Menu Service - Design Document

Technology Stack

Current Implementation

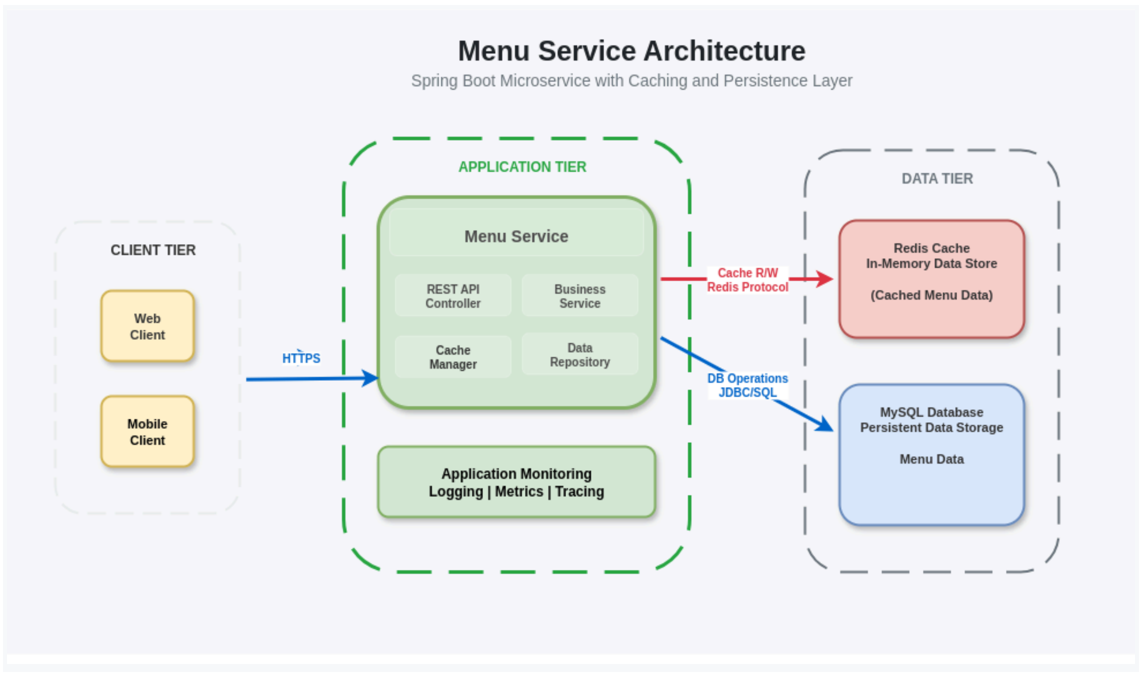
1. **Architecture:** Monolith (from the assignment perspective)
2. **Framework:** Java 17, Spring Boot 3.5.3
3. **Persistence:** MySQL (Dockerized for local)
4. **Local Caching:** Caffeine
5. **Distributed Caching:** Redis (Dockerized)
6. **API Layer:** REST APIs (versioned under /api/v1/...)
7. **Container:** Docker and Docker Compose

Responsibilities

1. **Spring Boot - Menu-Service:** REST API, business logic, caching, rate limiting
2. **MySQL:** Persistent storage for restaurants and menu items
3. **Redis:** Menu data caching, rate limiting counters

High-Level Architecture

Current Implementation Architecture



- Spring Boot Menu Service: Exposes REST APIs, handles business logic, and manages caching and DB access.
- Redis: Distributed cache for menus and menu items.
- MySQL: Persistent storage for restaurants, menus, and menu items.

Component Responsibilities

- Controller Layer: REST API endpoints, request validation
- Service Layer: Business logic, transaction management
- Repository Layer: JPA-based data access
- Mapper Layer: DTO ↔ Entity conversions
- Cache Layer: Redis (future: add Caffeine for L1)

Data Modeling

Database Schema (MySQL)

Restaurant Table

```
CREATE TABLE restaurant (
    id             BIGINT PRIMARY KEY AUTO_INCREMENT,
    name           VARCHAR(255) NOT NULL,
    address        VARCHAR(500) NOT NULL,
    phone          VARCHAR(15)  NOT NULL,
    email          VARCHAR(100) NOT NULL,
    contact_person VARCHAR(50)  NOT NULL,
    pincode        VARCHAR(10)  NOT NULL,
    city           VARCHAR(25)  NOT NULL,
    state          VARCHAR(25)  NOT NULL,
    image_url      VARCHAR(255),
    description    VARCHAR(500),
    cuisine        VARCHAR(50)  NOT NULL,
    status         VARCHAR(30)  NOT NULL,
    created_at     TIMESTAMP    NOT NULL,
    updated_at     TIMESTAMP    NOT NULL
);

CREATE INDEX idx_restaurant_city ON restaurant (city);
CREATE INDEX idx_restaurant_cuisine ON restaurant (cuisine);
```

