**Technical Task: Django Microservice for School Menu System**

### Objective

Develop a standalone Django microservice that manages the **School Menu System**, providing APIs to be consumed by your existing Laravel-based School Management System (SMS).

## 🧱 1**. Menu System Functional Sections**

Your Django app should support the following **sections/features**:

1. **Meal Types**
   * Define meal types like Breakfast, Lunch, Dinner
   * Support display ordering and activation/deactivation
2. **Nutrition Info** (Optional but recommended)
   * Define nutrition facts (calories, protein, fats, carbs)
   * Link nutrition profiles to menu items
3. **Menu Items**
   * Add and manage food items
   * Attach images and nutrition info
   * Categorize by meal type
4. **Menu Schedules**
   * Assign menu items to specific dates and meal types
   * Filter by school, class, section, academic year
   * Allow weekly and daily scheduling
5. **Reports/Analytics** (Optional)
   * Aggregate most-served items, menu coverage

## 2. **Django Models to Create**

Define models in menu/models.py:

* MealType
* NutritionInfo
* MenuItem
* MenuSchedule

Ensure each model includes school\_id, timestamps, and active flags.

# 3. Django REST API Endpoints

**Meal Types:**

* GET /api/v1/meal-types/
* POST /api/v1/meal-types/
* GET/PUT/DELETE /api/v1/meal-types/<id>/

**Menu Items:**

* GET /api/v1/menu-items/
* POST /api/v1/menu-items/
* GET/PUT/DELETE /api/v1/menu-items/<id>/

**Menu Schedule:**

* GET /api/v1/menu-schedule/
* POST /api/v1/menu-schedule/
* GET /api/v1/menu-schedule/<id>/
* GET /api/v1/menu-schedule/date/<date>/
* GET /api/v1/menu-schedule/week/<start\_date>/
* GET /api/v1/menu-schedule/class/<class\_id>/

**Nutrition Info:**

* GET /api/v1/nutrition-info/
* POST /api/v1/nutrition-info/
* GET /api/v1/nutrition-info/<id>/

# 4. Authentication & Permissions

### **Token Validation**

* Authenticate Laravel Bearer tokens via a /api/auth/validate-token endpoint (in Laravel).
* Use a Django custom authentication class LaravelPassportAuthentication.

### **Role-Based Access**

Create permission classes like:

* IsSuperAdmin role\_id=1
* IsAdmin role\_id=5
* IsStudent role\_id=2
* IsParent role\_id=3
* Use them to restrict viewsets and serializers accordingly.
* ( Hozircha Bosh oshpaz systemda mavjud emas(kelajakda HeadChief with role\_id=10 bilan yaratiladi).

# 5. Database Configuration

Use MySQL with UTF-8 support.

Example ENV:

DB\_NAME=

DB\_USER=

DB\_PASSWORD=

DB\_HOST=

**6. Django Project Structure**

school\_menu\_service/

├── menu/

│ ├── models.py

│ ├── views.py

│ ├── serializers.py

│ ├── urls.py

│ └── permissions.py

├── authentication/

│ └── auth.py

├── school\_menu/

│ └── settings.py

├── manage.py

## 7. Caching & Performance (Optional)

* Use Redis cache to store commonly fetched data like today's menu per school/class.
* Cache keys: menu\_schedule\_<school\_id>\_<date>

# 8. Error Handling & Monitoring

* Standardize success/error JSON responses
* Add a health check endpoint: GET /api/v1/health/

# 9. Deployment Plan

* Deploy Django using Docker
* Use docker-compose for Django + MySQL + Redis
* Configure CORS and API keys