SportyshoesApplication.java

```
package com.MyFirstSpringBootProj.sportyshoes;

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

@SpringBootApplication
public class SportyshoesApplication {

	public static void main(String[] args) {
		SpringApplication.run(SportyshoesApplication.class, args);
	}

}
```

SwaggerConfig.java

```
package com.MyFirstSpringBootProj.sportyshoes.config;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import springfox.documentation.builders.ApiInfoBuilder;
import springfox.documentation.builders.PathSelectors;
import springfox.documentation.builders.RequestHandlerSelectors;
import springfox.documentation.service.ApiInfo;
import springfox.documentation.service.Contact;
import springfox.documentation.spi.DocumentationType;
import springfox.documentation.spring.web.plugins.Docket;
import springfox.documentation.swagger2.annotations.EnableSwagger2;

@Configuration
@EnableSwagger2
public class SwaggerConfig {
    @Bean
    public Docket api() {
        return new Docket(DocumentationType.SWAGGER_2).select()
            .apis(RequestHandlerSelectors
                .basePackage("com.MyFirstSpringBootProj.sportyshoes"))
            .paths(PathSelectors.regex("/.*"))
            .build().apiInfo(apiEndPointsInfo());
    }

    private ApiInfo apiEndPointsInfo() {

        return new ApiInfoBuilder().title("Sporty-Shoes.Com")
            .description("Sporty Shoes Online Store")
            .contact(new Contact("Kiran Kumar c", "Sporty Shoes.com", "kiranc2210@gmail.com"))
            .license("Apache 2.0")
            .licenseUrl("http://www.apache.org/licenses/LICENSE-2.0.html")
            .version("1.0.0")
            .build();
    }
}
MODEL CLASS
Admin.java
package com.MyFirstSpringBootProj.sportyshoes.models;
```

```
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import lombok.AllArgsConstructor;
import lombok.Data;

import lombok.NoArgsConstructor;

@Data
@AllArgsConstructor
@NoArgsConstructor
@Entity
@ApiModel(description = "All details about the Admin. ")
@Table(name = "ADMIN_TBL")
public class Admin {


	@Id
	@ApiModelProperty(notes = "The database generated ID")
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	@Column(name = "ID")
	private long ID;

	@ApiModelProperty(notes = "Admin Id")
	@Column(name = "admin_id")
	private String adminId;

	@ApiModelProperty(notes = "Admin Password")
	@Column(name = "admin_pwd")
	private String pwd;


	public long getID() {return this.ID; }
	public String getAdminId() { return this.adminId;}
	public String getAdminPwd() { return this.pwd;}

	public void setID(long id) { this.ID = id;}
	public void setAdminId(String value) { this.adminId= value;}
	public void setAdminPwd(String value) { this.pwd = value;}
}
```

MODELS

Users.Java

```
package com.MyFirstSpringBootProj.sportyshoes.models;

import java.util.Date;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
```

```
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@AllArgsConstructor
@NoArgsConstructor
@Entity
@ApiModel(description = "All details about the User. ")
@Table(name = "USER_TBL")
public class Users {

	@Id
	@ApiModelProperty(notes = "The database generated user ID")
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	@Column(name = "ID")
	private long ID;

	@ApiModelProperty(notes = "user First Name")
	@Column(name = "fname")
	private String fname;

	@ApiModelProperty(notes = "user Last Name")
	@Column(name = "lname")
	private String lname;

	@ApiModelProperty(notes = "user Address")
	@Column(name = "address")
	private String address;

	@ApiModelProperty(notes = "user Age")
	@Column(name = "age")
	private int age;

	@ApiModelProperty(notes = "user Added Date")
	@Column(name = "date_added")
	private Date dateAdded;

	@ApiModelProperty(notes = "user Email-ID")
	@Column(name = "emailid")
	private String emailId;

	@ApiModelProperty(notes = "user Password")
	@Column(name = "pwd")
	private String pwd;

	@Override
	public String toString() {
		return "Users [ID=" + ID + ", fname=" + fname + ", lname=" + lname + ", address=" +
address + ", age=" + age
				+ ", dateAdded=" + dateAdded + ", emailId=" + emailId + ", pwd=" + pwd +
"]";
	}

	public long getID() {
```

```
	return ID;
}

public void setID(long iD) {
	ID = iD;
}

public String getFname() {
	return fname;
}

public void setFname(String fname) {
	this.fname = fname;
}

public String getLname() {
	return lname;
}

public void setLname(String lname) {
	this.lname = lname;
}

public String getAddress() {
	return address;
}

public void setAddress(String address) {
	this.address = address;
}

public int getAge() {
	return age;
}

public void setAge(int age) {
	this.age = age;
}

public Date getDateAdded() {
	return dateAdded;
}

public void setDateAdded(Date dateAdded) {
	this.dateAdded = dateAdded;
}

public String getEmailId() {
	return emailId;
}

public void setEmailId(String emailId) {
	this.emailId = emailId;
}

public String getPwd() {
	return pwd;
}

public void setPwd(String pwd) {
	this.pwd = pwd;
```

```
	}

}
```

CartItem.java

```
package com.MyFirstSpringBootProj.sportyshoes.models;

import java.util.Date;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@AllArgsConstructor
@NoArgsConstructor
@Entity
@ApiModel(description = "All details about the User. ")
@Table(name = "USER_TBL")
public class Users {

	@Id
	@ApiModelProperty(notes = "The database generated user ID")
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	@Column(name = "ID")
	private long ID;

	@ApiModelProperty(notes = "user First Name")
	@Column(name = "fname")
	private String fname;

	@ApiModelProperty(notes = "user Last Name")
	@Column(name = "lname")
	private String lname;

	@ApiModelProperty(notes = "user Address")
	@Column(name = "address")
	private String address;

	@ApiModelProperty(notes = "user Age")
	@Column(name = "age")
	private int age;

	@ApiModelProperty(notes = "user Added Date")
	@Column(name = "date_added")
	private Date dateAdded;
```

```
	@ApiModelProperty(notes = "user Email-ID")
	@Column(name = "emailid")
	private String emailId;

	@ApiModelProperty(notes = "user Password")
	@Column(name = "pwd")
	private String pwd;

	@Override
	public String toString() {
		return "Users [ID=" + ID + ", fname=" + fname + ", lname=" + lname + ", address=" +
address + ", age=" + age
				+ ", dateAdded=" + dateAdded + ", emailId=" + emailId + ", pwd=" + pwd +
"]";
	}

	public long getID() {
		return ID;
	}

	public void setID(long iD) {
		ID = iD;
	}

	public String getFname() {
		return fname;
	}

	public void setFname(String fname) {
		this.fname = fname;
	}

	public String getLname() {
		return lname;
	}

	public void setLname(String lname) {
		this.lname = lname;
	}

	public String getAddress() {
		return address;
	}

	public void setAddress(String address) {
		this.address = address;
	}

	public int getAge() {
		return age;
	}

	public void setAge(int age) {
		this.age = age;
	}

	public Date getDateAdded() {
		return dateAdded;
	}

	public void setDateAdded(Date dateAdded) {
```

```
		this.dateAdded = dateAdded;
	}

	public String getEmailId() {
		return emailId;
	}

	public void setEmailId(String emailId) {
		this.emailId = emailId;
	}

	public String getPwd() {
		return pwd;
	}

	public void setPwd(String pwd) {
		this.pwd = pwd;
	}

}
```

