Cymbal Eats: Menu Service

A detailed overview of the Menu Service architecture, features, and operations.

Agenda

1. **Introduction:** What is the Menu Service?
2. **Core Responsibilities:** Its role in the Cymbal Eats ecosystem.
3. **Architecture:** A look under the hood.
4. **API Endpoints:** How other services interact with it.
5. **Data Management:** Database and storage details.
6. **Deployment & CI/CD:** From code to production.
7. **Q&A**

1. Introduction

The **Menu Service** is a critical backend microservice within the Cymbal Eats platform.

- **Purpose:** To manage all aspects of restaurant menus, including items, categories, pricing, and availability.
- **Owner:** Restaurant Operations Team
- **Tech Stack:** Java, Spring Boot, Maven, Docker

2. Core Responsibilities

The service is the single source of truth for all menu-related data.

- **CRUD Operations:** Create, Read, Update, and Delete menu items and categories.
- **Menu Publishing:** Manages the visibility of menus to customers.
- **Pricing Information:** Handles item prices, discounts, and special offers.
- **Real-time Availability:** Tracks which menu items are currently available or out of stock.

3. Architecture

Built as a standard, containerized Spring Boot application.

- **Language:** Java 11+
- **Framework:** Spring Boot
- **Build Tool:** Apache Maven (`pom.xml`)
- **Containerization:** Docker (`Dockerfile`)
- **Principles:** Follows RESTful API design and stateless service principles.

4. API Endpoints

Exposes a set of RESTful endpoints for interaction.

Method	Endpoint
`GET`	`/api/v1/menus/{restaurantId}`
`GET`	`/api/v1/menus/items/{itemId}`

5. Data Management

- **Primary Database:** The service connects to a dedicated database instance to persist all menu data.
- **Database Schema:** Includes tables for `Restaurants`, `Menus`, `MenuCategories`, and `MenuItems`.
- **Data Integrity:** Enforces relationships between tables (e.g., an item must belong to a category and a menu).
- **Caching:** Implements a caching layer to reduce database load and improve response times for frequently accessed menus.

6. Deployment & CI/CD

- **Container Registry:** Docker images are built and pushed to a container registry (e.g., Google Artifact Registry).
- **Deployment Target:** Deployed as a containerized workload, likely on Google Cloud Run or Google Kubernetes Engine (GKE).
- **CI/CD Pipeline:** The `pipelines/` directory suggests an automated CI/CD process that builds, tests, and deploys the service upon code changes.
- **Configuration:** Environment-specific settings are

Q&A

Thank you!

Any questions about the Menu Service?