TechCiti Software Consulting Private Limited.

CIN: U72900KA2018PTC117376 **D-U-N-S No. :** 86 14 54180

No. 22 23 24 25/101, BNR Complex, J.P. Nagar, Bengaluru, Karnataka 560078.
Landline: 080 4162 8482 **Email**: info@techcitisoftware.in **Website:** www.techcitisoftware.in

---

Internship

Project Report

Details :

Name	**Deepak Sharma**
Email	**deepak.ynr305@gmail.com**
Position	**Software Developer Intern**
Domain	**Java**
Start Date	**1st June 2021**
End Date	**20th July 2021**
Project	**E-Commerce WebApp**
WebApp link	**http://ecommercewebapp-env.eba-kxq9mt2j.us-east-1.elasticbeanstalk.com/**
Source code	**https://github.com/deepak3005/E-Commerce-WebApp**

I was assigned a task to create an E-Commerce platform. And as we know the most common and user pleasing way is to build a web app for such purpose. So, to build a web app, I decided to use **Spring Tool Suite** (STS) which is an IDE to build a **Spring Boot application**.

Firstly, let's go through some important terms here :

- **Spring Framework** - The Spring Framework provides a comprehensive programming and configuration model for modern Java-based enterprise applications - on any kind of deployment platform. A key element of Spring is infrastructural support at the application level: Spring focuses on the "plumbing" of enterprise applications so that teams can focus on application-level business logic, without unnecessary ties to specific deployment environments.

- **Spring Boot** - Spring Boot makes it easy to create stand-alone, production-grade Spring based Applications that you can "just run". We take an opinionated view of the Spring platform and third-party libraries so you can get started with minimum fuss. Most Spring Boot applications need minimal Spring configuration.

  Features of Spring Boot :

  1. Create stand-alone Spring applications
  2. Embed Tomcat, Jetty or Undertow directly (no need to deploy WAR files)
  3. Provide opinionated 'starter' dependencies to simplify your build configuration
  4. Automatically configure Spring and 3rd party libraries whenever possible

5. Provide production-ready features such as metrics, health checks, and externalized configuration
6. Absolutely no code generation and no requirement for XML configuration

With the development of Spring Boot, the Spring framework has become substantially more user-friendly. There is no need to use the old framework unless you have a good reason for doing so.

Since Spring Boot is an integration framework, it makes sense to learn how to configure your libraries using it. Though the process is generally simple, these libraries often need some configuration.

Spring Boot has a huge user community which means you can find free learning materials and courses. Spring Boot is multi-threaded. This is useful when performing long or repetitive operations. When the main thread is consumed, others are used concurrently.

Spring Boot Application in Spring Tool Suite IDE :

When you start building a spring boot application, the first thing you have to do is add the required dependencies to your project and this process is known as dependency injection.

Dependency Injection is a fundamental aspect of the Spring framework, through which the Spring container "injects" objects into other objects or "**dependencies**". This allows for loose coupling of components and moves the responsibility of managing components onto the container.

So, for my project, I required the following dependencies :

1. **Spring Web** - The spring-web dependency contains common web specific utilities for both Servlet and Portlet environments. Starter of Spring web uses Spring MVC, REST and Tomcat as a default embedded server. The single spring-boot-starter-web dependency transitively pulls in all dependencies related to web development. It also reduces the build dependency count.

2. **Spring Data JPA** - When we implement a new application, we should focus on the business logic instead of technical complexity and boilerplate code. That's why the Java Persistent API (JPA) specification and Spring Data JPA are extremely popular. Spring Data JPA adds a layer on the top of JPA. It means, Spring Data JPA uses all features defined by JPA specification, especially the entity, association mappings, and JPA's query capabilities.

3. **Spring Security** - Spring Security is a powerful and highly customizable authentication and access-control framework. It is the de-facto standard for securing Spring-based applications. Spring Security is a framework that focuses on providing both authentication and authorization to Java applications. It provides comprehensive and extensible support for both Authentication and Authorization, protection against attacks like session fixation, clickjacking, cross site request forgery, etc.

4. **Thymeleaf** - Thymeleaf is a Java template engine for processing and creating HTML, XML, JavaScript, CSS, and text. The library is extremely extensible and its natural templating capability ensures templates can be prototyped without a back-end – which makes development very fast when compared with other popular template engines such as JSP.