Category.java

```
package com.MyFirstSpringBootProj.sportyshoes.models;

import java.util.Date;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@AllArgsConstructor
@NoArgsConstructor
@Entity
@ApiModel(description = "All details about the User. ")
@Table(name = "USER_TBL")
public class Users {

	@Id
	@ApiModelProperty(notes = "The database generated user ID")
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	@Column(name = "ID")
	private long ID;

	@ApiModelProperty(notes = "user First Name")
	@Column(name = "fname")
	private String fname;

	@ApiModelProperty(notes = "user Last Name")
```

```
	@Column(name = "lname")
	private String lname;

	@ApiModelProperty(notes = "user Address")
	@Column(name = "address")
	private String address;

	@ApiModelProperty(notes = "user Age")
	@Column(name = "age")
	private int age;

	@ApiModelProperty(notes = "user Added Date")
	@Column(name = "date_added")
	private Date dateAdded;

	@ApiModelProperty(notes = "user Email-ID")
	@Column(name = "emailid")
	private String emailId;

	@ApiModelProperty(notes = "user Password")
	@Column(name = "pwd")
	private String pwd;

	@Override
	public String toString() {
		return "Users [ID=" + ID + ", fname=" + fname + ", lname=" + lname + ", address=" +
address + ", age=" + age
				+ ", dateAdded=" + dateAdded + ", emailId=" + emailId + ", pwd=" + pwd +
"]";
	}

	public long getID() {
		return ID;
	}

	public void setID(long iD) {
		ID = iD;
	}

	public String getFname() {
		return fname;
	}

	public void setFname(String fname) {
		this.fname = fname;
	}

	public String getLname() {
		return lname;
	}

	public void setLname(String lname) {
		this.lname = lname;
	}

	public String getAddress() {
		return address;
	}

	public void setAddress(String address) {
		this.address = address;
```

```
	}

	public int getAge() {
		return age;
	}

	public void setAge(int age) {
		this.age = age;
	}

	public Date getDateAdded() {
		return dateAdded;
	}

	public void setDateAdded(Date dateAdded) {
		this.dateAdded = dateAdded;
	}

	public String getEmailId() {
		return emailId;
	}

	public void setEmailId(String emailId) {
		this.emailId = emailId;
	}

	public String getPwd() {
		return pwd;
	}

	public void setPwd(String pwd) {
		this.pwd = pwd;
	}

}
```

Product.java

```
package com.MyFirstSpringBootProj.sportyshoes.models;
import java.math.BigDecimal;
import java.util.Date;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;
import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
@Data
@AllArgsConstructor
@NoArgsConstructor
@Entity
@ApiModel(description = "All details about the Products. ")
@Table(name = "PRODUCT_TBL")
public class Product {
```

```
	@Id
	@ApiModelProperty(notes = "The database generated ID")
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	@Column(name = "ID")
	private long ID;

	@ApiModelProperty(notes = "Product Name")
	@Column(name = "name")
	private String name;

	@ApiModelProperty(notes = "Product Price")
	@Column(name = "price")
	private BigDecimal price;

	@ApiModelProperty(notes = "Product Added Date")
	@Column(name = "date_added")
	private Date dateAdded;

	@ApiModelProperty(notes = "Product Category ID")
	@Column(name = "category_id")
	private long categoryId;


	public long getID() {return this.ID; }
	public String getName() { return this.name;}
	public BigDecimal getPrice() { return this.price;}
	public long getCategoryId() { return this.categoryId;}
	public Date getDateAdded() { return this.dateAdded;}


	public void setID(long id) { this.ID = id;}
	public void setName(String value) { this.name = value;}
	public void setPrice(BigDecimal value) { this.price = value;}
	public void setCategoryId(long value) { this.categoryId = value;}
	public void setDateAdded(Date date) { this.dateAdded = date;}
}
```

Purchases.java

```
package com.MyFirstSpringBootProj.sportyshoes.models;

import java.math.BigDecimal;
import java.util.Date;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.Getter;
import lombok.NoArgsConstructor;
```

```
import lombok.Setter;


@Data
@Getter
@Setter
@AllArgsConstructor
@NoArgsConstructor
@Entity
@ApiModel(description = "All details about the User Purchase. ")
@Table(name = "PURCHASE_TBL")
public class Purchase {


	@Id
	@ApiModelProperty(notes = "The database generated ID")
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	@Column(name = "ID")
	private long ID;

	@ApiModelProperty(notes = "User ID")
	@Column(name = "user_id")
	private long userId;

	@ApiModelProperty(notes = "Date of Added")
	@Column(name = "date")
	private Date date;

	@ApiModelProperty(notes = "Total")
	@Column(name = "gross_total")
	private BigDecimal total;


	public long getID() {return this.ID; }
	public long getUserId() { return this.userId;}
	public BigDecimal getTotal() { return this.total;}
	public Date getDate() { return this.date;}

	public void setID(long id) { this.ID = id;}
	public void setUserId(long value) { this.userId = value;}
	public void setTotal(BigDecimal value) { this.total = value;}
	public void setDate(Date date) { this.date = date;}
}
```

PurchaseItem.java

```
package com.MyFirstSpringBootProj.sportyshoes.models;

import java.math.BigDecimal;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
```

```
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;


@Data
@AllArgsConstructor
@NoArgsConstructor
@Entity
@ApiModel(description = "All details about the Purchase Items. ")
@Table(name = "PURCHASEITM_TBL")
public class PurchaseItem {


	@Id
	@ApiModelProperty(notes = "The database generated ID")
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	@Column(name = "ID")
	private long ID;

	@ApiModelProperty(notes = "Purchase ID")
	@Column(name = "purchase_id")
	private long purchaseId;

	@ApiModelProperty(notes = " Product ID")
	@Column(name = "product_id")
	private long productId;

	@ApiModelProperty(notes = "user ID")
	@Column(name = "user_id")
	private long userId;

	@ApiModelProperty(notes = "ProductId")
	@Column(name = "rate")
	private BigDecimal rate;

	@ApiModelProperty(notes = "Purchase Quantity")
	@Column(name = "qty")
	private int qty;

	@ApiModelProperty(notes = "Product Price")
	@Column(name = "price")
	private BigDecimal price;


	public long getID() {return this.ID; }
	public long getPurchaseId() { return this.purchaseId;}
	public long getProductId() { return this.productId;}
	public long getUserId() { return this.userId;}
	public BigDecimal getRate() { return this.rate;}
	public int getQty() { return this.qty;}
	public BigDecimal getPrice() { return this.price;}

	public void setID(long id) { this.ID = id;}
	public void setPurchaseId(long value) { this.purchaseId = value;}
	public void setProductId(long value) { this.productId = value;}
	public void setUserId(long value) { this.userId = value;}
	public void setRate(BigDecimal value) { this.rate = value;}
	public void setQty(int value) { this.qty= value;}
	public void setPrice(BigDecimal value) { this.price= value;}
```

```
}
```

