**KitchenStory.Com – Specification Document**

This document contains sections for:

* [Sprint planning and Task completion](file:///C:\Users\user\Desktop\Santhi%20java%20projects\lockerspvt%20ltd%20project\LockersPvtLtd%20project\LockersPvtLtd_Project%20Specification%20document.docx#Sprint_plan)
* [Core concepts used in project](file:///C:\Users\user\Desktop\Santhi%20java%20projects\lockerspvt%20ltd%20project\LockersPvtLtd%20project\LockersPvtLtd_Project%20Specification%20document.docx#Core_concepts)
* [Demonstrating the project code and its output.](file:///C:\Users\user\Desktop\Santhi%20java%20projects\lockerspvt%20ltd%20project\LockersPvtLtd%20project\LockersPvtLtd_Project%20Specification%20document.docx#Product_capability)
* [Unique Selling Points of the Application](file:///C:\Users\user\Desktop\Santhi%20java%20projects\lockerspvt%20ltd%20project\LockersPvtLtd%20project\LockersPvtLtd_Project%20Specification%20document.docx#USP)

The code for this project is located at https://github.com/\*\*\*\*\*\*\*\*\*\*

The project is developed by Santhi mogili.

## **Sprints planning and Task completion**

The project is planned to be complete the project in 2 sprints.

Tasks to be completed in the sprint are:

* Creating the flow of the application.
* Writing the Java program to fulfill the requirements of the project.
* Testing the Java program with different kinds of User input.
* Pushing code to GitHub.
* Creating this specification document to show project code and output of project.

## **Core concepts used in project**

Springboot, devtools, logger, maven, apache tomcat server, localhost server port, classes, interfaces, webapp, jsp pages, postman, swagger UI, browser for viewing jsp pages,exception handling etc.,

## **Demonstrating the project code and output of the code.**

To demonstrate the project code, below are the sub-sections configured to highlight output of the project:

1. **Creating the project, package and class in STS**
2. **Writing a program in Java for SportShoes.com .**
3. [**Pushing the code to GitHub repository**](file:///C:\Users\user\Desktop\Santhi%20java%20projects\lockerspvt%20ltd%20project\LockersPvtLtd%20project\LockersPvtLtd_Project%20Specification%20document.docx#Step_6)

## **Step 1: Creating a new spring starter project, package classes and jsps in STS**

* Open STS
* Go to File -> New -> Spring starter project -> Name
* Type in Name-> **Kitchenstory,** in type->**Maven, in Packaging->War,** in Java **version->8,** in language**->Java,** in package**->com.simolilearn.kitchenstory** and click on “Finish.”
* Select your package and go to File -> New ->Class
* Enter required classes names and interfaces names using new Class and new Interface and click on “Finish.”
* From src->main->webapp->right click->New->Other->jsp file->type appropriate jsp name and click on “Finish.”

**Step 2: Writing a program in Java for Lockers Pvt Ltd app .**

**2.1: Admin Class**

**package** com.simplilearn.kitchenstory;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** lombok.Data;

@Data

@Entity

**public** **class** Admin {

@Id

@GeneratedValue(strategy=GenerationType.***AUTO***)

**private** **int** id;

**private** String ad\_email;

**private** String ad\_password;

}

**2.2: Admin Repo Class**

package com.simplilearn.kitchenstory;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.data.jpa.repository.Query;

public interface AdminRepo extends JpaRepository<Admin, Integer>{

@Query("select admin from Admin admin where admin.ad\_email=?1 and admin.ad\_password=?2")

public Admin findByadminpwd(String ad\_email,String ad\_password) ;

}

**2.3: Admin DAO Class**

package com.simplilearn.kitchenstory;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.data.jpa.repository.Query;

public interface AdminRepo extends JpaRepository<Admin, Integer>{

@Query("select admin from Admin admin where admin.ad\_email=?1 and admin.ad\_password=?2")

public Admin findByadminpwd(String ad\_email,String ad\_password) ;