5. **MySQL Driver** – Provides JDBC driver for connecting to MySQL database.

6. **Spring Boot DevTools** - DevTools stands for Developer Tool. The aim of the module is to try and improve the development time while working with the Spring Boot application. Spring Boot DevTools pick up the changes and restart the application.

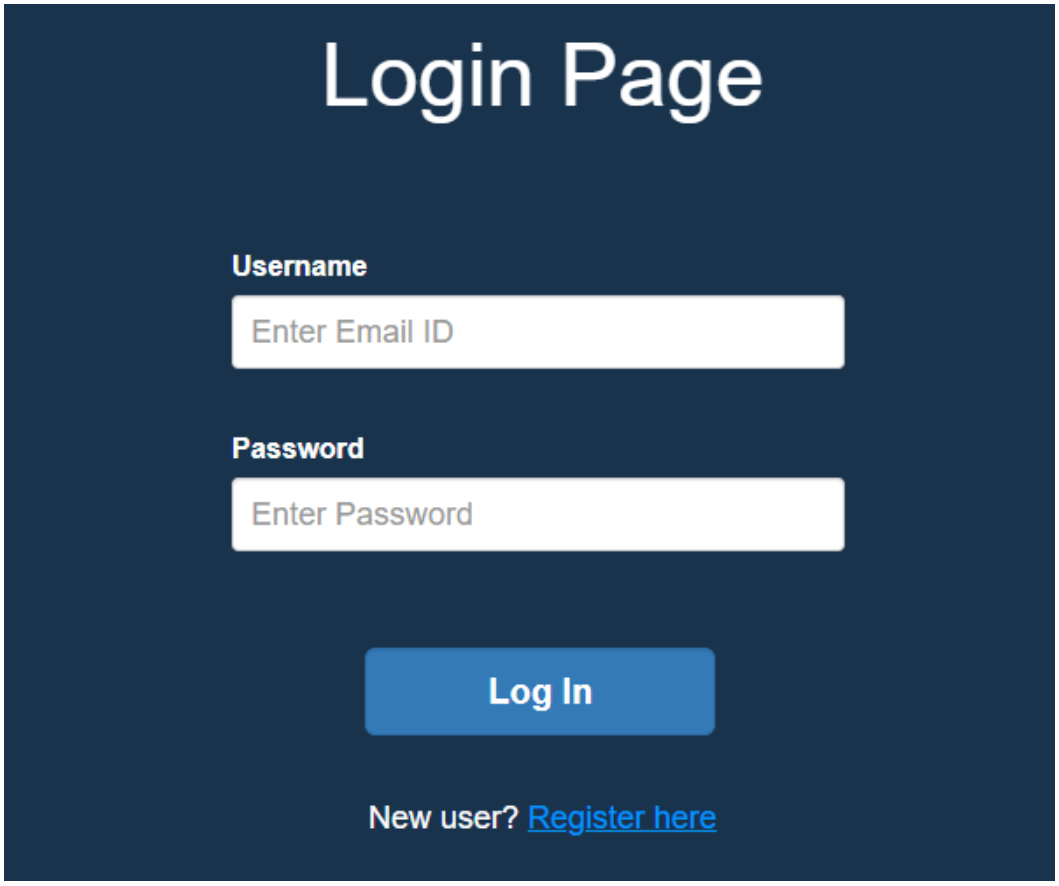
So, after doing the basic configuration of my project, I divided the whole project into different modules namely,

