin)

Payments:
POST /api/payments/verify

PROJECT STRUCTURE
(simplified)

spice-shop/
spring-backend/
next-frontend/
assets/screenshots/
README.md

RUNNING THE APPLICATION

Backend:
mvn spring-boot:run

Frontend:
npm run dev

Production:
Backend -> mvn clean package
Frontend -> npm run build && npm start

ADDITIONAL NOTES

- Spring Session JDBC for user sessions
- Gmail SMTP for OTP & email notifications
- Razorpay integration for payments
- Secure BCrypt password hashing
- Image uploads supported

- CORS configured

SUPPORT
Email: nibin.joseph.career@gmail.com