SDE API Round - Dining Reservation System

Statement for Candidates:

Below is the problem statement for your project.

1. Your task is to create APIs that support the requirements mentioned in Problem Statement
2. You have 3 hours to develop and submit the API. You'll need to submit the link to the repo (GitHub / bitbucket / gitlab etc) where you've pushed your commits.
3. Ensure that your repository includes a `README.md` file where you've mentioned how to setup, run and run tests (optional) for your code. When in doubt make a plausible assumption and state it explicitly in your `README.md` file.
4. You can use Google to find solutions, however, copying an existing project is not allowed.
5. Do not use ChatGPT / copilot or any tool that generates code for you, otherwise your submission will not be considered for evaluation.
6. Once completed, please make your submission here or on the link shared by the recruiter: https://forms.gle/LGAMftxhc9zXVrGW9
7. Feel free to use your own dummy data

Problem Statement

Hey there, Mr. X. You have been appointed to design a dining reservation system for Zomato, where users can browse different dining places and can book a specific place for a particular time slot.

Before booking, check if the slot is available. Also, give the user the next available slot in case of unavailability of the asked timings.

For eg: If a booking is made with the "start_time" as "2024-01-01T16:00:00Z" and "end_time" as "2024-01-01T17:00:00Z" then the "next_available_slot" for this dining place will be "2024-01-01T17:00:00Z".

Do not accept any bookings for an already booked slot. Make a robust and optimal system to handle multiple bookings simultaneously.

There is a Role Based Access provision and 2 types of users would exist :

1. Admin - can perform all operations like adding dining places, updating existing dining place details, etc.
2. Login users - can check all the dining places, make reservations, etc.

Tech Stack:

1. Any web server of your choice (Python Flask / Django, NodeJS / ExpressJS , Java/Springboot, etc)
2. Database: MySQL/PostgreSQL (Compulsory)

Requirements

1. **Register a User**
   Create an endpoint for registering a user.

```
[POST] /api/signup

Request Data : {
  "username": "example_user",
  "password": "example_password",
  "email": "user@example.com"
}

Response Data : {
  "status": "Account successfully created",
  "status_code": 200,
  "user_id": "123445"
}
```

2. **Login User**

   Provide the ability for the user to log into his account.

```
[POST] /api/login

Request Data : {
  "username": "example_user",
  "password": "example_password"
}

For successful login
Response Data : {
  "status": "Login successful",
  "status_code": 200,
  "user_id": "12345",
  "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9"
}

For failure
Response Data: {
  "status": "Incorrect username/password provided. Please retry",
  "status_code": 401
}
```

3. **Add a New Dining Place**

   An endpoint for the admin to add a new dining place.

```
[POST] /api/dining-place/create

Request Data : {
  "name": "Gatsby",
  "address": "HSR Layout",
  "phone_no": "9999999999",
  "website": "http://workindia.in/"
  "operational_hours": {
    "open_time": 08:00:00,
    "close_time": 23:00:00
  },
  "booked_slots": []
}

Response Data : {
  "message": "Gatsby added successfully",
  "place_id": "12345",
  "status_code": 200
}
```

   NOTE: Feel free to add CRUD Operations for users after implementing the basic flow.

4. **Search Dining Places by Name**

   An endpoint for the users to check all dining places whose title contains few specific keywords

```
[GET] /api/dining-place?name={search_query}

Request Data : {}

Params : {
  "search_query": str
}
```

```

Response Data : {
  "results": [
    {
      "place_id": "12345",
      "name": "Gatsby",
      "address": "HSR Layout",
      "phone_no": "9999999999",
      "website": "http://workindia.in/"
      "operational_hours": {
        "open_time": 08:00:00,
        "close_time": 23:00:00
      },
      "booked_slots": [
        {
          "start_time": 2023-01-01T12:00:00Z,
          "end_time": 2023-01-01T14:00:00Z
        },
        {
          "start_time": 2023-01-01T15:00:00Z,
          "end_time": 2023-01-01T16:00:00Z
        }
      ]
    },
    {
      "place_id": "12346",
      "name": "Reservoire",
      "address": "HSR Layout",
      "phone_no": "9999999999",
      "website": "http://workindia.in/"
      "operational_hours": {
        "open_time": 10:00:00,
        "close_time": 22:00:00
      },
      "booked_slots": [
        {
          "start_time": 2023-01-01T12:00:00Z,
          "end_time": 2023-01-01T14:00:00Z
        },
        {
          "start_time": 2023-01-01T15:00:00Z,
          "end_time": 2023-01-01T16:00:00Z
        }
      ]
    }
  ]
}
```

5. **Get Dining Place Availability**

   Get the availability of a particular dining place based on its id.

```
[GET] /api/dining-place/availability

Request Data : {}

Params : {
  "place_id": "12345",
  "start_time": 2023-01-01T16:00:00Z,
  "end_time": 2023-01-01T18:00:00Z
```

```
}

For available slots
Response Data : {
  "place_id": "98765",
  "name": "Gatsby",
  "phone_no": "9999999999",
  "available": true,
  "next_available_slot": null
}

For already booked slots
Response Data : {
  "place_id": "98765",
  "name": "Gatsby",
  "phone_no": "9999999999",
  "available": false,
  "next_available_slot": 2023-01-01T17:00:00Z   //Closest available start time
}
```

6. **Make a booking**

   An endpoint for the users to make a booking for a particular time slot. User ID is to be present in the request data and should be a valid user in order to make a booking. Take the user_id from the bearer token.

```
[POST] /api/dining-place/book

Headers : {
  "Authorization": "Bearer {token}"
}

Request Data : {
  "place_id": "12345",
  "start_time": 2023-01-02T12:00:00Z,
  "end_time": 2023-01-01T13:00:00Z
}

For available slots
Response Data : {
  "status": "Slot booked successfully",
  "status_code": 200
  "booking_id": "54321"
}

For already booked slots
Response Data : {
  "status": "Slot is not available at this moment, please try some other place",
  "status_code": 400
}
```

Mandatory requirement:

1. You need to protect all the admin API endpoints with an **API key** that will be known only to you and the admin so that no one can add false data to your system.
2. For booking a dining place, you need to send the **Authorization Token** received in the login endpoint.
3. You can assume there will be no half-hour bookings and all will be on hourly basis. For ex: 1, 2 hour slots are possible, but not 1.5 hours, etc.