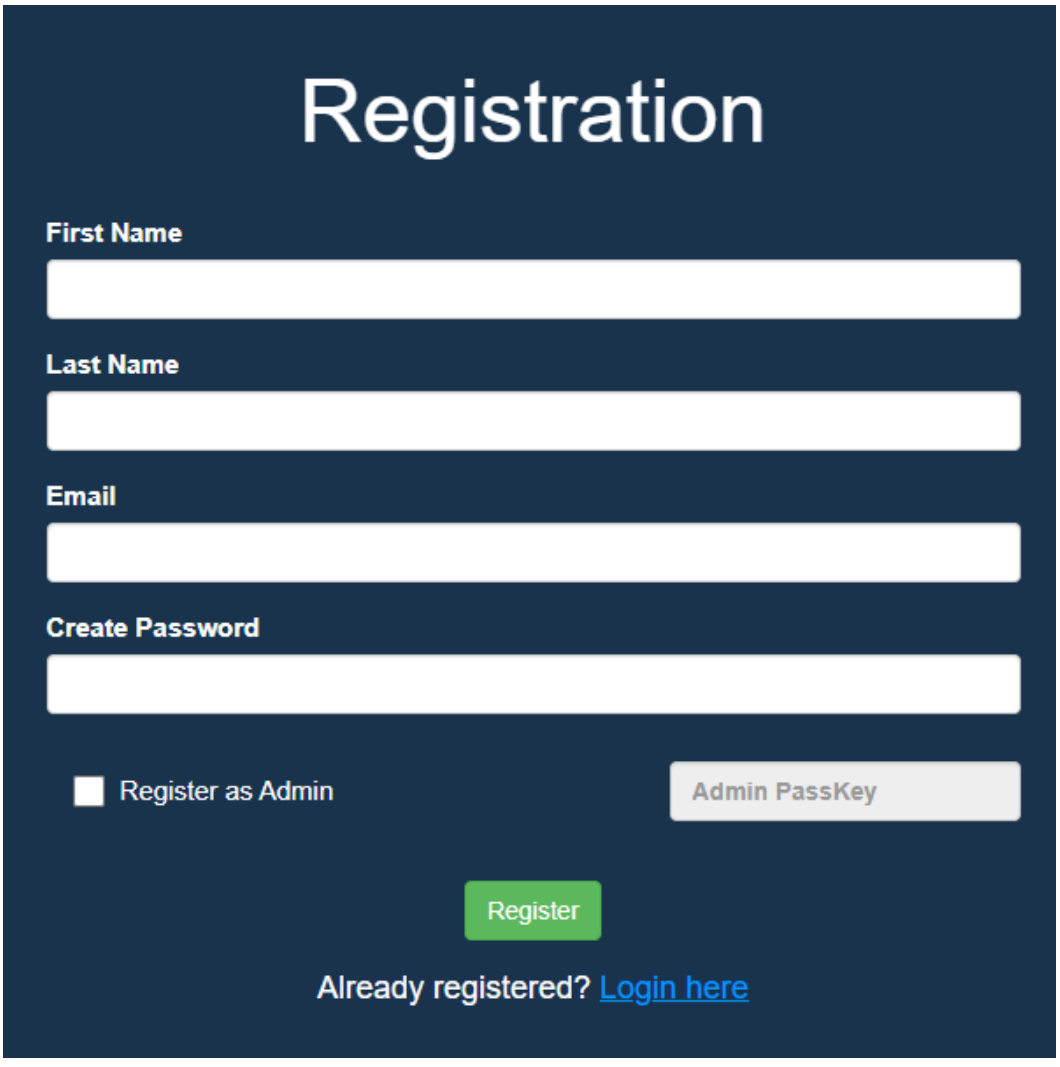
1. Login / Register (Authentication and Authorization)
2. Solutions
3. Products
4. Cart
5. Checkout
    - Address
    - Payment
6. Orders

Now, let’s go through each module separately.

1. Login / Register (Authentication and Authorization)

- Create user class with required variables.
- Create controller class for the url mappings regarding login and registration.
- Connect to JPA repository, make functions in Services to validate the login credentials.
- Use BCryptPasswordEncoder to encode the user password and then save it to the database.
- Use custom login success handler to override the default login template provided by spring security.
- Make Login and Registrations templates using thymeleaf.
- Connect them to mysql database using the injected driver dependency.
- Once this Authentication process is successfully coded, further proceed to Authorization part.
- Need to authorize based on : ADMIN and CUSTOMER.
- Map users to specific roles.
- Add the concept of AdminPassKey, so that only the users that have this passkey are able to register as Admin to this portal.
- In the Security Configuration class, match the URLs to appropriate ROLES i.e. either ADMIN or USER.
- Keeping in mind the CRUD operations, allow specific operations to be accessed by specific roles.
- Now here your Authorization part ends.
- At last, provide a Logout option.


