Day 1

**14 Apr. 24**

Capstone Project

**Domain: Food Delivery**

**Domain: Entertainment**

**Domain: Healthcare**

**Domain: Travel**

Frontend 🡪 angular (HTML, CSS, Bootstrap , JavaScript (Typescript))

Backend -🡪 spring boot rest api

Database 🡪 mysql

Docker and Docker compose

Jenkins

AWS 🡪 EC2 instance

Two actor or users

Admin User --🡪 Sign In (no signup for admin from frontend technology)

[admin@gmail.com](mailto:admin@gmail.com) and admin@123

After admin login : open the admin dashboard

Admin : Food -🡪 Menu details or food details.(CRUD Operation )

MenuItems or FoodItems

FID 🡪 PK

FoodName 🡪

Typeoffood

Qty -🡪

Price

Image

Admin can view all order details.

Orders

OID (PK)

CName

FID

Price

Status (pending, processing, delivered / cancel)

Customer or Normal User -🡪 Signup and Signin

After customer or normal user 🡪 open the customer dashboard

Customer can view all food items and they can place the order for that specific food.

Customer can view its own order details.

**Entertainment**

**Admin**

**Add the movies details (CRUD Operation)**

**Customer**

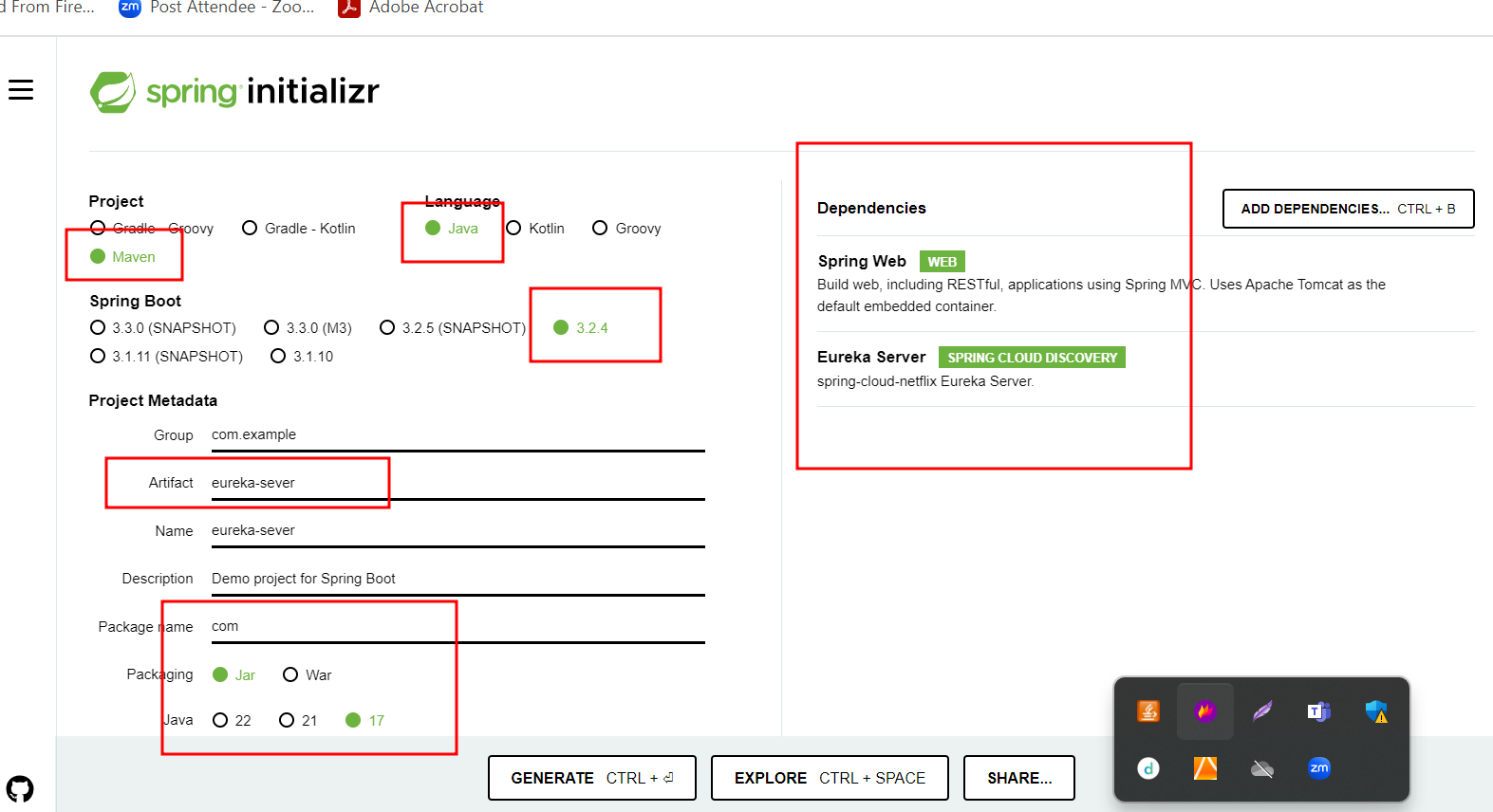
**Customer can view movies details and book the ticket.**

