Developer Guide — Pasindu

Restaurant Platform Engineering

Project Responsibility Summary

You are responsible for real-time restaurant operations and the AI food recommendation chat service:

- Redis-based Cart microservice
- Kitchen Display System (KDS)
- Waitstaff Fulfillment Console
- Feedback Service (MySQL)
- AI Assistant Service (FastAPI + MongoDB embeddings)
- Real-time WebSocket and SSE features
- MongoDB storage for embeddings and chat logs
- Integration with menu + user sessions

Your goal is to deliver a seamless real-time restaurant flow and smart menu assistant.

Technology Stack

- Backend: Spring Boot (Java), FastAPI (Python)
- Databases: MySQL, Redis, MongoDB
- Messaging: WebSockets / SSE
- Frontend: React, TailwindCSS, React Query

Database Responsibilities

MySQL Tables

```
kitchen_tickets(
 id, order_id, status, notes, created_at
)

serve_tasks(
 id, order_id, waiter_id, status, partial json, notes
)

feedback(
 id, order_id, food_rating, service_rating, comments, created_at
)
```

Redis Keys

```
cart:{sessionId} -> hash of itemId : qty
```

MongoDB Collections

```
embeddings { itemId, vector[], tags[] }
ai_logs { sessionId, prompt, response, timestamp }
```

Endpoints

Cart APIs

```
GET    /api/v1/sessions/{sessionId}/cart
POST   /api/v1/sessions/{sessionId}/cart/items
PATCH  /api/v1/sessions/{sessionId}/cart/items/{itemId}
DELETE /api/v1/sessions/{sessionId}/cart/items/{itemId}
POST   /api/v1/sessions/{sessionId}/cart/clear
```

Kitchen + Waitstaff

```
GET    /api/v1/kds/orders
PATCH  /api/v1/kds/orders/{orderId}/status
POST   /api/v1/kds/orders/{orderId}/notes

GET    /api/v1/fulfillment/orders
PATCH  /api/v1/fulfillment/orders/{orderId}/assign
PATCH  /api/v1/fulfillment/orders/{orderId}/serve
```

Feedback

```
POST /api/v1/feedback
GET  /api/v1/feedback/summary
```

AI (FastAPI)

```
POST /api/v1/ai/recommend
POST /api/v1/ai/chat
```

Backend Development Guidelines

Spring Boot

- Folder structure:

  ```
  controller/
  service/
  repository/
  dto/
  entity/
  config/
  ```

- Use `@Valid` and DTO validation
- Use Transactional for write operations
- Use RedisTemplate for cart operations
- WebSocket endpoint for kitchen/serving updates
- Publish events when order moves stages

Redis Cart Rules

- Key format: `cart:{sessionId}`
- Redis Hash for each cart
- Cart auto-expire after session ends

MongoDB Rules

- Store embeddings once per menu item
- Index `itemId` and `sessionId`
- Log AI chat sessions

AI Development Guidelines

FastAPI Rules

- Load model once globally
- Encode user message + menu embeddings
- Retrieve top matches by cosine similarity
- Cache embeddings in memory

Python Example

```
from sentence_transformers import SentenceTransformer
model = SentenceTransformer("all-MiniLM-L6-v2")
vector = model.encode("Spicy Chicken Pizza")
```

Frontend Development Guidelines

React Folder Structure

```
src/features/cart/
src/features/kds/
src/features/waitstaff/
src/features/feedback/
src/features/ai-chat/
```

UI Requirements

- Floating cart drawer like Uber Eats
- Live kitchen board with statuses
- Waitstaff task pickup UI
- Feedback popup after order served
- Chat widget with quick reply buttons

Definition of Done

- Real-time kitchen + waiter dashboards working
- AI suggestion system functional and fast
- Redis cart stable + scalable
- Users can submit feedback
- Clean UI and smooth UX