Login Page
Username
Enter Email ID
Password
Enter Password
Log In
New user? Register here


Registration
First Name
Last Name
Email
Create Password
Register as Admin
Admin PassKey
Register
Already registered? Login here

2. Solutions

- Aim is to provide IT Solutions to the outside world.
- IT Solutions like server, storage, computation, networking etc.
- Creating a Solutions class with required fields like id, name, description etc.
- Creating table in database using spring annotations. These variables go as columns to the table.
- Create controller, repository, and services using the appropriate annotations.
- Controller maps the URLs to appropriate functions.
- These functions are basically used to View all solutions, Add a solution, Update a particular solution and to Delete a solution.
- Implement these functions in the Services class using the Repository predefined functions.
- Once the main class, controller, repository and service are made, now it's time to create templates using thymeleaf.
- In this template, you need to create appropriate and user friendly views to access all functions.
- Map the forms, buttons with specific URLs that would basically redirect to the Controller and provide the required result that could either be a new template or another redirected page.

Solutions

+ Add Solution

Type	Description	Actions
Server	First Solution	View Update Delete
Storage	Second Solution	View Update Delete
Networking	Third Solution	View Update Delete

Add Solution

Solution Type

Solution Description

Save Solution

Update Solution

Server

First Solution

Update Solution

3. Products

- Create a Products class with required fields like id, name, type, specification, original price, net price etc.
- It's important to note here that products are basically under specific solutions, so here we need to provide mapping of products to solutions using the product_Id and solution_Id.
- Creating table in database using spring annotations. These variables go as columns to the table.
- Create controller, repository, and services using the appropriate annotations.
- Controller maps the URLs to appropriate functions.
- These functions are basically used to View all products, View products under a specific solution, View a single product, Add a product, Update a particular product and to Delete a product.
- Implement these functions in the Services class using the Repository predefined functions.
- Once the main class, controller, repository and service are made, now it's time to create templates using thymeleaf.
- In this template, you need to create appropriate and user friendly views to access all functions.
- Map the forms, buttons with specific URLs that would basically redirect to the Controller and provide the required result that could either be a new template or another redirected page.

