Food Delivery Management Project

Technology Stack: Spring Boot (Java), MongoDB

Overview

This project simulates the backend of a food delivery system for a restaurant (RoofTop). It allows customers to place orders and manages the assignment of delivery drivers to orders, following specified allocation rules.

Major Functionalities

1. Driver Management

- **Add Drivers:**
  An admin can add delivery drivers using a REST API. Each driver has a name, status ("AVAILABLE" or "BUSY"), and a field indicating when they are next available for delivery.
- **List All Drivers:**
  View all drivers in the system using a REST API.

2. Order Management

- **Place Order:**
  Customers can place orders specifying the time and delivery travel time. The system automatically tries to assign the lowest-indexed available driver who can deliver at or before the desired order time.
- **Order Assignment Logic:**
  - If a driver is available at the order placement time, the order is assigned to them.
  - If multiple drivers are available, the one with the lowest index (or id) is chosen.
  - After taking an order, the driver becomes busy and won't be available until after delivering that order (their next available time is updated).
  - If all drivers are busy at the order's placement time, the order receives a status of **"No Food :-("** (order cannot be fulfilled).
- **View All Orders:**
  Retrieve a list of all orders in the system via a REST API.
- **View Single Order:**
  Retrieve details for a specific order using its unique id.

Key API Endpoints

Endpoint	Method	Description	Example Body/Usage
/drivers/add	POST	Add a new driver	{ "name": "D1" }
/drivers/all	GET	View all drivers	N/A
/drivers/{id}	GET	View one driver by id	N/A
/orders/place	POST	Place a new order and auto-assign driver	{ "orderTime": 4, "travelTime": 20 }
/orders/all	GET	View all orders	N/A
/orders/{id}	GET	View a single order	N/A

Note: All the requested parameters are in JSON format

Technologies Used

- **Spring Boot:** Java framework for building REST APIs.
- **MongoDB:** NoSQL database for persisting driver and order data.
- **Postman:** Tool for testing API endpoints manually.
- **IDE:** Intellij

Project Workflow

- **Entity Definition:**
  Data models for both OrderEntity and DriverEntity are defined to represent customer orders and delivery drivers, respectively.
- **Repository Layer:**
  OrderRepo and DriverRepo extend the MongoRepository interface, providing access to a rich set of prebuilt database operations for each entity.
- **Service Layer:**
  OrderService and DriverService are responsible for implementing the core business logic of the application. These services interact with the repositories to perform all required data access and manipulation.

- **Controller Layer:**
  OrderController and DriverController expose REST API endpoints that allow external clients (such as Postman or a frontend application) to interact with the system.
- **Request Flow:**
  When a request is made to an endpoint, it is first received by the corresponding controller. The controller delegates the underlying business logic to the appropriate service. The service, in turn, utilizes the repository layer to execute CRUD operations or custom queries as needed.

Prepared by:
Vedant Gurav