}

**2.4:Product Class**

package com.simplilearn.kitchenstory;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.NoArgsConstructor;

@Data

@Entity

@NoArgsConstructor

@AllArgsConstructor

public class Product {

@Id

@GeneratedValue(strategy=GenerationType.AUTO)

private int id;

private String name;

private String category;

private String description;

private float price;

}

**2.5: Product Repo Class**

**package** com.simplilearn.kitchenstory;

**import** org.springframework.data.jpa.repository.JpaRepository;

**public** **interface** ProductRepo **extends** JpaRepository<Product, Integer> {

}

**2.6: Product DAO Class**

package com.simplilearn.kitchenstory;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

@Service

public class PoductDao {

@Autowired

ProductRepo repo;

public Product insert(Product p) {

return repo.save(p);

}

public List<Product> getall(){ //Method for select \* from Shoes

return repo.findAll();

}

public void deletebyId(Integer id) {

repo.deleteById(id);

}

public Product getShoebyId(Integer id) {

return repo.getReferenceById(id);

}

public Product editbyId(Product p)

{

Product existingProduct=repo.findById(p.getId()).orElse(null);

existingProduct.setName(p.getName());

existingProduct.setCategory(p.getCategory());

existingProduct.setDescription(p.getDescription());

existingProduct.setPrice(p.getPrice());

repo.save(existingProduct);

return existingProduct;

}

}

**2.6: Product Controller Class**

package com.simplilearn.kitchenstory;

import java.util.List;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.ResponseBody;

import org.springframework.web.client.RestTemplate;

import org.springframework.web.servlet.ModelAndView;

@Controller

public class ProductController {

@Autowired

AdminDao ad\_dao;

@Autowired

AdminRepo arepo;

@Autowired

UserDao u\_dao;

@Autowired

UserRepo urepo;

@Autowired

ProductRepo prepo;

@Autowired

PoductDao Pdao;

float Total=20000.0f;

@ResponseBody

@RequestMapping("/")

public ModelAndView displaypage(HttpServletRequest request,HttpServletResponse response) {

ModelAndView mv=new ModelAndView();

mv.setViewName("index.jsp");

return mv;

}

@RequestMapping("/loginadmin")

public ModelAndView adminlogin(HttpServletRequest req,HttpServletResponse res) {

ModelAndView mv=new ModelAndView();

String ad\_email = req.getParameter("ad\_email");

String ad\_password=req.getParameter("ad\_password");

Admin admin=arepo.findByadminpwd(ad\_email, ad\_password);

if(admin!=null) {

mv.setViewName("AdminMainIndex.jsp");

}

else {

mv.setViewName("fail.jsp");

}

return mv;

}

@RequestMapping("/loginuser")

public ModelAndView userlogin(HttpServletRequest req,HttpServletResponse res) {

ModelAndView mv=new ModelAndView();

String username = req.getParameter("username");

String password=req.getParameter("password");

User u=urepo.findByuserpwd(username, password);

if(u!=null) {

List<Product> list=Pdao.getall();

mv.setViewName("UserMain.jsp");

mv.addObject("list", list);

}

else {

mv.setViewName("fail.jsp");

}

return mv;

}

@ResponseBody

@RequestMapping("/registeruser")

public ModelAndView registermslogin(HttpServletRequest req,HttpServletResponse res) {

ModelAndView mv=new ModelAndView();

User u=new User();

u.setUsername(req.getParameter("username"));

u.setEmail(req.getParameter("useremail"));

u.setPassword(req.getParameter("userpwd"));

User uu=u\_dao.insert(u);

if(uu!=null) {

mv.setViewName("User\_succ\_reg.jsp");

}

return mv;

}

@ResponseBody

@RequestMapping("/registeradmin")

public ModelAndView registeradmin(HttpServletRequest req,HttpServletResponse res) {

ModelAndView mv=new ModelAndView();

Admin a=new Admin();

a.setAd\_email(req.getParameter("ad\_email"));

a.setAd\_password(req.getParameter("ad\_pwd"));

