1. **Functional Requirements**

|  |  |
| --- | --- |
| Requirement | Achieved? |
| *Business requirement:* |  |
| • Build a metric TO imperial(and vice versa) conversion system. | Yes |
| • Please include 3 conversions (must include temperature) | Yes |
| *Technical requirements:* |  |
| • Create a Spring Boot application using rest API’s to do conversions | Yes |
| •  Ensure that you have proper Unit and Integration testing that accompany the project. | Unit testing implemented, Integration test are written |
| •  Postman/Jmeter project with your RestAPI's. | Yes |
| •  Upload your project into your own GIT repository and share the project with us. | Yes |
| •  The project must be able to run in its own Docker environment. | Yes |
| •  Build system must use Maven. | Yes |
| •  Use Java 8+ | Yes |

1. Authentication & Authorisation of users using OAuth 2.X (SSO / Token based authorization using JWT) along with Spring security
2. Validating of input data unit types on UI.
3. Non Functional Requirements
4. Our service needs to be highly available
5. The system should be highly reliable
6. Ensure multiple conversion at same time, with data accuracy
7. Must be highly available & durable
8. Hosting of app along with all security - AWS
9. Site reliability to 99.9 percent -
10. Performance of application (container based Jmeter env)
11. Security of app (container based pentest env)
12. Scalability
13. Maintainability
14. Daily foot fall of 1000 users with 10 concurrent users
15. Browser support..Chrome
16. Architecture - Define Components - tech stack & HLD diagram

Tech Stack -

o   ReactJS, Java17, Spring boot, Microservice, AWS

o   TDD/BDD-based development

Tools -

o   SonarQube (Code review)

o   Selenium, TestNG, Junit (UI & Unit testing)

o   Kubernetes/terraform (Containerization)

o   Pentest (Penetration testing)

o   JMeter (Performance Testing)

o   Git (version control, branching)

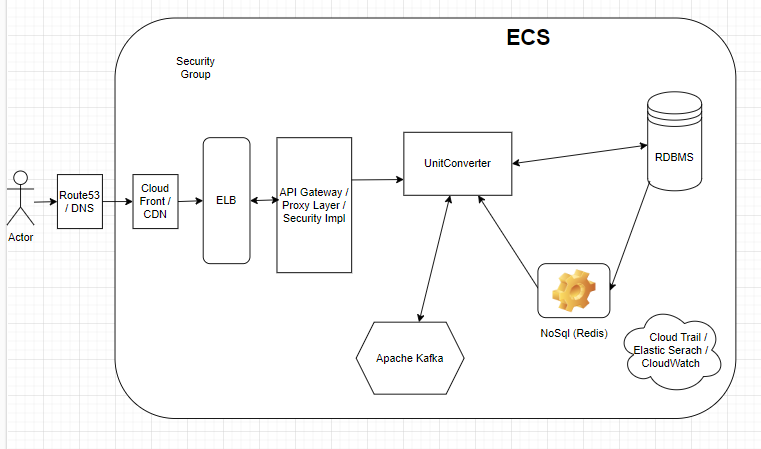
o   Bitbucket (CI/CD)

o   AWS CodeDeploy

o   Elasti search (Kibana) for logs

o   AWS Cloud Trail

o   AWS Cloud Watch



1. Production Design - Infra needed

o   Brancing - Main, Home, Dev, hotfix, Test, UAT/Staging, Production

o   Envirenment - Dev, Test, UAT, Prod

o   Cloud - AWS ECS or Serverless-lamdas

o   Continues Delivery and Deployment - Git & Bitbucket

1. Milestones

o     Infra (DevOps) setup

o     DB schema design setup

o     Frontend development

o     Backend development

o     Quality assurance testing

o     User acceptance testing

o     deployment setup

1. Team Structure and Timelines as per milestones

Team Structure

1. Reach JS developer - 1
2. Spring boot/Microservice with cloud developer - 2
3. SDET - with automation -1
4. DevOps - 1

Timelines as per milestones

o     Infra devops setup - 15 days

o     database schema design with storage capacity planning in whole - 10 days

o     frontend development - 50 days

o     backend development - 60 days

o     deployment setup - 15 days

**Github URL:** [**https://github.com/sanjuvds/MyAssignment/tree/master**](https://github.com/sanjuvds/MyAssignment/tree/master)

