1. DAO - Database accessing layer.
Spring web application flow:
MVC Architecture:
* UI(View) --> (passing Http request)--> Controller(handles request) --> Service(In service, login service) <--> (DAO) <--> DataBase
* UI - login form page
* controller handles request - An object is created based on the request params.
2. Requirements: For running the Application , JRE is necessary. For development, the Technologies used are: JDK 1.8 Eclipse (IDE).
Technical Specifications: Java, hibernate, Spring MVC, Spring Boot, mysql, html, css, maven, junit, Mockito.

Task list:we planned 2 sprints and achieved the following schedules-
Sprint 1:
1. Project Setup.
2. Writing Entities(Pojos) .
3. Writing Service Interfaces.
4. Desgined Database scheme and tables.
5. Implementing Services(Actual Task).
Sprint 2:
6. Writing Controller requests.
7. Junit Mockito Test cases.
8. Integration testing and Documentation.

--> Types pf Users: Admin & Customer.
1. User activities:
1.1 register
1.2 login
1.3 search items
1.4 select items
1.5 access the cart
1.6 payment

2. Admin Operations:
2.1 Admin can change password whenever he/she wants to.
Implemented in UserService:
1. changePassword
2.2 Managing the items in the store, including categorizing them.
Implemented in ProductService:
1. saveProduct
2. addProductStock
3. updateProductStock
4. getProducts.
2.3 Browse the list of users who have signed up and be able to search users.
Implemented in UserService:
1. getUsers:
Search for, and get list of users with user Id,
Search for, and get list of users with name,
Browse list of all Active Users.
2.4 See purchase reports filtered by date and category.
Implemented in ProductService:
1. getProducts.

Services:
1. UserService:
1.1 registerUser(User user):

* User can register by providing name, email Id, Password & mobile number.

1.2 loginUser(User user):

* User can login with email Id and password.

1.3 getUser(User user):

* Browse all Active users or Search for specific user by user Id/ user name.

2. ProductService:

2.1 saveProduct(Product product):

This service allows admin to add Insert new product.

2.2 addProductStock(ProductStock productStock):

This service allows admin to add number of products to the stock.

2.3 updateProductStock(Long skuId, Integer updateQuantity):

This service allows admin to update product stock.

2.4 getProducts(Product product, Integer pageNo, Integer pageLimit):

This service allows User to browse vaiours products

3. CartService:

3.1 saveCart(Cart cart):

This service adds products to the cart.

3.2 getCart(Long userId):

This service provides list of items in the cart.

3.3 deleteCart(Long cartId):

This service deletes products from the cart.

4. OrderService:

4.1 saveOrder(Order order):

This service creates an order with the products from the cart.

4.2 getOrder(OrderSearchCriteria orderSearchCriteria):

This service provides list of products orders placed.

Description & Java concepts:

1. The application allows user to register and browse various categorized products.
2. Selected products will be added into cart.
3. User can place orders from the cart.
4. User can remove orders from cart
4. Based on orders placed, product stock will be updated.
5. Used Spring MVC, Spring boot, hibernate.
5. Achieved Unit testing through junit test cases & conducted mockito tests.

Steps to run the Aplication in tomcat server:

1. Right click on the project in eclipse.
2. Select Maven build:
   it performs Maven clean and then Maven install.
3. Copy the sportyshoes.var file from the target folder after Maven build.
4. Paste the var file in the webapps folder in tomcat files.
5. Up the tomcat server in webapp folder in tomcat
6. Application runs on the default tomcat port - localhost:8080.
7. You can test it using Postman.

Developer details: M. Koutilya Theertha