Menu Table

```
CREATE TABLE menu (
    id            BIGINT PRIMARY KEY AUTO_INCREMENT,
    name          VARCHAR(50) NOT NULL,
    restaurant_id BIGINT      NOT NULL,
    status        VARCHAR(30) NOT NULL,
    description   VARCHAR(500),
    created_at    TIMESTAMP   NOT NULL,
    updated_at    TIMESTAMP   NOT NULL,
    FOREIGN KEY (restaurant_id) REFERENCES restaurant (id) ON DELETE CASCADE
);

CREATE INDEX idx_menu_restaurant ON menu (restaurant_id);
```

MenuItem Table

```
CREATE TABLE menu_item (
    id          BIGINT PRIMARY KEY AUTO_INCREMENT,
    name        VARCHAR(255)   NOT NULL,
    description VARCHAR(500),
    price       DECIMAL(10, 2) NOT NULL,
    status      VARCHAR(20)    NOT NULL,
    food_type   VARCHAR(30)    NOT NULL,
    category    VARCHAR(30)    NOT NULL,
    image_url   VARCHAR(255),
    menu_id     BIGINT         NOT NULL,
    created_at  TIMESTAMP      NOT NULL,
    updated_at  TIMESTAMP      NOT NULL,
    FOREIGN KEY (menu_id) REFERENCES menu (id) ON DELETE CASCADE
);

CREATE INDEX idx_item_menu ON menu_item (menu_id);
CREATE INDEX idx_item_category ON menu_item (category);
CREATE INDEX idx_item_food_type ON menu_item (food_type);
```

API Data Models

RestaurantRequestDto

```
{
  "name": "Hotel Goodluck Cafe",
  "address": "FC Road, Pune",
  "phone": "+91-9876543210",
  "email": "contact@goodluck.com",
  "contactPerson": "Jacky Chan",
  "pincode": "411010",
  "city": "Pune",
  "state": "Maharashtra",
  "imageUrl": "http://image.com/goodluck",
  "description": "Irani cafe",
  "cuisine": "INDIAN",
  "status": "ACTIVE",
  "menus": [ ... ]
}
```

MenuRequestDto

```
{
  "name": "Main Menu",
  "status": "ACTIVE",
  "description": "All day menu",
  "menuItems": [ ... ]
}
```

MenuItemRequestDto

```
{
  "name": "Bun Maska Chai",
  "description": "Bun Maskaa",
  "price": 49.99,
  "status": "AVAILABLE",
  "foodType": "VEG",
  "category": "Snacks",
  "imageUrl": "http://..."
}
```

RestaurantMenuResponseDto (Paginated)

```
{
  "restaurantId": 1,
  "restaurantName": "Good luck Cafe",
  "menus": [ ... ],
  "page": 0,
  "size": 10,
  "totalElements": 25,
  "totalPages": 3
}
```

Caching Strategy

- L2 Cache (Redis): Used for restaurant menus and menu items. TTL and invalidation to be implemented.
- L1 Cache (Caffeine, planned): For most frequently accessed data.
- Cache-Aside Pattern: Check cache first, then DB, update cache on DB read/write.
- Cache Invalidation: On menu/restaurant update or delete.

Rate Limiting, Pagination & Versioning

- Rate Limiting: (Planned) Redis-based, per-client/IP, configurable limits.
- Pagination: Supported in menu fetch APIs (page, size params).
- API Versioning: URL-based (`/api/v1/restaurant/...`).

Assumptions and Trade-offs

Assumptions

- Menu changes are infrequent compared to reads.
- Each restaurant has at least one menu.
- Restaurant data changes less frequently than menu items.
- Eventual consistency is acceptable for menu reads.

Trade-offs

- Cache vs. Consistency: Prioritise read performance and accept eventual consistency.
- Data Model Simplicity: Single menu per restaurant for now, extensible for multiple menus.
- Performance vs Resource Usage: Caching increases memory usage but improves latency.

Monitoring & Observability

- Spring Boot Actuator: Health, metrics, info endpoints.
- Logging: Structured logs, correlation IDs, and error stack traces.
- Metrics: Response times, error rates, cache hit/miss, DB/Redis health.
- Future: Prometheus, Grafana, ELK, distributed tracing.