**Backend**

**Spring boot**

1. **Eureka Server**
2. **Backend-micro-service-app**

**Eureka Server**



**Application.properties files**

spring.application.name=eureka-sever

server.port=8761

eureka.client.register-with-eureka=false

eureka.client.fetch-registry=false

**Main class**

**package com;**

**import org.springframework.boot.SpringApplication;**

**import org.springframework.boot.autoconfigure.SpringBootApplication;**

**import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;**

**@SpringBootApplication**

**@EnableEurekaServer**

**public class EurekaSeverApplication {**

**public static void main(String[] args) {**

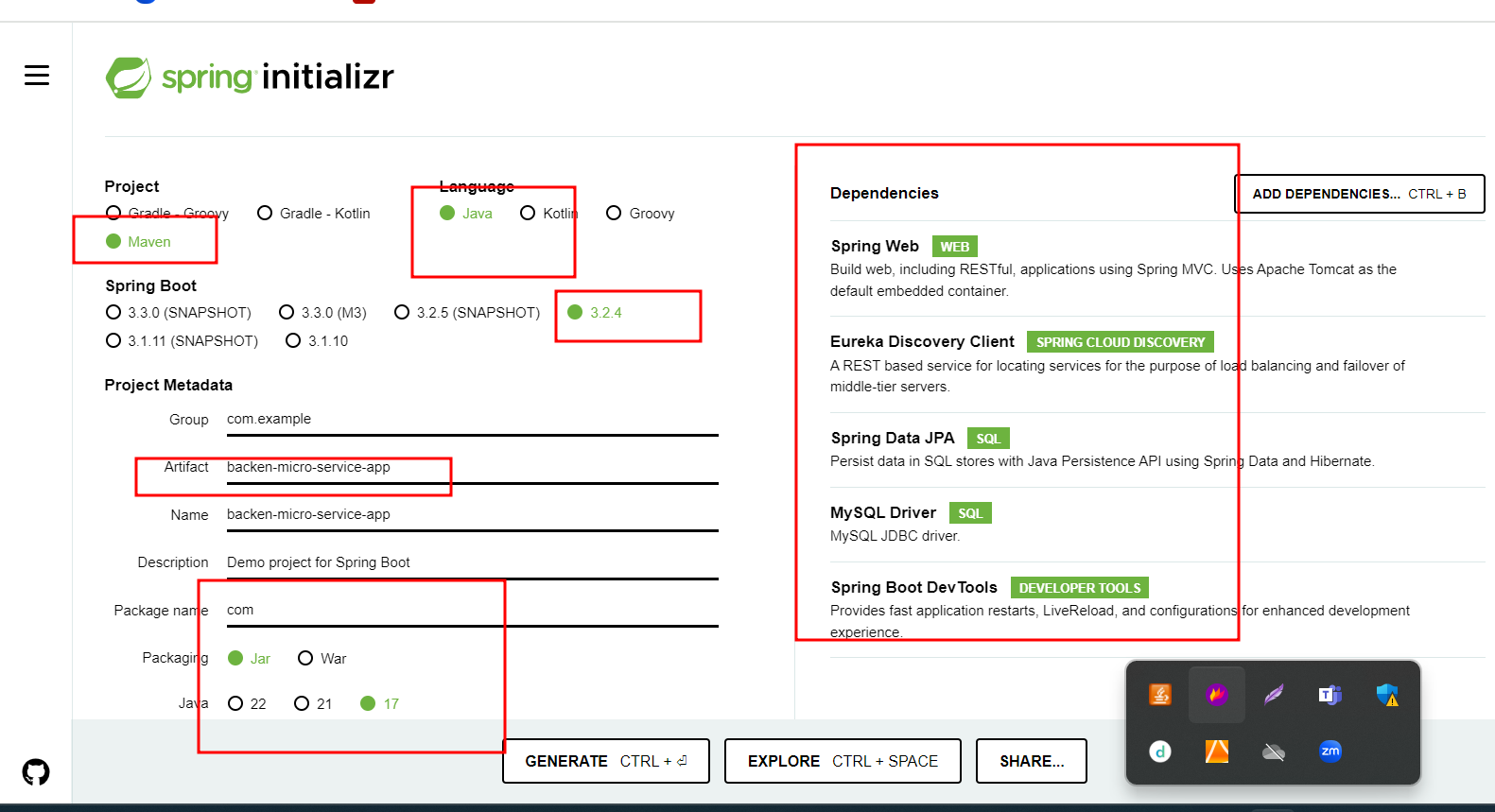
**SpringApplication.run(EurekaSeverApplication.class, args);**