Admin ad=ad\_dao.insert(a);

if(ad!=null) {

mv.setViewName("Admin\_succ\_reg.jsp");

}

return mv;

}

@RequestMapping("/insertproduct")

public ModelAndView insertControl(HttpServletRequest request,HttpServletResponse response) {

ModelAndView mv=new ModelAndView();

Product s=new Product();

s.setId(Integer.parseInt(request.getParameter("pid")));

s.setName(request.getParameter("pname"));

s.setCategory(request.getParameter("pcategory"));

s.setDescription(request.getParameter("pdescription"));

s.setPrice(Float.parseFloat(request.getParameter("pprice")));

Product ep=Pdao.insert(s);

List<Product> list=Pdao.getall();

if(ep!=null) {

System.out.println("insertion is successful!\n");

mv.setViewName("display.jsp");

mv.addObject("list",list);

}

return mv;

}

@RequestMapping("/deleteproduct")

public ModelAndView deleteControl(HttpServletRequest request,HttpServletResponse response,@RequestParam("deleteid") Integer id) {

ModelAndView mv=new ModelAndView();

Product p=new Product();

Pdao.deletebyId(id);

List<Product> list=Pdao.getall();

if(list!=null) {

System.out.println("deleteion is successful!\n");

mv.setViewName("display.jsp");

mv.addObject("list",list);

}

else {

System.out.println("Check the Product id..Deletion unsuccessful!\n");

}

return mv;

}

@RequestMapping("/getallproducts")

public ModelAndView displayAll(HttpServletRequest request,HttpServletResponse response) {

ModelAndView mv=new ModelAndView();

List<Product> list=Pdao.getall();

mv.setViewName("display.jsp");

mv.addObject("list",list);

return mv;

}

@RequestMapping("/updateproduct")

public ModelAndView updateAll(HttpServletRequest request,HttpServletResponse response) {

ModelAndView mv=new ModelAndView();

List<Product> list=Pdao.getall();

mv.setViewName("display.jsp");

mv.addObject("list",list);

return mv;

}

@RequestMapping("/editproduct")

public ModelAndView edit(HttpServletRequest request,HttpServletResponse response) {

ModelAndView mv=new ModelAndView();

Product s=new Product();

s.setId(Integer.parseInt(request.getParameter("pid")));

s.setName(request.getParameter("pname"));

s.setCategory(request.getParameter("pcategory"));

s.setDescription(request.getParameter("pdescription"));

s.setPrice(Float.parseFloat(request.getParameter("pprice")));

Product p=new Product();

if(p!=null) {

System.out.println("Edit is successful!\n");

mv.setViewName("prodcut\_updated.jsp");

}

return mv;

}

@RequestMapping("/buynow")

public ModelAndView buynow(HttpServletRequest request,HttpServletResponse response,@RequestParam("pid") Integer id, @RequestParam("price") Float price) {

ModelAndView mv=new ModelAndView();

float remain;

remain=Total-price;

if(remain>0)

{

System.out.println("Product Bought!\n");

Pdao.deletebyId(id);

List<Product> list=Pdao.getall();

mv.setViewName("BuySuccess.jsp");

mv.addObject("list",list);

}

else {

System.out.println("Low Balance...");

List<Product> list=Pdao.getall();

mv.setViewName("UserMain.jsp");

mv.addObject("list",list);

}

return mv;

}

@RequestMapping("/Payment")

public ModelAndView Payment(HttpServletRequest request,HttpServletResponse response,@RequestParam("pid") Integer id,@RequestParam("price") Float price) {

ModelAndView mv=new ModelAndView();

Product s=new Product();

float Total=20000.0f;

float remain;

remain=Total-price;

if(remain>price)

{

System.out.println("Product Bought!\n");

Pdao.deletebyId(id);

List<Product> list=Pdao.getall();

mv.setViewName("UserMain.jsp");

mv.addObject("list",list);

}

return mv;