SERVICES

AdminServices.java

```
package com.MyFirstSpringBootProj.sportyshoes.services;

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

import org.springframework.stereotype.Component;
import org.springframework.transaction.annotation.Transactional;

import com.MyFirstSpringBootProj.sportyshoes.models.Admin;

import com.MyFirstSpringBootProj.sportyshoes.repositories.AdminRepository;

@Component
public class AdminService {

	@Autowired
	private AdminRepository adminRepo;

	@Transactional
	private Admin getAdminByAdminId(String adminId) {
		Admin admin = adminRepo.findByAdminId(adminId);
		return admin;
	}

	@Transactional
	public Admin getAdminById(long id) {
		return adminRepo.findById(id).orElse(null);
	}

	@Transactional
	public void updatePwd(Admin admin) {
		Admin existingAdmin = adminRepo.findById((long) admin.getID()).orElse(null);
		existingAdmin.setAdminPwd(admin.getAdminPwd());
		adminRepo.save(admin);
	}

	@Transactional
	public Admin authenticate(String adminId, String pwd) {
		Admin admin = getAdminByAdminId(adminId);
		if (!(adminId == null) && !(pwd == null) && adminId.equals(admin.getAdminId())
				&& pwd.equals(admin.getAdminPwd())) {
			return admin;
		}
		return null;

	}

}
```

CategoryService.java

```
package com.MyFirstSpringBootProj.sportyshoes.services;

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

import org.springframework.stereotype.Component;
import org.springframework.transaction.annotation.Transactional;

import com.MyFirstSpringBootProj.sportyshoes.models.Admin;

import com.MyFirstSpringBootProj.sportyshoes.repositories.AdminRepository;

@Component
public class AdminService {

	@Autowired
	private AdminRepository adminRepo;

	@Transactional
	private Admin getAdminByAdminId(String adminId) {
		Admin admin = adminRepo.findByAdminId(adminId);
		return admin;
	}

	@Transactional
	public Admin getAdminById(long id) {
		return adminRepo.findById(id).orElse(null);
	}

	@Transactional
	public void updatePwd(Admin admin) {
		Admin existingAdmin = adminRepo.findById((long) admin.getID()).orElse(null);
		existingAdmin.setAdminPwd(admin.getAdminPwd());
		adminRepo.save(admin);
	}

	@Transactional
	public Admin authenticate(String adminId, String pwd) {
		Admin admin = getAdminByAdminId(adminId);
		if (!(adminId == null) && !(pwd == null) && adminId.equals(admin.getAdminId())
				&& pwd.equals(admin.getAdminPwd())) {
			return admin;
		}
		return null;

	}

}
```

ProductService.java

```
package com.MyFirstSpringBootProj.sportyshoes.services;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.jpa.repository.Query;

import org.springframework.stereotype.Component;
```

```
import org.springframework.transaction.annotation.Transactional;

import com.MyFirstSpringBootProj.sportyshoes.models.Product;

import com.MyFirstSpringBootProj.sportyshoes.repositories.ProductRepository;


@Component
public class ProductService {

	@Autowired
	private ProductRepository productRepo;


		@Transactional
		public Product getProductById(long id) {
			return productRepo.findById(id).orElse(null);
		}


		@Transactional
		public void updateProduct(Product product) {

			Product existingProduct = productRepo.findById((long)
product.getID()).orElse(null);
			if(existingProduct!=null) {
			existingProduct.setName(product.getName());
			existingProduct.setPrice(product.getPrice());
			existingProduct.setDateAdded(product.getDateAdded());
			productRepo.save(existingProduct);}
			else {
				productRepo.save(product);
			}
		}


		@Transactional
		public void deleteProduct(long id) {

			productRepo.deleteById(id);
		}

		@Transactional
		@Query("FROM City ORDER BY name ASC")
		public List<Product> getAllProducts() {
			return productRepo.findAll();
		}

//		@Transactional
//		public List<Object> getAllProductsWithJoin() {
//			return productRepo.getAllProductsWithJoin();
//		}
//
}
```

PurchaseItemService.java

```
package com.MyFirstSpringBootProj.sportyshoes.services;

import java.util.List;


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

import org.springframework.stereotype.Component;
import org.springframework.transaction.annotation.Transactional;

import com.MyFirstSpringBootProj.sportyshoes.models.PurchaseItem;
import com.MyFirstSpringBootProj.sportyshoes.repositories.PurchaseItemReporsitory;


@Component
public class PurchaseItemService {

	@Autowired
	private PurchaseItemReporsitory purchaseItemRepo;

	@Transactional
	      public PurchaseItem getItemById(long id) {
	       return purchaseItemRepo.findById(id).orElse(null);

	      }

	@Transactional
	      public List<PurchaseItem> getAllItemsByPurchaseId(long purchaseId) {
	       return  purchaseItemRepo.findAllByID(purchaseId);
	      }

	@Transactional
	      public void updateItem(PurchaseItem item) {
	       PurchaseItem pItem = purchaseItemRepo.findById((long) item.getID()).orElse(null);
	       if(pItem!=null) {
	       pItem.setPrice(item.getPrice());
	       pItem.setProductId(item.getProductId());
	       pItem.setPurchaseId(item.getPurchaseId());
	       pItem.setQty(item.getQty());
	       pItem.setRate(item.getRate());
	       pItem.setUserId(item.getUserId());
	       purchaseItemRepo.save(pItem);
	       }else { purchaseItemRepo.save(item);}
	      }


	@Transactional
	      public void deleteItem(long id) {
	       purchaseItemRepo.deleteById(id);
	      }

	@Transactional
	      public void deleteAllItemsForPurchaseId(long purchaseId) {
	       purchaseItemRepo.deleteAllItemsByPurchaseId(purchaseId);
	      }

}
```

PurchaseService.java

```
package com.MyFirstSpringBootProj.sportyshoes.services;
import java.util.Calendar;
import java.util.List;


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

import org.springframework.stereotype.Component;
import org.springframework.transaction.annotation.Transactional;

import com.MyFirstSpringBootProj.sportyshoes.models.Purchase;
import com.MyFirstSpringBootProj.sportyshoes.repositories.PurchaseRepository;



@Component
public class PurchaseService {

	@Autowired
	private PurchaseRepository purchaseRepo;

		@Transactional
		public Purchase getPurchaseById(long id) {
			return purchaseRepo.findById(id).orElse(null);
		}

		@Transactional
		public List<Purchase> getAllItems() {
			return purchaseRepo.findAll();
		}

		@Transactional
		public List<Purchase> getAllItemsByUserId(long userId) {
			return purchaseRepo.getAllItemsByUserId(userId);
		}

		@Transactional
		public Purchase  updatePurchase(Purchase purchase) {
			Purchase purchas= purchaseRepo.findById((long) purchase.getID()).orElse(null);
			if(purchas!=null) {
			purchas.setUserId(purchase.getUserId());
			purchas.setTotal(purchase.getTotal());
			purchas.setDate(Calendar.getInstance().getTime());
			return purchaseRepo.save(purchas);
			}else {
				return purchaseRepo.save(purchase);
			}
		}


		@Transactional
		public void deletePurchase(long id) {
			purchaseRepo.deleteById(id);
		}

}
```

UserService.java

```
package com.MyFirstSpringBootProj.sportyshoes.services;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;
import org.springframework.transaction.annotation.Transactional;

import com.MyFirstSpringBootProj.sportyshoes.models.Users;
import com.MyFirstSpringBootProj.sportyshoes.repositories.UserRepository;

@Component
public class UserService {

	@Autowired
	private UserRepository userRepo;

	@Transactional
	public Users addUsers(Users user) {

		return userRepo.save(user);

	}

	@Transactional
	public List<Users> getAllUsers() {
		return (List<Users>) userRepo.findAll();
	}

	@Transactional
	public Users getUserById(long l) {
		return userRepo.findById((long) l).orElse(null);
	}

	@Transactional
	public Users getUserByEmailId(String emailId) {
		return userRepo.findByEmailId(emailId);
	}

	@Transactional
	public void updateUser(Users user) {
		Users existingUser = userRepo.findById((long) user.getID()).orElse(null);

		existingUser.setFname(user.getFname());
		existingUser.setLname(user.getLname());
		existingUser.setAge(user.getAge());
		existingUser.setAddress(user.getAddress());
		existingUser.setEmailId(user.getEmailId());
		existingUser.setPwd(user.getPwd());
		existingUser.setDateAdded(user.getDateAdded());
		userRepo.save(existingUser);
	}
	@Transactional
	public void deleteUserById(long Id) {
		 userRepo.deleteById(Id);
	}

}
```