My Orders Cart Logout

Solutions

Category

- All
- Server
- Storage
- Networking

Products

ASUS Rack Server RS520 E8 RS8-level3
Dual Processor E5-2678 v3

Original Price : Rs. 2.0

Net Price : Rs. 2.0

View Product

Add Product

Product Name

Product Type

Product Specs

0.0

0.0

Product Image : Choose file No file chosen

Save Product

Update Product

ASUS Rack Server RS520 E8 RS8-level3 Dual Proc

Rack Server

Intel Xeon Bronze 3204 (2nd Gen.,1.92GHz,6Core

2.0

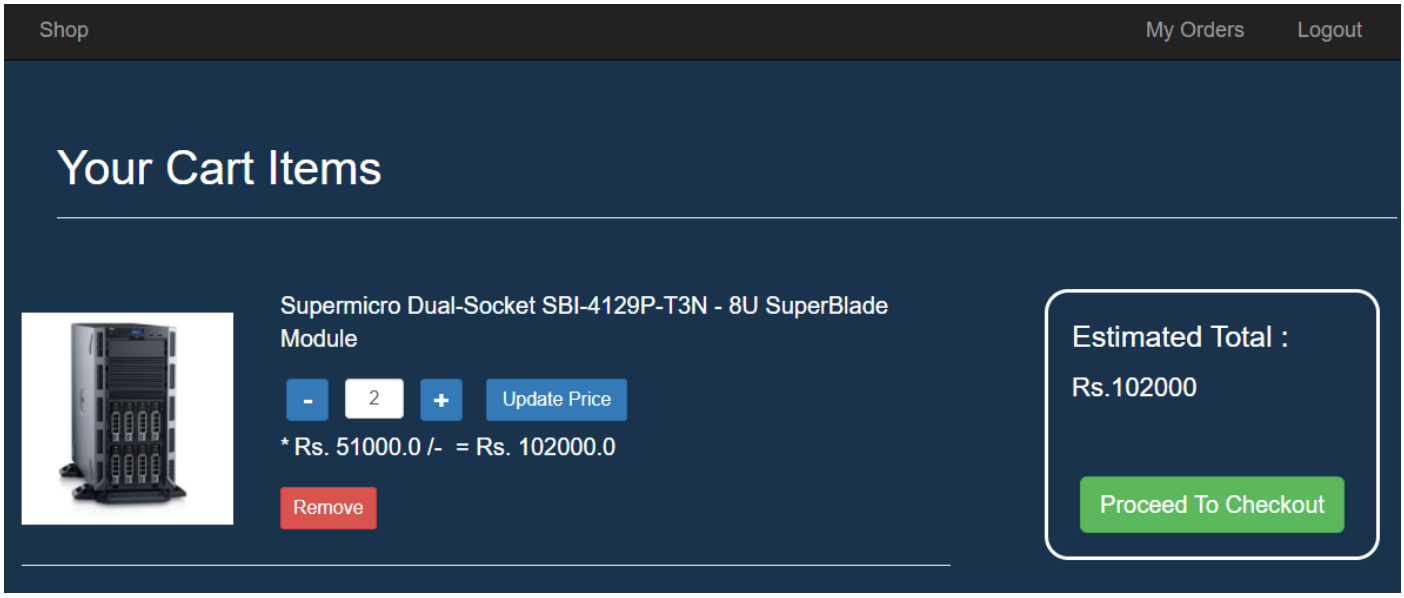
2.0

Product Image : Choose file No file chosen

Update Product

4. Cart

- Create a Cart class with required fields like cart_id, product_id, quantity etc.
- It's important to note here that each cart belongs to a particular user, so here we need to provide mapping of cart to user by cart_id and user_id.
- Creating table in database using spring annotations. These variables go as columns to the table.
- Create controller, repository, and services using the appropriate annotations.
- Controller maps the URLs to appropriate functions.
- These functions are basically used to View cart, Add product to cart, Update quantity and price, Delete a product from the cart.
- Implement these functions in the Services class using the Repository predefined functions.
- Once the main class, controller, repository and service are made, now it's time to create templates using thymeleaf.
- In this template, you need to create appropriate and user friendly views to access all functions.
- Map the forms, buttons with specific URLs that would basically redirect to the Controller and provide the required result that could either be a new template or another redirected page.