**System.err.println("Eureka Server running on port number 8761");**

**}**

**}**

**We create micro service projects**



**Please refer the eclipse IDE code for micro service**

**Frontend**

**ng new frontend-app**

**ng g c login**

**ng g c signup**

**ng g class login**

**ng g s login**

**ng g c admindashboard**

**ng g c customerdashboard**

**has been blocked by CORS policy**

**CORS : Cross Origin Resource Sharing**

**Front end 🡪 running on port number 4200**

**Backend 🡪 running on port number 9090**

**Two domain going to communicate with each other using http proto col with browser.**

Day 2

**21 Apr. 24**

Capstone Project

1. **Create jar file for both the projects (eureka server and micro service). Using eclipse run with maven install option** 
   1. **Create the jar file for eureka server**
   2. **Create the jar file for micro service project**
2. **Build the angular frontend project using ng build command.**
3. **Then create Dockerfile for frontend and backend project.**
4. **Then create docker-compose file to run four container** 
   1. **Eureka server**
   2. **Micro service**
   3. **Mysql sever container**
   4. **Angular container**
5. **In application.properties file please provide mysql container details rather than local database information.**
6. **Please remove testing dependencies from micro service project ie pom.xml file.**
7. **Then once again create jar file.**
8. **Test this docker-compose file locally.** 
   1. **docker-compose up --build -d**
   2. **using docker ps check all running container**
   3. **then open url as** [**http://localhost:80**](http://localhost:80)
9. **from both project open .gitignore file and remove dist(from frontend) as well as target(backend spring boot) path.**
10. **Jenkinsfile which is responsible to run the docker-compose file**
11. **then create EC2 instance in AWS cloud with 2 CPU with 4gibram**
12. **then in frontend project provide public ip address of ec2 instance in service class means replace localhost by ip address and build angular project once again.**
13. **Then push fresh code in github account.**
14. **installed required software please refer EC2 instance plugin file.**
15. **git, java, jenkin, docker and docker-compose** 
    1. **install git** 
       1. **sudo yum install git -y**
    2. **install java** 
       1. **sudo yum install java-21**
    3. **jenkin install** 
       1. **sudo wget -O /etc/yum.repos.d/jenkins.repo** [**https://pkg.jenkins.io/redhat/jenkins.repo**](https://pkg.jenkins.io/redhat/jenkins.repo)
       2. **sudo rpm --import** [**https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key**](https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key)
       3. **sudo yum install jenkins**
       4. **sudo service jenkins start**
       5. **sudo systemctl status jenkins**
    4. **install docker** 
       1. **sudo yum install docker**
       2. **sudo service docker start**
       3. **sudo docker images**
    5. **install docker-compose** 
       1. **sudo curl -L https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m) -o /usr/local/bin/docker-compose**
       2. **sudo chmod +x /usr/local/bin/docker-compose**
       3. **sudo docker-compose –version**
    6. **if we want to run docker-compose in ec2 instance run below commands.** 
       1. **sudo usermod -a -G docker jenkins**
       2. **sudo usermod -a -G docker ec2-user**
       3. **sudo chmod 777 /var/run/docker.sock**
       4. **sudo service jenkins restart**
16. **Now we need to open Jenkins dashboard using your EC2 instance public ip address with port number 8080** 
    1. [**http://publicipaddress:8080**](http://publicipaddress:8080)
    2. **It will ask the password please run below command to get the password.**
    3. **sudo cat /var/lib/jenkins/secrets/initialAdminPassword**
    4. **please install all suggested plugin**
17. **Then create jenkin pipeline job which is responsible to run the docker-compose file** 
    1. **New items**
    2. **Provide job name and type must be pipeline job.**
    3. **Description optional**
    4. **Pipeline option** 
       1. **Pipeline script from scm**
       2. **Scm is git**
       3. **Repository URL please provide your remote repository URL which contains backend, frontend, docker-compose file and jenkinsfile**
       4. **Please verify branch name it may be master or main.**
       5. **Please verify Jenkinsfile with case sensitive.**
       6. **Apply and save then**
       7. **Build it.**
       8. **May be success or failure.**
18. **Open EC2 instance terminal and write the command as sudo docker ps**
19. **Then open** [**http://publicipaddress:80**](http://publicipaddress:80)