Controllers

AdminController

```
package com.MyFirstSpringBootProj.sportyshoes.controllers;

import java.math.BigDecimal;
import java.util.HashMap;
import java.util.List;

import javax.servlet.http.HttpSession;

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

import org.springframework.stereotype.Controller;

import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;

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

import com.MyFirstSpringBootProj.sportyshoes.models.Admin;
import com.MyFirstSpringBootProj.sportyshoes.models.Category;
import com.MyFirstSpringBootProj.sportyshoes.models.Product;
import com.MyFirstSpringBootProj.sportyshoes.models.Purchase;
import com.MyFirstSpringBootProj.sportyshoes.models.PurchaseItem;
import com.MyFirstSpringBootProj.sportyshoes.models.Users;
import com.MyFirstSpringBootProj.sportyshoes.services.AdminService;
import com.MyFirstSpringBootProj.sportyshoes.services.CategoryService;
import com.MyFirstSpringBootProj.sportyshoes.services.ProductService;
import com.MyFirstSpringBootProj.sportyshoes.services.PurchaseItemService;
import com.MyFirstSpringBootProj.sportyshoes.services.PurchaseService;
import com.MyFirstSpringBootProj.sportyshoes.services.UserService;

import io.swagger.annotations.Api;

@Api(value = "Admin Controller For Sporty SHoes Application")
@Controller
public class AdminController {

	@Autowired
	private AdminService adminService;

	@Autowired
	private CategoryService categoryService;

	@Autowired
	private ProductService productService;

	@Autowired
	private PurchaseService purchaseService;

	@Autowired
	private PurchaseItemService purchaseItemService;

	@Autowired
```

```
	private UserService userService;

//-----------------------------------------------------------------
	@GetMapping("/adminlogin")
	public String login(ModelMap map, javax.servlet.http.HttpServletRequest request) {
		map.addAttribute("pageTitle", "ADMIN LOGIN");
		return "admin/login";
	}

	// -----------------------------------------------------------------
	@PostMapping("/adminloginaction")
	public String loginAction(ModelMap map, javax.servlet.http.HttpServletRequest request,
			@RequestParam(value = "admin_id", required = true) String adminId,
			@RequestParam(value = "admin_pwd", required = true) String adminPwd) {

		Admin admin = adminService.authenticate(adminId, adminPwd);

		if (admin == null) {
			map.addAttribute("error", "Admin login failed");
			return "admin/login";
		}
		// store admin id in session

		HttpSession session = request.getSession();
		session.setAttribute("admin_id", admin.getID());
		 return "admin/dashboard";
	}

//-------------------------------------------------------------------------
	@GetMapping("/admindashboard")
	public String dashboard(ModelMap map, javax.servlet.http.HttpServletRequest request) {
		// check if session is still alive
		HttpSession session = request.getSession();
		if (session.getAttribute("admin_id") == null) {
			return "admin/login";
		}

		map.addAttribute("pageTitle", "ADMIN DASHBOARD");
		return "admin/dashboard";
	}

	// -------------------------------------------------------------------------
	@GetMapping("/adminchangepassword")
	public String changePwd(ModelMap map, javax.servlet.http.HttpServletRequest request) {
		// check if session is still alive
		HttpSession session = request.getSession();
		if (session.getAttribute("admin_id") == null) {
			return "admin/login";
		}

		com.MyFirstSpringBootProj.sportyshoes.models.Admin admin = adminService
				.getAdminById((Long) session.getAttribute("admin_id"));

		map.addAttribute("admin", admin);
		map.addAttribute("pageTitle", "ADMIN CHANGE PASSWORD");
		return "admin/change-password";
	}

	// -------------------------------------------------------------------------

	@PostMapping("/adminchangepwdaction")
```

```
	public String updatePassword(ModelMap map, @RequestParam(value = "pwd", required = true)
String pwd,
			@RequestParam(value = "pwd2", required = true) String pwd2,
javax.servlet.http.HttpServletRequest request) {
		// check if session is still alive
		HttpSession session = request.getSession();
		if (session.getAttribute("admin_id") == null) {
			return "admin/login";
		}

		if (pwd == null || pwd2 == null || pwd.equals("") || pwd2.equals("")) {
			map.addAttribute("error", "Error , Incomplete passwords submitted.");
			return "admin/change-password";
		}
		if (!pwd.equals(pwd2)) {
			map.addAttribute("error", "Error , Passwords do not match.");
			return "admin/change-password";
		}
		Admin admin = adminService.getAdminById((Long) session.getAttribute("admin_id"));
		admin.setAdminPwd(pwd);
		adminService.updatePwd(admin);

		return "admin/dashboard";
	}

	// ------------------------------------------------------------------------------

	@GetMapping("/adminproducts")
	public String products(ModelMap map, javax.servlet.http.HttpServletRequest request) {
		// check if session is still alive
		HttpSession session = request.getSession();
		if (session.getAttribute("admin_id") == null) {
			return "admin/login";
		}
		List<Product> list = productService.getAllProducts();

		// use a MAP to link category names to each product in list
		HashMap<Long, String> mapCats = new HashMap<Long, String>();

		for (Product product : list) {
			Category category = categoryService.getCategoryById(product.getCategoryId());
			if (category != null)
				mapCats.put(product.getID(), category.getName());
		}
		map.addAttribute("list", list);
		map.addAttribute("mapCats", mapCats);

		map.addAttribute("pageTitle", "ADMIN SETUP PRODUCTS");
		return "admin/products";
	}

	// ------------------------------------------------------------------------
	@GetMapping("/admincategories")
	public String categories(ModelMap map, javax.servlet.http.HttpServletRequest request) {
		// check if session is still alive
		HttpSession session = request.getSession();
		if (session.getAttribute("admin_id") == null) {
			return "admin/login";
		}

		List<Category> list = categoryService.getAllCategories();
```

```
            map.addAttribute("list", list);
            map.addAttribute("pageTitle", "ADMIN SETUP PRODUCT CATEGORIES");
            return "admin/categories";
      }

//---------------------------------------------------------------------------
      @GetMapping("/adminmembers")
      public String members(ModelMap map, javax.servlet.http.HttpServletRequest request) {
            // check if session is still alive
            HttpSession session = request.getSession();
            if (session.getAttribute("admin_id") == null) {
                  return "admin/login";
            }
            List<Users> list = userService.getAllUsers();

            map.addAttribute("list", list);
            map.addAttribute("pageTitle", "ADMIN BROWSE MEMBERS");
            return "admin/members";
      }

//-----------------------------------------------------------------------------
      @GetMapping("/adminpurchases")
      public String purchases(ModelMap map, javax.servlet.http.HttpServletRequest request) {
            // check if session is still alive
            HttpSession session = request.getSession();
            if (session.getAttribute("admin_id") == null) {
                  return "admin/login";
            }

            List<Purchase> list = purchaseService.getAllItems();

            BigDecimal total = new BigDecimal(0.0);