**The conversion which are achieved as part of this implementation**

|  |  |
| --- | --- |
| **From** | **To** |
| Kilogram | pound |
| pound | Kilogram |
| Meter | Kilometer |
| Kilometer | Meter |
| mile | Kilometer |
| Kilometer | mile |
| meter | mile |
| mile | meter |
| FAHRENHEIT | CELSIUS |
| CELSIUS | FAHRENHEIT |

**Variation of units and their conversion applicable in functionality**

Kilos = kilos, kilo, kilogram, kgs, kilograms

METRE = meter, meters, metres, metre

FAHRENHEIT = °F, Fahrenheit, farenheit

CELSIUS = °C, celsius

POUND = pounds, lbs, pound

MILE = mile, ml, miles, mis

KILOMETRE = kilometers, kilometers, kms, kilometer, kilometer

**The Java project created using following :**

Java version : **17**

Project type: **Maven**

Dependency : **Web**

Spring boot: **3.1.1**

Junit **: 4**

**Scenario which are yet to implement:**

1. Null check on ‘**number**’ field and getting user friendly message
2. Entering alphanumeric or garbage data in ‘**To**’ and ‘**From**’ field generating the exception but getting user friendly message in not yet implemented

**Note: The application can only be tested using postman**

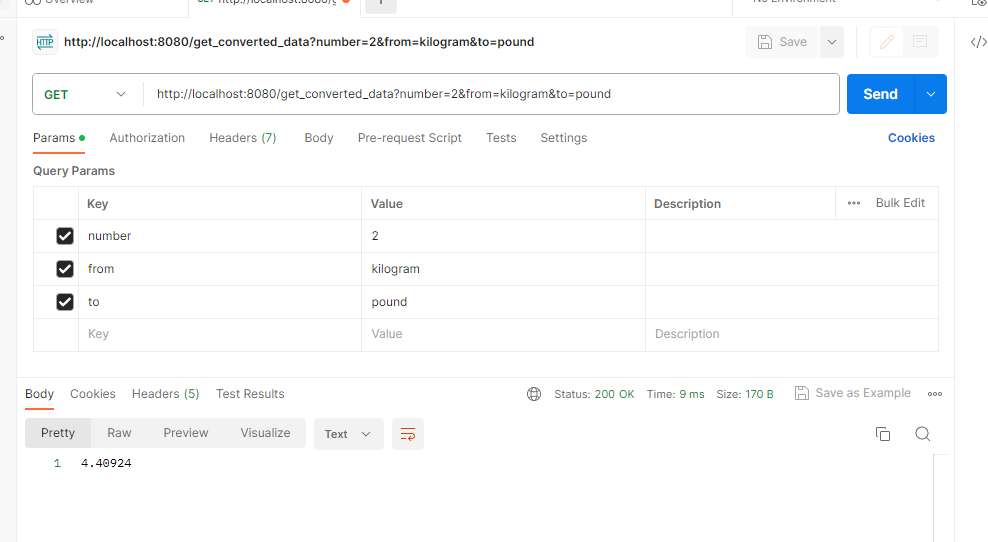
**The sample URL for testing the application on local**

**URL :** <http://localhost:8080/get_converted_data?number=2&from=kilogram&to=pound>

Number= 2,

From = kilogram,

To = pound

****

**Unit test case written and implemented for this functionality**

* check\_number\_not\_null
* check\_fromvalue\_not\_null
* check\_tovalue\_not\_null
* check\_fromvalue\_not\_blank
* check\_tovalue\_not\_blank
* varify\_conversion\_of\_kilogram\_to\_pound
* varify\_conversion\_of\_FAHRENHEIT\_to\_CELSIUS
* varify\_conversion\_of\_KILOMETRE\_to\_MILE
* varify\_conversion\_of\_pound\_to\_kilogram
* varify\_conversion\_of\_CELSIUS\_to\_FAHRENHEIT
* varify\_conversion\_of\_MILE\_to\_KILOMETRE

**Building docker Image :** Docker build –t springboot-docker-metrictoimperial-demo:latest .

**The image created :**

REPOSITORY TAG IMAGE ID CREATED SIZE

springboot-docker-metrictoimperial-demo latest a954972d8906 5 minutes ago 673MB

**Running the docker image with diff port :**

docker run -p 8081:8080 springboot-docker-metrictoimperial-demo

**URL for testing the docker image based application:**

http://localhost:8081/get\_converted\_data?number=2&from=kilogram&to=pound