Food-Delivery – Full Documentation

1. Introduction

Food-Delivery is a full-stack sample system for ordering and delivering food, built with **Spring Boot 3 (Java 17)**, **Angular 17**, and **MySQL 8**. Its purpose is to demonstrate:

- Role-based access (customer, delivery, admin)
- CRUD operations and business logic via REST API with JWT security
- A simple Angular SPA frontend that consumes backend services

---

2. Tech Stack and Versions

Layer	Technology	Version
Backend	Spring Boot	3.0.1
Persistence	Spring Data JPA + Hibernate	
Security	Spring Security 6, JWT (jjwt 0.9.1)	
Database	MySQL	8.x
Frontend	Angular CLI	17.x
Build Tools	Maven 3.6+, Node 18+/npm 9+	

---

3. Features

- **Register/Login** (JWT-based)
- **View food menu** – public access
- **Cart and create order** (customer)
- **Track your own orders** (customer)
- **View pending orders & update status** (delivery)
- **Manage foods and users** (admin)

---

4. Project Structure

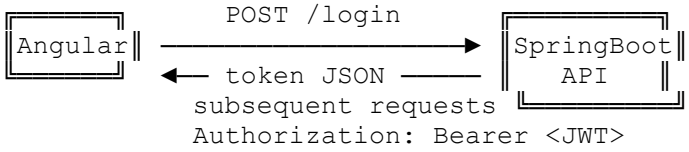
```
food-delivery-project/
├── food-delivery-backend/        # Spring Boot
│   ├── src/main/java/com/example/fooddelivery/
│   │   ├── config/       (SecurityConfig, JwtUtil, JwtFilter)
│   │   ├── controller/   (AuthController, FoodController, OrderController,
AdminController)
│   │   ├── model/        (User, Role, Food, Order, OrderItem, OrderStatus)
│   │   ├── repository/   (UserRepository, FoodRepository, OrderRepository)
│   │   └── service/      (OrderService, CustomUserDetailsService)
│   ├── src/main/resources/ (application.properties, data.sql)
```

```
│   └── pom.xml
└── food-delivery-frontend/       # Angular
    └── src/app/
        ├── components/ (login, register, food-list ...)
        ├── services/ (auth.service, food.service ...)
        ├── guards/ (auth, admin, delivery)
        ├── models/  (interfaces)
        └── app.module / routing
```

---

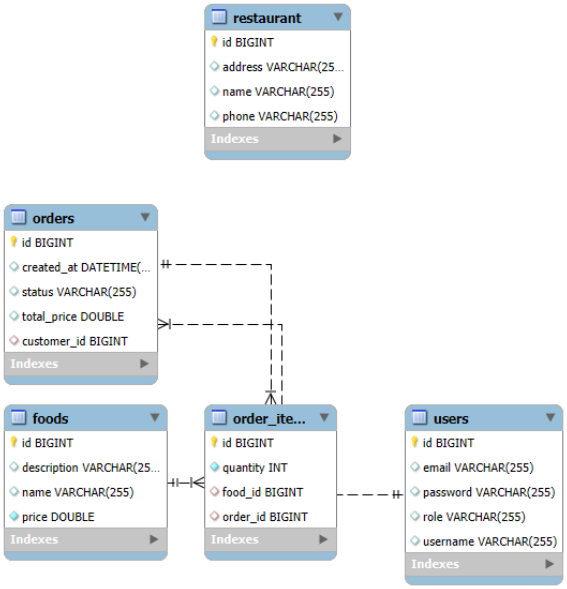
5. Architecture

- REST API on port **8080** (JSON-based)
- Angular SPA on port **4200** (in dev mode) – CORS enabled
- **JWT** (Bearer token) sent in `Authorization` header



---

6. Database Schema (ER Diagram)



- **User** (id, username, password, email, role)
- **Food** (id, name, description, price)
- **Order** (id, customer_id, status, total_price, created_at)
- **OrderItem** (id, order_id, food_id, quantity)

---

7. REST API Reference

7.1 Authentication

Method	Path	Body	Response
POST	`/api/auth/register`	`{username,email,password}`	`200 OK`
POST	`/api/auth/login`	`{username,password}`	`{token, role}`

7.2 Foods

Method	Path	Description	Role
GET	`/api/foods`	List all foods	Public
POST	`/api/foods`	Create new food	ADMIN

7.3 Orders

Method	Path	Description	Role
POST	`/api/orders/create`	Create order	CUSTOMER
GET	`/api/orders/my`	View my orders	CUSTOMER
GET	`/api/orders/pending`	List pending orders	DELIVERY
POST	`/api/orders/{id}/status?status=DELIVERED`	Update status	DELIVERY

7.4 Admin

Method	Path	Description
GET	`/api/admin/users`	List all users
POST	`/api/admin/users/{id}/role`	Change user role

---

8. Security (JWT Flow)

1. User logs in via `/login` and receives a **JWT token**.
2. Angular stores the token in `localStorage` and sends it with every request.
3. The **JwtFilter** validates the token and sets the security context.
4. Controllers use `@PreAuthorize` based on the user's role claim.

---

9. Setup and Run

1. Ensure **MySQL** is running on *localhost:3306* with root user and no password.
2. Run `FoodDeliveryApplication.java` → starts on port 8080.
3. In `food-delivery-frontend` folder, run:

```
4. npm install && npm start
```

Starts Angular dev server on port 4200.

5. Sample data (users, foods) is loaded from `data.sql`.

---

10. Sample Accounts

- **admin / admin**
- **testuser / customer**
- **courier1 / delivery**

---

11. Deployment (Production)

- Run `mvn clean package` → generates `food-delivery-backend-0.0.1-SNAPSHOT.jar`
- Angular: `ng build --prod` → serve `dist/` with Nginx, Apache or copy into Spring `static/` folder.

---

12. Future Improvements

- Payment integration (Stripe / PayPal)
- Upload food images (S3 or local storage)
- WebSocket notifications for order status
- Docker Compose setup (MySQL + backend + frontend)
- Add unit & e2e tests (JUnit 5, Cypress)