            for (Purchase purchase : list) {
                  total = total.add(purchase.getTotal());
            }

            // use MAPs to mape users to each purchase and item names to each purchase item
            // row
            HashMap<Long, String> mapItems = new HashMap<Long, String>();
            HashMap<Long, String> mapUsers = new HashMap<Long, String>();

            for (Purchase purchase : list) {
                  total = total.add(purchase.getTotal());
                  Users user = userService.getUserById(purchase.getUserId());
                  if (user != null)
                        mapUsers.put(purchase.getID(), user.getFname() + " " + user.getLname());

                  List<PurchaseItem> itemList =
purchaseItemService.getAllItemsByPurchaseId(purchase.getID());
                  StringBuilder sb = new StringBuilder("");
                  for (PurchaseItem item : itemList) {
                        Product product = productService.getProductById(item.getProductId());
                        if (product != null)
                              sb.append(product.getName() + ", " + item.getQty() + " units @" +
item.getRate() + " = "
                                          + item.getPrice() + "<br>");
                  }
                  mapItems.put(purchase.getID(), sb.toString());
            }
            map.addAttribute("totalAmount", total);
```

```
            map.addAttribute("list", list);
            map.addAttribute("mapItems", mapItems);
            map.addAttribute("mapUsers", mapUsers);
            map.addAttribute("pageTitle", "ADMIN PURCHASES REPORT");
            return "admin/purchases";
      }

      // ------------------------------------------------------------------------------------
      @DeleteMapping("/admindeletecat")
      public String deleteCategory(ModelMap map, @RequestParam(value = "id", required = true)
String id,
                  javax.servlet.http.HttpServletRequest request) {
            // check if session is still alive
            HttpSession session = request.getSession();
            if (session.getAttribute("admin_id") == null) {
                  return "admin/login";
            }
            long idValue = Long.parseLong(id);
            Category category = new Category();
            if (idValue > 0) {
                  categoryService.deleteCategory(idValue);
            }
            return "redirect:admincategories";
      }

//------------------------------------------------------------------------------------
      @GetMapping("/admineditcat")
      public String editCategory(ModelMap map, @RequestParam(value = "id", required = true)
String id,
                  javax.servlet.http.HttpServletRequest request) {
            // check if session is still alive
            HttpSession session = request.getSession();
            if (session.getAttribute("admin_id") == null) {
                  return "admin/login";
            }
            long idValue = Long.parseLong(id);
            Category category = new Category();
            if (idValue > 0) {
                  category = categoryService.getCategoryById(idValue);
            } else {
                  category.setID(0);
            }
            map.addAttribute("category", category);
            map.addAttribute("pageTitle", "ADMIN EDIT PRODUCT CATEGORY");
            return "admin/edit-category";
      }

      // ------------------------------------------------------------------------------------
-----
      @PostMapping("/admineditcataction")
      public String updateCategory(ModelMap map, @RequestParam(value = "id", required = true)
String id,
                  @RequestParam(value = "name", required = true) String name,
javax.servlet.http.HttpServletRequest request) {
            long idValue = Long.parseLong(id);

            if (name == null || name.equals("")) {
                  map.addAttribute("error", "Error, A category name must be specified");
                  return "redirect:admineditcat?id=" + id;
            }
            Category category = new Category();
```

```
			category.setID(idValue);
			category.setName(name);

			categoryService.updateCategory(category);

			return "redirect:admincategories";
		}

//-----------------------------------------------------------------------------
		@DeleteMapping("/admindeleteproduct")
		public String deleteProduct(ModelMap map, @RequestParam(value = "id", required = true)
String id,
				javax.servlet.http.HttpServletRequest request) {
			// check if session is still alive
			HttpSession session = request.getSession();
			if (session.getAttribute("admin_id") == null) {
				return "admin/login";
			}
			long idValue = Long.parseLong(id);
			Product product = new Product();
			if (idValue > 0) {
				productService.deleteProduct(idValue);
			}
			return "redirect:adminproducts";
		}

//-------------------------------------------------------------------------------
		@GetMapping("/admineditproduct")
		public String editProduct(ModelMap map, @RequestParam(value = "id", required = true) String
id,
				javax.servlet.http.HttpServletRequest request) {
			// check if session is still alive
			HttpSession session = request.getSession();
			if (session.getAttribute("admin_id") == null) {
				return "admin/login";
			}

			long idValue = Long.parseLong(id);
			Product product = new Product();
			if (idValue > 0) {
				product = productService.getProductById(idValue);
			} else {
				product.setID(0);
				product.setCategoryId(0);
			}
			String dropDown = categoryService.getCategoriesDropDown(product.getCategoryId());
			map.addAttribute("product", product);
			map.addAttribute("catDropdown", dropDown);
			map.addAttribute("pageTitle", "ADMIN EDIT PRODUCT");
			return "admin/edit-product";
		}

//-------------------------------------------------------------------------------
		@PostMapping("/admineditproductaction")
		public String updateProduct(ModelMap map, javax.servlet.http.HttpServletRequest request,
				@RequestParam(value = "id", required = true) String id,
				@RequestParam(value = "name", required = true) String name,
				@RequestParam(value = "price", required = true) String price,
				@RequestParam(value = "category", required = true) String categoryId) {
			long idValue = Long.parseLong(id);
			long categoryIdValue = Long.parseLong(categoryId);
```

```
		BigDecimal priceValue = new BigDecimal(0.0);

		if (name == null || name.equals("")) {
			map.addAttribute("error", "Error, A product name must be specified");
			return "redirect:admineditproduct?id=" + id;
		}
		try {
			priceValue = new BigDecimal(price);

		} catch (Exception ex) {
			map.addAttribute("error", "Error, Price is invalid");
			return "redirect:admineditproduct?id=" + id;
		}

		Product product = new Product();
		product.setID(idValue);
		product.setCategoryId(Long.parseLong(categoryId));
		product.setName(name);
		product.setPrice(priceValue);

		productService.updateProduct(product);

		return "redirect:adminproducts";
	}


	//-----------------------------------------------------------------------------------------

		@GetMapping("/adminlogout")

		public String logout(ModelMap map, javax.servlet.http.HttpServletRequest request) {

			HttpSession session = request.getSession();

			session.invalidate();


			return "admin/login";

		}

	}
```

