**Rate Management system(RMS) in Logistic domain**

**Brief overview**: Logistics company XYZ charges customers for shipping goods. Charges are calculated based on rate amt and surcharge amt

 - Design a table "RATE" with fields in MySQL

                - RateId <Long>, RateDescription <String>, RateEffectiveDate <Date>, RateExpirationDate <Date>, Amount <Integer>

                - RateId is the primary key

                - All are Not Null fields expect RateDescription

- Create a **Spring Boot application with JPA** that exposes below endpoints:

**Search Rate**:

                                - Call "<https://surcharge.free.beeceptor.com/surcharge>" to fetch the VAT surcharge

                                - Search the DB with RateId and fetch the corresponding record

                                - Output all the fields of the Rate table and surcharge details in JSON format

                                - If not found, throw status code 404 and description - "RateId not found in RMS" as response

**Add Rate**

                                - Provide valid data and it should successfully store data in DB

                                - Any runtime exception while saving record, throw status code 500 and description "Internal server error. Please contact admin"

**Update Rate**

                                - Provide valid data and it should successfully store data in DB

                                - Any runtime exception while saving record, throw status code 500 and description "Internal server error. Please contact admin"

**Delete Rate**

                                - Delete Rate record based on RateId

                                - If not found, throw status code 404 and description - "RateId not found in RMS" as response

- Write **Unit Testing using JUnit & Mockito** for controller, service, repository layers (Use H2 DB)

 - Implement **Basic authentication/ Oauth2 using Okta** (Free Developer account) in Spring Security

- Expose all metrics of **Spring Actuator** endpoints

 - Implement **Hystrix** for failure scenarios like DB instance down/ API endpoint times out/ down

 - Configure **logback** as the logging framework with rollover logic i.e. rollover once log file size is 5MB

**Protocol to be used**: HTTP/ RESTful

**Data interchange format to be used**: JSON

**Technologies to be used**

**Framework** - Sprint Boot >2.0, Spring cloud Hoxton SR2 or greater

**ORM**: Spring JPA/Hibernate (Compatible)

**DB**: >MYSQL5.7

**Build**: Gradle 6 and above