Programming Exercise: Reservation System

Requirements

Develop restaurant reservation application called BookNow. When user makes a reservation, table should be blocked for 2 hours. There are only 20 tables and reservation API should cater to max reservation count. The restaurant is open from 11 AM to 7 PM only.

Application should expose following endpoints:

- **GET /v1/availableSlots{date} - Retrieves all available slots for a given date**

```
[
 {
  "tableName": "table1",
  "availableDate": "2016-09-30",
  "availableTime": "5PM-7PM"
 },
 {
  "tableName": "table1",
  "availableDate": "2016-09-30",
  "availableTime": "3PM-5PM"
 },
 {
  "tableName": "table2",
  "availableDate": "2016-09-29",
  "availableTime": "3PM-5PM"
 }
]
```

- **GET /v1/reservations{date} - Retrieves all reservations for a given date.**

```
[
 {
  "id":"04000000",
  "tableName": "table1",
  "reservationDate": "2016-09-30",
```

```
  "reservationTime": "5PM-7PM",
  "name": "Ram",
  "contact": "040000"
 },
 {
  "id":"04000001",
  "tableName": "table2",
  "reservationDate": "2016-09-30",
  "reservationTime": "3PM-5PM",
  "name": "Anna",
  "contact": "040001"
 }
]
```


- **GET /v1/reservations/{id} -Retrieves a specific reservation based on ID**

  Response:


```
{
 "id": "04000000",
 "name": "Ram Manohar",
 "contact": "04000000",
 "reservationDate": "2016-09-30",
 "reservationTime": "5PM-7PM",
 "tableName": "table1"
}
```


- **POST /v1/reservations - Creates a new reservation**

  Request:


```
{
 "name": "Ram Manohar",
 "contact": "04000000",
 "reservationDate": "2016-09-30",
 "reservationTime": "5PM-7PM",
 "tableName": "table1"
}
```


  Response :Successful


```
{
  "id": "04000000",
```

"status" : "BOOKED"
}

Response: Unavailable

{
"id": "*0*",
"status" : "UNAVAILABLE"
}

- **PUT /v1/reservations/{id} - Updates an existing reservation.**

{
"id": "*04000000*",
"name": "Ram Manohar",
"contact": "04000000",
"reservationDate": "2016-09-30",
"reservationTime": "5PM-7PM",
"tableName": "table2"
}

Response :Successful

{
"id": "*04000000*",
"status" : "BOOKED"
}

Response: Unavailable

{
"id": "*0*",
"status" : "UNAVAILABLE"
}

- **DELETE /v1/reservations/{id} - Removes a specific reservation based on ID**

Response :Successful

{
"id": "*0*",

```
  "status" : "UNRESERVED"
}
```

Technology Stack

- Spring Boot
- Spring Data JPA
- Maven / Gradle

Additional Considerations

- The solution should handle concurrent reservation issues when multiple users try to reserve same table for the same day, same slot.

Deliverables

- The solution submitted should include structure, source code, configuration and any tests or test code you deem necessary
- Solve the problem as though it were "production level" code.
- It’s not required to provide any graphical interface.
- Expose rest end-points to preform operations.