CartController

```
package com.MyFirstSpringBootProj.sportyshoes.controllers;

import java.math.BigDecimal;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.List;

import javax.servlet.http.HttpSession;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.DeleteMapping;
```

```
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;
import com.MyFirstSpringBootProj.sportyshoes.models.CartItem;
import com.MyFirstSpringBootProj.sportyshoes.models.Product;
import com.MyFirstSpringBootProj.sportyshoes.models.Purchase;
import com.MyFirstSpringBootProj.sportyshoes.models.PurchaseItem;
import com.MyFirstSpringBootProj.sportyshoes.services.ProductService;
import com.MyFirstSpringBootProj.sportyshoes.services.PurchaseItemService;
import com.MyFirstSpringBootProj.sportyshoes.services.PurchaseService;

import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;

@Api(value="Cart Controller of Sporty Shoes Application")
@Controller
public class CartController {

	@Autowired
	private ProductService productService;

	@Autowired
	private PurchaseService purchaseService;

	@Autowired
	private PurchaseItemService purchaseItemService;

	@ApiOperation(value = "User/Admin Will redirect to Cast Page")
	  @SuppressWarnings("unchecked")
	@GetMapping("/cart")
	    public String cart(ModelMap map, javax.servlet.http.HttpServletRequest request)
	    {
	        // check if user is logged in
	        HttpSession session = request.getSession();
	        if (session.getAttribute("user_id") == null) {
	            map.addAttribute("error", "Error, You need to login before adding items to
cart");
	        } else {
	            // if cart is already in session then retrieve it else create a new cart
list and
	            // save it to session
	            List<CartItem> cartItems = new ArrayList<CartItem>();
	            if (session.getAttribute("cart_items") != null)
	                cartItems = (List<CartItem>) session.getAttribute("cart_items");

	            // get total of all cart items
	            BigDecimal totalValue = getCartValue(cartItems);
	            map.addAttribute("cartValue", totalValue);
	            map.addAttribute("cartItems", cartItems);
	        }

	        map.addAttribute("pageTitle", "SPORTY SHOES - YOUR CART");
	       return "cart";
	    }
	@ApiOperation(value = "Add product to cart")
	  @SuppressWarnings("unchecked")
	@GetMapping("/cartadditem")
	    public String cartAddItem(ModelMap map, javax.servlet.http.HttpServletRequest request,
	            @RequestParam(value="id", required=true) String productId)
	    {
	        // check if user is logged in
	        HttpSession session = request.getSession();
```

```
		if (session.getAttribute("user_id") == null) {
			map.addAttribute("error", "Error, You need to login before adding items to
cart");
		} else {

			long idValue = Long.parseLong(productId);
			// if cart is already in session then retrieve it else create a new cart
list and
			// save it to session
			List<CartItem> cartItems = new ArrayList<CartItem>();
			if (session.getAttribute("cart_items") != null)
				cartItems = (List<CartItem>) session.getAttribute("cart_items");
			if (isItemInCart(cartItems, idValue)) {
				map.addAttribute("error", "This item is already in your cart");
			} else {
				Product product = productService.getProductById(idValue);
				CartItem item = new CartItem();
				item.setProductId(idValue);
				item.setQty(1);
				item.setRate(product.getPrice());
				BigDecimal dprice = item.getRate().multiply(new
BigDecimal(item.getQty()));
				item.setPrice(dprice);
				item.setName(product.getName());
				cartItems.add(item);

				session.setAttribute("cart_items", cartItems);
			}
		}

		return "redirect:cart";
	}

	@ApiOperation(value = "Delete the product from the cart by ID")
	  @DeleteMapping("/cartdeleteitem")
	  public String cartDeleteItem(ModelMap map, javax.servlet.http.HttpServletRequest
request,
			@RequestParam(value="id", required=true) String id)
	{
		// check if user is logged in
		HttpSession session = request.getSession();
		if (session.getAttribute("user_id") == null) {
			map.addAttribute("error", "Error, You need to login before deleting items
from cart");
		} else {
			long idValue = Long.parseLong(id);
			List<CartItem> cartItems = new ArrayList<CartItem>();
			if (session.getAttribute("cart_items") != null)
				cartItems = (List<CartItem>) session.getAttribute("cart_items");

			for(CartItem item: cartItems) {
				if (item.getProductId() == idValue) {
					cartItems.remove(item);
					session.setAttribute("cart_items", cartItems);
					break;
				}
			 }
		}
		return "redirect:cart";
	}
```

```
	@ApiOperation(value = "Procduct Checkout")
	  @SuppressWarnings("unchecked")
	@GetMapping("/checkout")
	    public String checkout(ModelMap map, javax.servlet.http.HttpServletRequest request)
	    {
	        // check if user is logged in
	        HttpSession session = request.getSession();
	        if (session.getAttribute("user_id") == null) {
	            map.addAttribute("error", "Error, You need to login before checking out");
	        } else {
	            List<CartItem> cartItems = new ArrayList<CartItem>();
	            if (session.getAttribute("cart_items") != null)
	                cartItems = (List<CartItem>) session.getAttribute("cart_items");
	            BigDecimal totalValue = getCartValue(cartItems);
	            map.addAttribute("cartValue", totalValue);
	            map.addAttribute("cartItems", cartItems);
	        }
	        map.addAttribute("pageTitle", "SPORTY SHOES - CHECKOUT");
	       return "checkout";
	    }

	@ApiOperation(value = "Complete Purchase")
	  @SuppressWarnings("unchecked")
	@GetMapping("/completepurchase")
	    public String completePurchase(ModelMap map, javax.servlet.http.HttpServletRequest
request)
	    {
	        // check if user is logged in
	        HttpSession session = request.getSession();
	        if (session.getAttribute("user_id") == null) {
	            map.addAttribute("error", "Error, You need to login before completing
purchase");
	        } else {
	            // take items from cart and update the databae
	            List<CartItem> cartItems = new ArrayList<CartItem>();
	            if (session.getAttribute("cart_items") != null)
	                cartItems = (List<CartItem>) session.getAttribute("cart_items");
	            BigDecimal totalValue = getCartValue(cartItems);

	            long userId = (Long) session.getAttribute("user_id") ;

	            Purchase purchase = new Purchase();
	            purchase.setUserId(userId);
	            purchase.setDate(Calendar.getInstance().getTime());
	            purchase.setTotal(totalValue);
	            long purchaseId = purchaseService.updatePurchase(purchase).getID();

	            for(CartItem item: cartItems) {
	                PurchaseItem pItem = new PurchaseItem();
	                pItem.setPurchaseId(purchaseId);
	                pItem.setProductId(item.getProductId());
	                pItem.setUserId(userId);
	                pItem.setRate(item.getRate());
	                pItem.setQty(item.getQty());
	                pItem.setPrice(item.getPrice());

	                purchaseItemService.updateItem(pItem);
	            }
	            map.addAttribute("cartValue", totalValue);
	            map.addAttribute("cartItems", cartItems);
```

```
            }

            return "redirect:confirm";
        }


    @ApiOperation(value = "Payment Gatway")
      @SuppressWarnings("unchecked")
    @GetMapping("/gateway")
        public String gateway(ModelMap map, javax.servlet.http.HttpServletRequest request)
        {
            // check if user is logged in
            HttpSession session = request.getSession();
            if (session.getAttribute("user_id") == null) {
                  map.addAttribute("error", "Error, You need to login before making payment");
            } else {
                  List<CartItem> cartItems = new ArrayList<CartItem>();
                  if (session.getAttribute("cart_items") != null)
                        cartItems = (List<CartItem>) session.getAttribute("cart_items");
                  BigDecimal totalValue = getCartValue(cartItems);
                  map.addAttribute("cartValue", totalValue);
                  map.addAttribute("cartItems", cartItems);

            }

            map.addAttribute("pageTitle", "SPORTY SHOES - PAYMENT GATEWAY");
            return "gateway";
        }


    @ApiOperation(value = "Payment Confirmation")
      @SuppressWarnings("unchecked")
    @GetMapping("/confirm")
        public String confirm(ModelMap map, javax.servlet.http.HttpServletRequest request)
        {
            // check if user is logged in
            HttpSession session = request.getSession();
            if (session.getAttribute("user_id") == null) {
                  map.addAttribute("error", "Error, You need to login before completing the
purchase");
            } else {
                  // clear items from cart as order has completed
                  List<CartItem> cartItems = new ArrayList<CartItem>();
                  if (session.getAttribute("cart_items") != null)
                        cartItems = (List<CartItem>) session.getAttribute("cart_items");
                  BigDecimal totalValue = getCartValue(cartItems);
                  map.addAttribute("cartValue", totalValue);


                  cartItems.clear();
                  session.setAttribute("cart_items", null);
            }
            map.addAttribute("pageTitle", "SPORTY SHOES - PURCHASE CONFIRMATION");
            return "confirm";
        }

      /**
       * Check if an item is already in the cart
       * @param list
       * @param item
       * @return
```

```
	 */
	private boolean isItemInCart(List<CartItem> list, long item) {
		boolean retVal = false;

		for(CartItem thisItem: list) {
			if (item == thisItem.getProductId()) {
				retVal = true;
				break;
			}
		}
		return retVal;
	}

	/**
	 * Get total value of items in cart
	 * @param list
	 * @return
	 */
	private BigDecimal getCartValue(List<CartItem> list) {
		BigDecimal total = new BigDecimal(0.0);

		for(CartItem item: list) {
			BigDecimal dprice = item.getRate().multiply(new BigDecimal(item.getQty()));
			total= total.add(dprice);
		 }
		return total;
	}

}
```