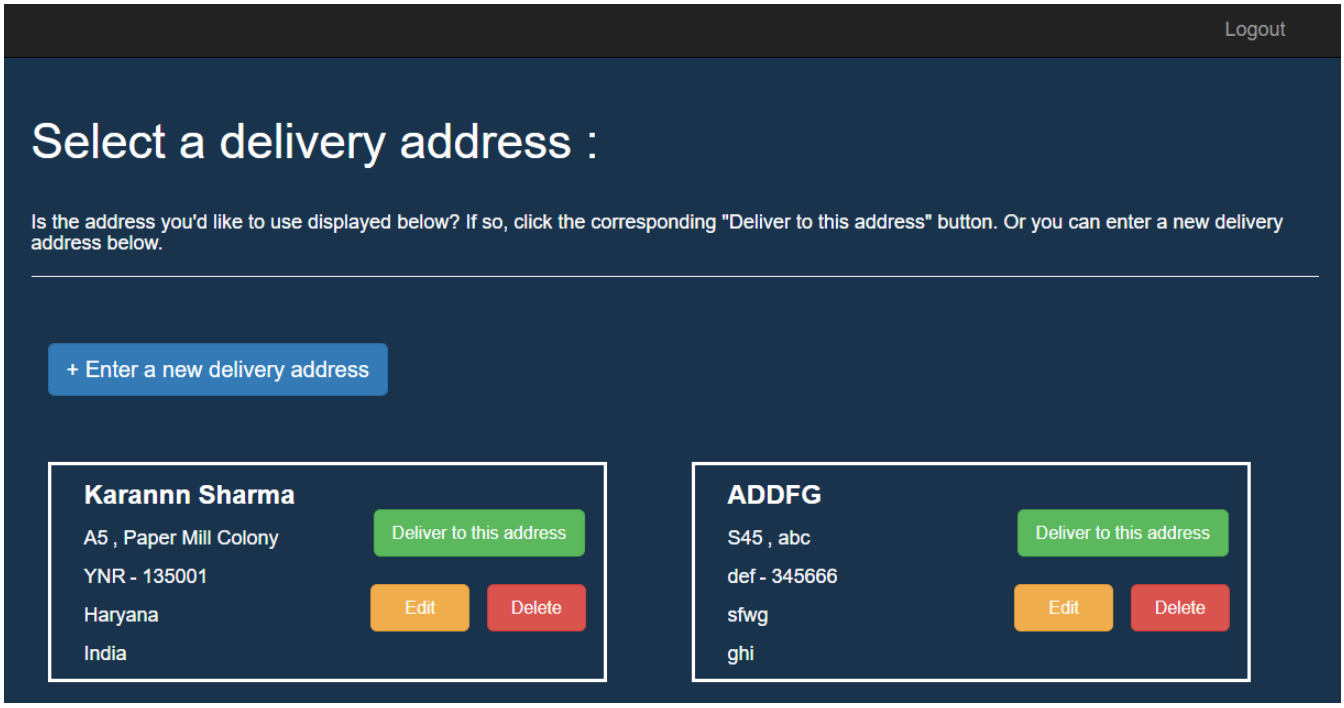


5. Checkout

(a) Address :

- Create an Address class with required fields like address_id, name, mobile, house number, area, city, pincode, state, country etc.
- It's important to note here that the addresses saved belong to a particular user, so here we need to provide mapping of address to user by address_id and user_id.
- Creating table in database using spring annotations. These variables go as columns to the table.
- Create controller, repository, and services using the appropriate annotations.
- Controller maps the URLs to appropriate functions.

- These functions are basically used to View addresses, Add new address, Update an address, Delete an address.
- Implement these functions in the Services class using the Repository predefined functions.
- Once the main class, controller, repository and service are made, now it's time to create templates using thymeleaf.
- In this template, you need to create appropriate and user friendly views to access all functions.
- Map the forms, buttons with specific URLs that would basically redirect to the Controller and provide the required result that could either be a new template or another redirected page.



(b) Payment

- I decided to integrate Paytm gateway for my payment process.
- So for this, firstly a Paytm checksum dependency needs to be injected to this project.
- Next, create a payment class with required fields like merchant_id, merchant_key, Paytm redirect url, order_id, customer_id, total amount etc.
- Here, there's no need for repository and service classes.
- Create a payment controller that maps the URLs to appropriate functions.
- We need to pass the user_id, address, cart_items, address_id, total_amount to this payment controller in order to get the required fields to redirect to the Paytm gateway.
- Now, pay using the payment page provided by the Paytm gateway and once the payment is successful, you'll be given an auto-generated receipt.
- Now it's time to create templates using thymeleaf to show that order has been placed successfully.
- On this template, provide appropriate links to go either go back to shop and/or review your order.