}

}

**2.7: User Class**

**package** com.simplilearn.kitchenstory;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** lombok.Data;

@Data

@Entity

**public** **class** User {

@Id

@GeneratedValue(strategy=GenerationType.***AUTO***)

**private** **int** id;

**private** String username;

**private** String password;

**private** String email;

}

**2.8: User DAO Class**

**package** com.simplilearn.kitchenstory;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Service;

@Service

**public** **class** UserDao {

@Autowired

UserRepo u\_repo;

**public** User insert(User u) {

**return** u\_repo.save(u);

}

}

**2.9: User Repo Class**

package com.simplilearn.kitchenstory;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.data.jpa.repository.Query;

public interface UserRepo extends JpaRepository<User, Integer>{

@Query("select user from User user where user.username=?1 and user.password=?2")

public User findByuserpwd(String username,String password) ;

}

**2.10: Bank Class**

package com.ecommerce.example;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import lombok.Data;

@Data

@Entity

public class Bank {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int bank\_Id;

private String Bank\_balance;

}

**2.11: Bank DAO Class**

package com.ecommerce.example;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

@Service

public class BankDAO {

@Autowired

BankRepo b\_repo;

public Bank b\_insert(Bank b)

{

return b\_repo.save(b);

}

//insert list of bank id's for restcontroller

public List<Bank> insertall\_ids(List<Bank> b) {

return b\_repo.saveAll(b);

}

}

**2.12: Bank Repo Class**

package com.ecommerce.example;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.data.jpa.repository.Query;

public interface BankRepo extends JpaRepository<Bank, Integer> {

@Query("select b from Bank b where b.bank\_Id=?1 and b.Bank\_balance=2")

public List<Bank> getByIdAndBalance(int bank\_Id,String Bank\_balance);

}

**2.13: Errormsg Class**

package com.ecommerce.example;

import java.util.Date;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.NoArgsConstructor;

@Data

@AllArgsConstructor

@NoArgsConstructor

public class ErrorMsg {

private Date timestamp;

private String errormsg;

private String path;

}

**2.14: GlobalExceptionHandler Class For ExceptionHandling**

package com.ecommerce.example;

import java.util.Date;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.ControllerAdvice;

import org.springframework.web.bind.annotation.ExceptionHandler;

import org.springframework.web.context.request.WebRequest;

@ControllerAdvice

public class GlobalExceptionHandler {

//general exception handler

@ExceptionHandler(Exception.class)

public ResponseEntity<?> handleNotFoundException(Exception exception,WebRequest request){

ErrorMsg details=new ErrorMsg(new Date(),exception.getMessage(),request.getDescription(false) );

return new ResponseEntity<>(details,HttpStatus.INTERNAL\_SERVER\_ERROR);

}

}

**2.15: SecurityConfig Class for Spring Security**

package com.ecommerce.example;

import org.springframework.context.annotation.Bean;

import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

import org.springframework.security.crypto.password.NoOpPasswordEncoder;

import org.springframework.security.crypto.password.PasswordEncoder;

@EnableWebSecurity

public class SecurityConfig extends WebSecurityConfigurerAdapter {

@Override

protected void configure(AuthenticationManagerBuilder auth) throws Exception {

auth.inMemoryAuthentication()

.withUser("admin")

.password("admin")

.roles("ADMIN");

}

//no password encoder used

@Bean

public PasswordEncoder getpassword() {

return NoOpPasswordEncoder.getInstance();

}

**2.18: jsp pages for view on browser localhost :8085 and created tables in sql database kitchenstory**



## **Pushing the code to GitHub repository**

* Open your command prompt and navigate to the folder where you have created your files.

**cd <** **D:\Kitchenstory.com\File>**

* Initialize repository using the following command:

**git init**

* Remote add the repository using the following command

**https://github.com/Mogilisanthi/Simplilearn\_Phase-4.git**

* Add all the files to your git repository using the following command:

**git add .**

* Commit the changes using the following command:

**git commit . -m “Kitchenstory project”**

* Push the files to the folder you initially created using the following command:

**git push -u origin master**