**Medicare Project Documentation**

**Submitted by:** Anil Kumar Verma

**DESCRIPTION**

Create a dynamic and responsive Java e-healthcare web application for ordering medicines of different categories.

**Background of the problem statement:**

Medicare is a company that supplies medicines and a couple of other healthcare essentials at an affordable price. It was established in 2012 in Delhi, India. It had been serving fine all these years, however, the business analysts noticed a decline in sales since 2017. They found out that online ordering of medicines with companies, such as 100mg and mfine are gaining more profits by eliminating middlemen from the equation. As a result, the team decided to hire a Full Stack developer to develop a healthcare web application with a rich and user-friendly interface.  
You are hired as the Full Stack Java developer and are asked to develop the web application. The management team has provided you with the requirements and their business model so that you can easily arrange different components of the application.

**Features of the application:**

1. Registration
2. Login
3. Payment gateway
4. Searching
5. Filtering
6. Sorting
7. Dynamic data
8. Responsive and compatible with different devices

**Technologies used:**

1. Database management: MySQL and Oracle
2. Backend logic: Java programming, NodeJS
3. Frontend development: JSP, Angular, Bootstrap, HTML/CSS, and Javascript
4. Automation and testing technologies: Selenium, Jasmine (frontend testing), and TestNG
5. DevOps and production technologies: Git, GitHub, Jenkins, Docker, Kubernetes, and AWS

**Project development guidelines:**

* The project will be delivered within four sprints with every sprint delivering a minimal viable product.
* It is mandatory to perform proper sprint planning with user stories to develop all the components of the project.
* The learner can use any technology from the above-mentioned technologies for different layers of the project.
* The web application should be responsive and should fetch or send data dynamically without hardcoded values.
* The learner must maintain the version of the application over GitHub and every new change should be sent to the repository.
* The learner must implement a CI/CD pipeline using Jenkins.
* The learner should also deploy and host the application on an AWS EC2 instance.
* The learner should also implement automation testing before the application enters the CI/CD pipeline.
* The learner should use Git branching to do basic automation testing of the application in it separately.
* The learner should make a rich frontend of the application, which is user-friendly and easy for the user to navigate through the application.
* There will be two portals in the application, namely admin and user portal.

**Admin Portal:**

The admin portal deals with all the backend data generation and product information. The admin user should be able to:

* Add or remove medicine details from the application to build a rich product line
* Edit medicine details like name, price, seller, product description, and offers to keep the product information updated with the current prices
* Enable or disable a medicine product

**User Portal:**

It deals with the user activities. The end-user should be able to:

* Sign-in to the application to maintain a record of activities
* Search for products based on the search keyword
* Apply filters and sort results based on different cuisines to get the best deals
* Add all the selected food items to the cart and customize the purchase at the end
* Perform a seamless payment gateway
* Get an order summary details page once the payment is complete

**Git url:** <https://github.com/akv272161/Anil-Kumar-Verma>

**Steps to deploy Application at Ec2 instance using Jenkins pipelines:**

NOTE:

1. Please do remember to change the 'spring.datasource.url' property value in application-prod.properties file where your database is running.
2. Also do change the ip address of backend in the front-end application as well.

**1.0 Go to official Amazon Web Services site**

https://console.aws.amazon.com/ec2

**2.0 Create New Instance**

**3.0 Connect to the Instance**

**4.0 Open Command Prompt in your machine and navigate to the path where you have downloaded the pem file**

cd Downloads

**5.0 Connect to EC2 Instance by executing the '3rd and example' commands in the ec2 instance**

chmod 400 my-movie-plan.pem

ssh -i "my-movie-plan.pem" ec2-user@ec2-54-172-237-186.compute-1.amazonaws.com

**6.0 Update the Instance Once connected using the following command**

sudo yum update -y

**7.0 After updating the instance, install Java using the following command**

sudo yum install java-1.8.0-openjdk

**7.1 Check if Java is installed or not by executing the java version command**

sudo java -version

**8.0 Install Maven**

sudo yum install maven

**8.1 Check Maven version**

sudo mvn -v

**9.0 Install Git**

sudo yum install git

**9.1 Check Git Version**

sudo git --version

**10.0 Install Jenkins. By executing the following commands one by one. For more details visit this link:**[**https://pkg.jenkins.io/redhat-stable/**](https://pkg.jenkins.io/redhat-stable/)

sudo wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo

sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io.key

sudo yum install jenkins

**10.1 Start Jenkins after installing**

sudo systemctl start jenkins

**10.2 Check if Jenkins is running on port 8080 along with Public IPv4 addresses like:**

Example:

The IPv4 addresses of my instance is: 54.172.237.186

The Jenkins is running on 8080 port: 8080

Finally, use both to view jenkins: '54.172.237.186:8080'

**10.3 For the first time Jenkins will ask for password, to find the password, execute the following command in the EC2 Instance console**

sudo cat /var/lib/jenkins/secrets/initialAdminPassword

**10.4 Install the recommended plugins in the jenkins after logging in. After installing plugins, jenkins will prompt to create an admin user, go-head and create the user**

sudo cat /var/lib/jenkins/secrets/initialAdminPassword

**11.0 Open EC2 Instance console and Install Docker**

**11.1 Amazon Linux 2**

sudo amazon-linux-extras install docker

**11.2 Amazon Linux**

sudo yum install docker

**11.3 Start Docker**

sudo systemctl start docker

**11.4 Add the ec2-user to the docker group so you can execute Docker commands without using sudo.**

sudo usermod -a -G docker ec2-user

**11.5 The user jenkins needs to be added to the group docker. For more details, please refer:**[**https://docs.aws.amazon.com/AmazonECS/latest/developerguide/docker-basics.html**](https://docs.aws.amazon.com/AmazonECS/latest/developerguide/docker-basics.html)**,**[**https://gist.github.com/npearce/6f3c7826c7499587f00957fee62f8ee9**](https://gist.github.com/npearce/6f3c7826c7499587f00957fee62f8ee9)**,**[**https://portal.cloud303.io/forum/aws-1/question/i-want-to-install-docker-compose-on-an-amazon-linux-2-ec2-instance-9**](https://portal.cloud303.io/forum/aws-1/question/i-want-to-install-docker-compose-on-an-amazon-linux-2-ec2-instance-9)

sudo usermod -a -G docker jenkins

**11.6 Reboot the EC2 instance to pick up the new docker group permissions.**

sudo reboot

**12.0 After rebooting the EC2 Instance, execute the following commands.**

**12.1 Start Docker**

sudo systemctl start docker

**12.2 Verify that the ec2-user can run Docker commands without sudo.**

docker info

**12.3 Start Jenkins**

sudo systemctl start jenkins

**13.0 Add Maven to Jenkins Global tool Configuration**

sudo systemctl start jenkins

**14.0 Open Jenkins and create a pipeline job for MYSQL**

**15.0 Open Jenkins and create a pipeline job for Spring Boot**

**15.1 Add Maven to Jenkins**

**16.0 Open Jenkins and create a pipeline job for Angular**

**17.0 Connect all the three job and build them**

)

**18. Check if the app is running**

The IPv4 addresses of EC2 instance and the port on which the angular app is running: http://54.172.237.186:4040/