DEEPAK SHARMA

GO BACK

DEEPAK SHARMA Order

Transaction ID: 1a2b3c

Amount to be paid

₹1,45,200

PAY INSTANTLY USING QR CODE

Scan QR code using your Paytm app

paytm

Click to enlarge

SELECT AN OPTION TO PAY

paytm

Pay easily using your saved payment methods

Mobile number registered with Paytm

+91 9306123103

Proceed

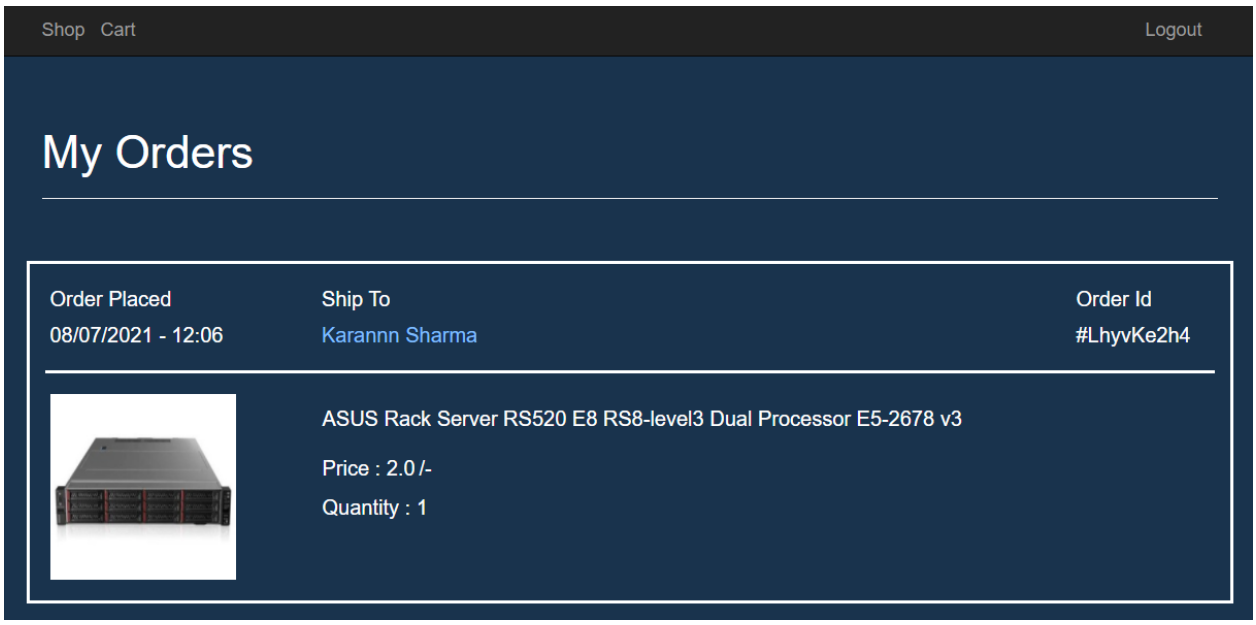
Remember me

Debit Card

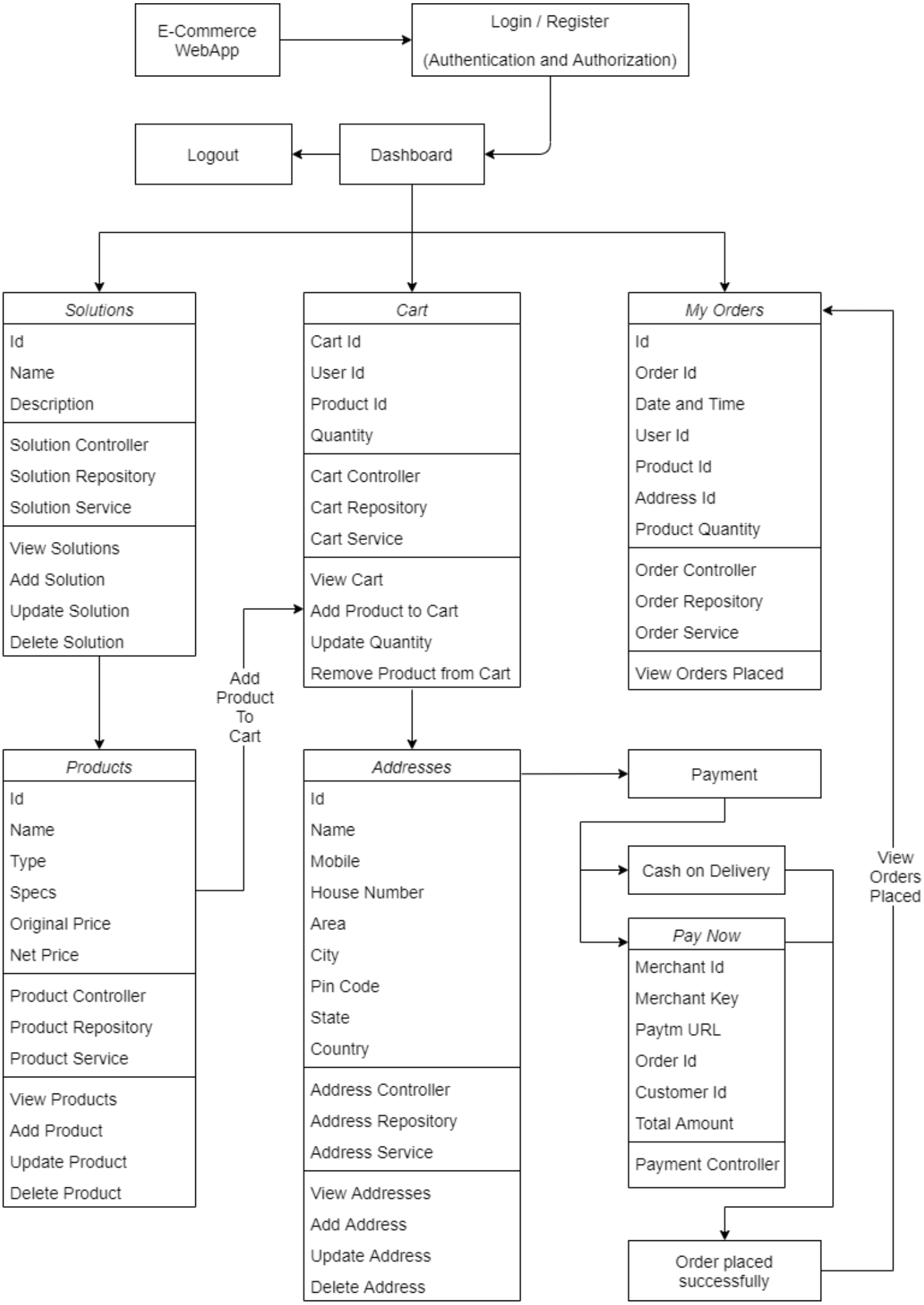
Credit Card

6. Orders

- Create an Order class with required fields like order_id, cart and product fields.
- It's important to note here that each order belongs to a particular user, so here we need to provide mapping of order to user by order_id and user_id.
- Creating table in database using spring annotations. These variables go as columns to the table.
- Create controller, repository, and services using the appropriate annotations.
- Controller maps the URLs to appropriate functions.
- Here you can only view the orders and no other function is to be provided.
- Implement this function in the Services class using the Repository predefined functions.
- Once the main class, controller, repository and service are made, now it's time to create templates using thymeleaf to view the orders placed.



Class Diagram :



Additional Module

Deploying web app on AWS using RDS and Elastic Beanstalk

After completing the whole project, I further planned on hosting this web app on AWS. After going through a variety of AWS services and other content available on the internet, I came across two essential services which turned out to be most appropriate for this scenario : RDS and Elastic Beanstalk.

