# 1 COMPONENT-3(END-TO-END) RUBRICS/EXPECTED DELIVERABLES

## 1.1 REST API (PRODUCTS & FRAMEWORKS -> COMPUTE & INTEGRATION):

**a.** Use Spring Boot to version and implement the REST endpoints.

**b.** Implement HTTP methods like GET, POST, PUT, DELETE, PATCH to implement RESTful resources:

| | | |
|---|---|---|
| **POST** | **/api/v1.0/tweets/register** | **Register as new user** |
| **GET** | **/api/v1.0/tweets/login** | **Login** |
| **GET** | **/api/v1.0/tweets/<username>/forgot** | **Forgot password** |
| **GET** | **/api/v1.0/tweets/all** | **Get all tweets** |
| **GET** | **/api/v1.0/tweets/users/all** | **Get all users** |
| **GET** | **/api/v/1.0/tweets/user/search/username*** | **Search by username** |
| **GET** | **/api/v1.0/tweets/username** | **Get all tweets of user** |
| **POST** | **/api/v1.0/tweets/<username>/add** | **Post new tweet** |
| **PUT** | **/api/v1.0/tweets/<username>/update/<id>** | **Update tweet** |
| **DELETE** | **/api/v1.0/tweets/<username>/delete/<id>** | **Delete tweet** |
| **PUT** | **/api/v1.0/tweets/<username>/like/<id>** | **Like tweet** |
| **POST** | **/api/v1.0/tweets/<username>/reply/<id>** | **Reply to tweet** |

**c.** ***username may be partial or complete username**

**d.** Use necessary configuration in place for REST API in application.properties or bootstrap.properties or application.yml; whichever is applicable.

e. Package Structure for Spring Boot Project will be like com.tweetapp.* with proper naming conventions for package and beans.

f. Use configuration class annotated with @Configuration and @Service for business layer.

g. Use constructor-based dependency injection in few classes and setter-based dependency injection in few classes.

h. Follow Spring Bean Naming Conventions

## 1.2 DATABASE (PRODUCTS & FRAMEWORKS -> DATABASE & STORAGE):

1. As an application developer:
    a. Implement ORM with Spring Data MongoRepositoryand MongoDB. For complex and custom queries, create custom methods and use @Query, Aggregations (AggregationOperation, MatchOperation, AggregationResults), implementation of MongoTemplateetc as necessary.
    b. Have necessary configuration in place for REST API in application.properties or bootstrap.properties or application.yml OR Java based configuration; whichever is applicable.

## 1.3 Maven (Tooling):

1. As an application developer:
    a. Create the spring boot project using Maven CLI
    b. Generate Surefire test reports and share it as a part of deliverables
    c. Using Maven CLI generate the project documentation, and share it as a part of deliverables

## 1.4 Messaging (Products & Frameworks -> Compute & Integration):

1. As an application developer:

    a. Have a centralized logging system
    b. Be able to communicate using a messaging infrastructure.
    c. Use Kafka/RabbitMQ/ActiveMQ/KubeMQTemplate for communication with Springboot and topics in Kafka/RabbitMQ/ActiveMQ/KubeMQ.
    d. Use Kafka/RabbitMQ/ActiveMQ/KubeMQ for messaging infrastructure and implement producers to write messages/tweets to topic and consumers to read messages/tweets from topic.
    e. Configure Springboot app to log all logging messages to Kafka/RabbitMQ/ActiveMQ/KubeMQ.
    f. Configure all Kafka/RabbitMQ/ActiveMQ/KubeMQ related configuration needed for Spring Boot in *.properties or *.yml file.

## 1.5 Log/ Monitoring (Products & Frameworks -> Governance & Tooling):

1. As an application developer:

    a. Containerize the complete application, which includes front-end, middleware and Kafka/RabbitMQ/ActiveMQ/KubeMQ (consumers and producers) using docker and Dockerfile.
    b. Use .dockerignore as necessary to avoid containerizing un-necessary packages.
    c. Integrate Spring Boot Actuator with Prometheus and Grafana to monitor middleware.
    d. Implement logs with logstash.
    e. Open the preconfigured Logstash in Kibana and check if it successfully connect to Elasticsearch Server.

## 1.6 Debugging & Troubleshooting

1. Generate bug report & error logs - Report must be linkedwith final deliverableswhich should also suggest the resolutionfor the encountered bugs and errors.

## 1.7 Code Quality

1. Associates should have written efficient code within acceptable limits of time and space complexity
2. Associates should have written clean code that is readable
3. Associates should have written testable code
4. Associate should have used the required Code Analyzer to ensure code quality and standard code style

### 1.8 Secured Coding

1. Associates should have applied basic secure coding principles
2. Associates should have avoided deprecated functions
3. Associates should have used Checkmarx/Veracode as the secured coding tool

# 2 Platform

## 2.1 Compute

1. Use ECS CLI (as an alternative to AWS Management Console) for container management and deployment of spring boot application. You should be able to explain and demonstrate the same in interview.
2. Use NoSQL instance of AWS DynamoDB/Aurora(SQL) as a database for the Tweet Application

## 2.2 Compute, Identity & Compliance, Security& Content Delivery

1. Use AWS Lambda and AWS Aurora to build a backend process for handling requests for Tweet App.
2. Use Serverless Java Container using AWS ECS and run the tweet app created with Spring Boot inside AWS Lambda.
3. Use Amazon API Gateway to expose the lambda functions built in the previous step to be accessible on public internet.
4. Use AWS ELB to configure the auto-scaling container instances.
5. Configure AWS SNS to issue messages whenever a ELB scales-up and scale-down container instances

Note – Minimum two rest endpoints should be hosted in cloud

# 3 Methodology

## 3.1 Agile

1. As an application developer, use project management tool along to update progress as you start implementing solution.
2. As an application developer, the scope of discussion with mentor is limited to:
    a. Q/A
    b. New Ideas, New feature implementations and estimation.
    c. Any development related challenges
    d. Skill Gaps
    e. Any other pointers key to UI/UX and Middleware Development