DashboardController.java

```
package com.MyFirstSpringBootProj.sportyshoes.controllers;


import org.springframework.stereotype.Controller;

import org.springframework.ui.ModelMap;

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


import io.swagger.annotations.Api;

import io.swagger.annotations.ApiOperation;


@Api(value="Dash board Controller For Sporty SHoes Application")

@Controller

public class DashboardController {
```

```
        @ApiOperation(value = "On succussfull Login user will redirect to \"Dashboard\"  or will redirect to Login page")
        @GetMapping("/dashboard")
    public String dashboard(ModelMap map)
    {
            map.addAttribute("pageTitle", "SPORTY SHOES - DASHBOARD");
       return "dashboard";
    }

}
```

HomeController.java

```
        package com.MyFirstSpringBootProj.sportyshoes.controllers;

        import java.util.HashMap;
        import java.util.List;

        import javax.servlet.http.HttpSession;

        import org.springframework.beans.factory.annotation.Autowired;
        import org.springframework.stereotype.Controller;
        import org.springframework.ui.ModelMap;
        import org.springframework.web.bind.annotation.GetMapping;


        import com.MyFirstSpringBootProj.sportyshoes.models.Category;
        import com.MyFirstSpringBootProj.sportyshoes.models.Product;
        import com.MyFirstSpringBootProj.sportyshoes.services.CategoryService;
        import com.MyFirstSpringBootProj.sportyshoes.services.ProductService;

        import io.swagger.annotations.ApiOperation;
```

```
@Controller
public class HomeController {
	@Autowired
	private CategoryService categoryService;

	@Autowired
	private ProductService productService;

	@ApiOperation(value = "On The Application Load User will be landed on to Index or Home Page")
	 @GetMapping({"/", "/home"})
	  public String home(ModelMap map,  javax.servlet.http.HttpServletRequest request)
	  {
			HttpSession session = request.getSession();
				List<Product> list = productService.getAllProducts();

				// use MAP to map the category names to product rows
				 HashMap<Long, String> mapCats = new HashMap<Long, String>();
				 for(Product product: list) {
					Category category =
categoryService.getCategoryById(product.getCategoryId());
					if (category != null)
						mapCats.put(product.getID(), category.getName());
				 }
				 // session.setAttribute(name, value);
				map.addAttribute("list", list);
				map.addAttribute("mapCats", mapCats);
			map.addAttribute("pageTitle", "SPORTY SHOES - HOMEPAGE");
		return "index";
	  }

}
```

MemberController.java

```
package com.MyFirstSpringBootProj.sportyshoes.controllers;

import javax.servlet.http.HttpSession;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;

import com.MyFirstSpringBootProj.sportyshoes.models.Users;
import com.MyFirstSpringBootProj.sportyshoes.services.UserService;

import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;

@Api(value="Member Controller For Sporty SHoes Application")
@Controller
public class MemberController {

	@Autowired
	private UserService userService;
//=============================================================================
	@ApiOperation(value = "Redirect To member Login Page")
	@GetMapping("/login")
	public String login(ModelMap map) {

		map.addAttribute("pageTitle", "SPORTY SHOES - MEMBER LOGIN");
```

```
		return "login";


	}
//=============================================================================
	@ApiOperation(value = "Redirect To member Registration Page")
	@GetMapping("/signup")
	public String signup(ModelMap map) {
		map.addAttribute("pageTitle", "SPORTY SHOES - MEMBER REGISTRATION");
		return "register";
	}
//=============================================================================
	@ApiOperation(value = "User Logged out from the Application & will  redirect to Home Page")
	@GetMapping("/logout")
	public String logout(javax.servlet.http.HttpServletRequest request) {
		HttpSession session = request.getSession();
		session.invalidate();


		return "redirect:home";
	}
//=============================================================================
	@ApiOperation(value = "User Logged out from the Application & will  redirect to Home Page")
	@PostMapping("/signup")
	public String signupAction(ModelMap map, javax.servlet.http.HttpServletRequest request,
				@RequestParam(value = "email_id", required = true) String emailId,
				@RequestParam(value = "pwd", required = true) String pwd,
				@RequestParam(value = "pwd2", required = true) String pwd2,
				@RequestParam(value = "fname", required = true) String fname,
				@RequestParam(value = "lname", required = true) String lname,
				@RequestParam(value = "age", required = true) String age,
				@RequestParam(value = "address", required = true) String address) {


		if (emailId == null || emailId.equals("")) {
```

```
            map.addAttribute("error", "Email id is required.");
            return "register";
}

if (pwd == null || pwd2 == null || pwd.equals("") || pwd2.equals("")) {
            map.addAttribute("error", "Error , Incomplete passwords submitted.");
            return "register";
}

if (!pwd.equals(pwd2)) {
            map.addAttribute("error", "Error , Passwords do not match.");
            return "register";
}

if (fname == null || fname.equals("")) {
            map.addAttribute("error", "First name is required.");
            return "register";
}

if (lname == null || lname.equals("")) {
            map.addAttribute("error", "Last name is required.");
            return "register";
}
if (age == null || age.equals("")) {
            age = "0";
}

Users user = userService.getUserByEmailId(emailId);
if (user != null) {
            map.addAttribute("error", "This email id already exists.");
            return "register";
}
```

```
		user = new Users();


		user.setEmailId(emailId);
		user.setFname(fname);
		user.setLname(lname);
		user.setAge(Integer.parseInt(age));
		user.setAddress(address);
		user.setPwd(pwd);


		Users user1=userService.addUsers(user);
		map.addAttribute("addedUser", user1.getFname()+" "+user1.getLname());
		return "register-confirm";
	}
//=============================================================================================
	@ApiOperation(value = "On succussfull Login user will redirect to \"Dashboard\"  or will redirect to Login page")
	@PostMapping("/login")
	public String loginAction(ModelMap map, @RequestParam(value = "email_id", required = true) String emailId,
			@RequestParam(value = "pwd", required = true) String pwd, javax.servlet.http.HttpServletRequest request) {


		Users user = userService.getUserByEmailId(emailId);


		if (user == null) {
			map.addAttribute("error", "Login failed");
			return "login";
		} else {
			if (!(user.getEmailId().equals(emailId) && user.getPwd().equals(pwd))) {
				map.addAttribute("error", "Username and Password are Incorrect");
				return "login";
			}
```

```
                }
                 HttpSession session = request.getSession();
                 session.setAttribute("user_id", user.getID());
                map.addAttribute("user", user);

                return "redirect:dashboard";
        }

//=============================================================================================
        @ApiOperation(value = " user will redirect to Edit page to edit his/her profile.")
        @GetMapping("/editprofile")
        public String editProfile(ModelMap map, javax.servlet.http.HttpServletRequest request) {
                HttpSession session = request.getSession();
                if (session.getAttribute("user_id") == null) {
                        return "login";
                }
                Users user = userService.getUserById( (long) session.getAttribute("user_id"));

                map.addAttribute("pageTitle", "SPORTY SHOES - MEMBER EDIT PROFILE");
                map.addAttribute("user", user);

                return "edit-profile";
        }
//=============================================================================================
        @ApiOperation(value = "user can edit his/her profile in this edit page")
         @PostMapping("/editprofile")
           public String editProfileAction(ModelMap map,
                        javax.servlet.http.HttpServletRequest request,
                        @RequestParam(value="user_id", required=true) String userId,
                         @RequestParam(value="pwd", required=true) String pwd,
                         @RequestParam(value="pwd2", required=true) String pwd2,
```

```
            @RequestParam(value="fname", required=true) String fname,
            @RequestParam(value="lname", required=true) String lname,
            @RequestParam(value="age", required=true) String age,
            @RequestParam(value="address", required=true) String address)
{

        HttpSession session = request.getSession();
        if (session.getAttribute("user_id") == null) {
                return "login";
        }

        Users user = userService.getUserById((Long) session.getAttribute("user_id"));
        map.addAttribute("user", user);

        if (pwd == null || pwd2 == null || pwd.equals("") || pwd2.equals("")) {
                map.addAttribute("error", "Error , Incomplete passwords submitted.");
                return "edit-profile";
        }

        if (!pwd.equals(pwd2)) {
                map.addAttribute("error", "Error , Passwords do not match.");
                return "edit-profile";
        }

        if (fname == null || fname.equals("")) {
                map.addAttribute("error", "First name is required.");
                return "edit-profile";
        }

        if (lname == null || lname.equals("")) {
                map.addAttribute("error", "Last name is required.");
                return "edit-profile";
```

```
                }
                if (age == null || age.equals("")) {
                        age = "0";
                }
                user.setFname(fname);
                user.setLname(lname);
                user.setAge(Integer.parseInt(age));
                user.setAddress(address);
                user.setPwd(pwd);

                userService.updateUser(user);

            return "dashboard";
        }

}
```