Elastic Beanstalk provides integration with Amazon Relational Database Service (Amazon RDS) to help you add a database instance to your Elastic Beanstalk environment. You can use Elastic Beanstalk to add a MySQL, PostgreSQL, Oracle, or SQL Server database to your environment during or after environment creation. When you add a database instance to your environment, Elastic Beanstalk provides connection information to your application by setting environment properties for the database hostname, port, user name, password, and database name.

Firstly, you have to create a mysql database in RDS and connect it to your database on your local machine. Also, you need to make the required changes in your code where the local database was configured earlier. Once the database connection is successful and it's up and running, now you need to build a .jar file of your project and then deploy it on Elastic Beanstalk. On successful deployment, you'll see a health monitor pointing to OK, indicating that your app is successfully deployed.

Following this procedure, I was able to host my web app on AWS and now it's live and you can access it through the following link :

http://techcitiecommercewebapp-env.eba-xunkneem.us-east-1.elasticbeanstalk.com/

References

- https://spring.io/guides/gs/spring-boot/
- https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-dependencies
- https://www.youtube.com/watch?v=L9oWG6aj_U8&list=PLIVs7Xl0M_DxWZXdjjgPwBSa_Gj--K-xp
- https://www.youtube.com/watch?v=wW9yEPcgZT0&list=PLIVs7Xl0M_DxWZXdjjgPwBSa_Gj--K-xp&index=4
- https://www.pixeltrice.com/image-gallery-spring-boot-application-using-mysql-and-thymeleaf/
- https://www.pixeltrice.com/payment-integration-with-paytm-in-spring-boot-application/
- https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/using-features.managing.db.html
- https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/AWSHowTo.RDS.html
- https://www.youtube.com/watch?v=JAyjGiQHWf8&list=WL&index=4
- https://www.youtube.com/watch?v=bi7ow5NGC-U&list=WL&index=7&t=190s