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/GGibtv2aFR5SX1J99
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.

```
1   [POST] /api/login
2
3   Request Data : {
4     "username": "example_user",
5     "password": "example_password"
6   }
7
8   For successful login
9   Response Data : {
10    "status": "Login successful",
11    "status_code": 200,
12    "user_id": "12345",
13    "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9"
14  }
15
16  For failure
17  Response Data: {
18    "status": "Incorrect username/password provided. Please retry",
19    "status_code": 401
20  }
```

3. Add a New Dining Place

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

```
1   [POST] /api/dining-place/create
2
3   Request Data : {
4     "name": "Gatsby",
5     "address": "HSR Layout",
6     "phone_no": "9999999999",
7     "website": "http://workindia.in/"
8     "operational_hours": {
9       "open_time": 08:00:00,
10      "close_time": 23:00:00
11    },
12    "booked_slots": []
13  }
14
15  Response Data : {
16    "message": "Gatsby added successfully",
17    "place_id": "12345",
18    "status_code": 200
19  }
```

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

```
1   [GET] /api/dining-place?name={search_query}
2
3   Request Data : {}
4
5   Params : {
6     "search_query": str
7   }
```

```
 8
 9  Response Data : {
10    "results": [
11      {
12        "place_id": "12345",
13        "name": "Gatsby",
14        "address": "HSR Layout",
15        "phone_no": "9999999999",
16        "website": "http://workindia.in/"
17        "operational_hours": {
18          "open_time": 08:00:00,
19          "close_time": 23:00:00
20        },
21        "booked_slots": [
22          {
23            "start_time": 2023-01-01T12:00:00Z,
24            "end_time": 2023-01-01T14:00:00Z
25          },
26          {
27            "start_time": 2023-01-01T15:00:00Z,
28            "end_time": 2023-01-01T16:00:00Z
29          }
30        ]
31      },
32      {
33        "place_id": "12346",
34        "name": "Reservoire",
35        "address": "HSR Layout",
36        "phone_no": "9999999999",
37        "website": "http://workindia.in/"
38        "operational_hours": {
39          "open_time": 10:00:00,
40          "close_time": 22:00:00
41        },
42        "booked_slots": [
43          {
44            "start_time": 2023-01-01T12:00:00Z,
45            "end_time": 2023-01-01T14:00:00Z
46          },
47          {
48            "start_time": 2023-01-01T15:00:00Z,
49            "end_time": 2023-01-01T16:00:00Z
50          }
51        ]
52      }
53    ]
54  }
```

5. **Get Dining Place Availability**

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

```
1  [GET] /api/dining-place/availability
2
3  Request Data : {}
4
5  Params : {
6    "place_id": "12345",
7    "start_time": 2023-01-01T16:00:00Z,
8    "end_time": 2023-01-01T18:00:00Z
```

```
 9  }
10
11  For available slots
12  Response Data : {
13    "place_id": "98765",
14    "name": "Gatsby",
15    "phone_no": "9999999999",
16    "available": true,
17    "next_available_slot": null
18  }
19
20  For already booked slots
21  Response Data : {
22    "place_id": "98765",
23    "name": "Gatsby",
24    "phone_no": "9999999999",
25    "available": false,
26    "next_available_slot": 2023-01-01T17:00:00Z   //Closest available start time
27  }
```

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.

```
 1  [POST] /api/dining-place/book
 2
 3  Headers : {
 4    "Authorization": "Bearer {token}"
 5  }
 6
 7  Request Data : {
 8    "place_id": "12345",
 9    "start_time": 2023-01-02T12:00:00Z,
10    "end_time": 2023-01-01T13:00:00Z
11  }
12
13  For available slots
14  Response Data : {
15    "status": "Slot booked successfully",
16    "status_code": 200
17    "booking_id": "54321"
18  }
19
20  For already booked slots
21  Response Data : {
22    "status": "Slot is not available at this moment, please try some other place",
23    "status_code": 400
24  }
```

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.