PurchaseController.java

```
package com.MyFirstSpringBootProj.sportyshoes.controllers;



import java.math.BigDecimal;
import java.util.HashMap;
import java.util.List;

import javax.servlet.http.HttpSession;

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

import org.springframework.ui.ModelMap;
```

```
import org.springframework.web.bind.annotation.GetMapping;



import com.MyFirstSpringBootProj.sportyshoes.models.Product;
import com.MyFirstSpringBootProj.sportyshoes.models.Purchase;
import com.MyFirstSpringBootProj.sportyshoes.models.PurchaseItem;
import com.MyFirstSpringBootProj.sportyshoes.services.ProductService;
import com.MyFirstSpringBootProj.sportyshoes.services.PurchaseItemService;
import com.MyFirstSpringBootProj.sportyshoes.services.PurchaseService;

import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;



@Api(value="Purshase Controller For Sporty Shoes Application")
@Controller
public class PurchaseController {

	@Autowired
	private ProductService productService;

	@Autowired
	private PurchaseService purchaseService;

	@Autowired
	private PurchaseItemService purchaseItemService;



	@ApiOperation(value = "User will be redirected to Purchase Page if he/she already log in to Application
or else  will redirect to member login page")
	  @GetMapping(value = "/memberpurchases")
	    public String memberpurchases(ModelMap map, javax.servlet.http.HttpServletRequest request)
```

```
{
        HttpSession session = request.getSession();
        if (session.getAttribute("user_id") == null) {
                return "login";
        }
        long userId = (Long) session.getAttribute("user_id");

        List<Purchase> list = purchaseService.getAllItemsByUserId(userId);

        BigDecimal total = new BigDecimal(0.0);
        // map purchase items to each purchase for display
        HashMap<Long, String> mapItems = new HashMap<Long, String>();

        for(Purchase purchase: list) {
                total = total.add(purchase.getTotal());

                List<PurchaseItem> itemList =
purchaseItemService.getAllItemsByPurchaseId(purchase.getID());
                StringBuilder sb = new StringBuilder("");
                for(PurchaseItem item: itemList) {
                        Product product = productService.getProductById(item.getProductId());
                        if (product != null) {
                                sb.append(product.getName() + ", " +
                                                item.getQty() + " units @" + item.getRate() + " = "
                                                + item.getPrice() + "<br>");
                        }
                }
                mapItems.put(purchase.getID(), sb.toString());
        }
        map.addAttribute("totalAmount", total);
        map.addAttribute("list", list);
        map.addAttribute("mapItems", mapItems);
```

```
                map.addAttribute("pageTitle", "SPORTY SHOES - YOUR ORDERS");

                return "purchases";
            }
        }
```

Repositories

AdminRepository.java

```
package com.MyFirstSpringBootProj.sportyshoes.repositories;

import org.springframework.stereotype.Repository;

import com.MyFirstSpringBootProj.sportyshoes.models.Admin;

import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Component;

@Repository
@Component
public interface AdminRepository extends CrudRepository<Admin, Long> {

    Admin findByAdminId(String adminId);

}
```

CategoryRepository.java

```
package com.MyFirstSpringBootProj.sportyshoes.repositories;
```

```
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Component;
import org.springframework.stereotype.Repository;

import com.MyFirstSpringBootProj.sportyshoes.models.Category;

@Repository
@Component
public interface CategoryRepository extends JpaRepository<Category, Long> {



}
```

ProductRepository.java

```
package com.MyFirstSpringBootProj.sportyshoes.repositories;

import org.springframework.stereotype.Repository;

import com.MyFirstSpringBootProj.sportyshoes.models.Product;

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

@Repository
@Component
public interface ProductRepository extends JpaRepository<Product, Long> {



}
```

PurchaseItemReporsitory.java

```
package com.MyFirstSpringBootProj.sportyshoes.repositories;



import org.springframework.stereotype.Repository;



import com.MyFirstSpringBootProj.sportyshoes.models.PurchaseItem;

import java.util.List;



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

import org.springframework.stereotype.Component;







@Repository

@Component

public interface PurchaseItemReporsitory extends JpaRepository<PurchaseItem, Long>{



	void deleteAllItemsByPurchaseId(long purchaseId);



	List<PurchaseItem> findAllByID(long purchaseId);

}
```

PurchaseRepository.java

```
package com.MyFirstSpringBootProj.sportyshoes.repositories;

import java.util.List;

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

import org.springframework.stereotype.Component;

import com.MyFirstSpringBootProj.sportyshoes.models.Purchase;

@Repository
@Component
public interface PurchaseRepository extends JpaRepository<Purchase, Long> {

	List<Purchase> getAllItemsByUserId(long userId);

}
```

UserRepository.java

```
package com.MyFirstSpringBootProj.sportyshoes.repositories;


import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Repository;
import org.springframework.stereotype.Component;

import com.MyFirstSpringBootProj.sportyshoes.models.Users;

@Repository
@Component
public interface UserRepository extends CrudRepository <Users, Long> {

	Users findByEmailId(String emailId);

}
```