Backend Developer - Coding Challenge

"LunchLog"

(Office Lunch Receipt Management and **Recommendation** System)

Objective

Develop a Django Rest Framework Backend application that allows users to manage their daily lunch receipts.

- Users can upload, edit, delete, and list receipts
- Additionally, the system should use AWS S3 for storing receipts, and a scheduled job should call the Google Places API to fetch and store information about the restaurants for future food recommendations

Requirements

1. Technical Requirements
   - Programming language: Python (version 3.x)
   - Backend framework: Django REST framework
   - Database: PostgreSQL with Django ORM

2. User Authentication
   - Implement user signup and login functionality with meaningful data

3. Receipt Management
   - Users can add receipts (image) with details including date, price, restaurant name, and address
   - The receipt image should be stored in AWS S3
   - Users can edit and delete receipts (OPTIONAL)
   - Users can list all receipts for any given month (OPTIONAL)

4. AWS S3 Integration
   - Use Django Storages to upload and store receipt images in AWS S3
   - Organize receipts in S3 using a folder structure based on user ID and date

5. Google Places API Integration
    - Implement a process (synchronous, asynchronous, or a scheduled job using AWS lambda) to fetch detailed information about restaurants from the Google Places API
    - Store meaningful information (Food type/ Restaurant name, cuisine etc.) from the API response into the database using an appropriate DB structure

6. Food Recommendation Endpoint
    - Implement an endpoint that takes a location/city and returns food recommendations based on the user's preferences using the stored restaurants/food/cuisine information already retrieved and looking up something similar using Google Places APIs

7. Tests
    - Write unit tests for the implemented features using pytest
    - Enough code coverage to test the application behavior in a standalone environment without any real API consumer involved

Instructions

1. Setup
    - Use Django and Django Rest Framework to implement the API
    - Ensure the application and Database is containerized using Docker
    - Provide instructions to run the application locally using Docker, perhaps with the help of a make file
    - Consider using poetry for the dependency management
    - It would be nice if the applicant is able to deploy the application using AWS Fargate/ECS/EC2, Cloudformation and AWS autopilot (OPTIONAL)

2. API Endpoints
    - /auth/signup/ - User signup
    - /auth/login/ - User login
    - /receipts/ - CRUD operations for receipts (Delete and Update are OPTIONAL)
    - /receipts/?month=<month>/ - List receipts for a specific month (OPTIONAL)
    - /recommendations/ - Get food recommendations for a given location

NOTE: Use Django's session authentication for user facing endpoints and Token authentication in case of a webhook to provide to a lambda or any scheduled job.

3. Database Models
    - User (Django's built-in user model but can be overridden as per requirements)
    - Receipt (with fields: date, price, restaurant name, address, receipt image URL etc)
    - Additionally, a DB model to store detailed and meaningful information from Google Places API)

    NOTE: While storing the information retrieved from the Google Places API, keep in mind that this info should play a part while fetching as good as possible recommendations for the user.

4. AWS S3 and Google Places API Integration
    - Configure django-storages and boto3 for AWS S3 integration. Alternatively local storage can also be used but S3 is preferred
    - Implement a scheduled job using Celery or AWS Lambda to fetch restaurant details from Google Places API and save them to the database (OPTIONAL)

5. Submission
    - Submit your code in a Git repository with a clear commit history
    - Provide the URL for the git repo
    - Include a README file with setup instructions and how to run the application
    - Code should be downloadable and runnable following the instructions

6. Evaluation
    - Correctness and completeness of the implemented features
    - Code quality and adherence to best practices
    - Proper use of Django Rest Framework and AWS services
    - Effective implementation of the Google Places API integration
    - Ability to write meaningful tests and documentation

Please reach out anytime if you do have questions. We wish you all success with your solution!