Sporty Shoes booking engine application

Sporty shoes | https://github.com/malinimurthy123/sportyShoes/tree/master

sporty shoes booking

Malini Murthy

2021

1. Github link:

https://github.com/malinimurthy123/sportyShoes/tree/master

**Project objective:**

As a Full Stack Developer, design and develop Sporty Shoes booking POC named as SportyShoes. Use the GitHub repository to manage the project artifacts.

As a Full Stack Developer, complete the features of the application by planning the development and pushing the source code to the GitHub repository.

**Background of the problem statement:**

SportyShoes is a shoes-booking POC that lets people create users, list shoes products, categories and purchase the shoes on their website.

 Sporty Shoes is a company that manufactures and sells sports shoes. They have a walk-in store, and now, they wish to launch their ecommerce portal sportyshoes.com.

You’re asked to develop a prototype of the application. It will be then presented to the relevant stakeholders for budget approval. Your manager has set up a meeting where you’re asked to do the following:

● Presenting the specification document which has the product’s capabilities, appearance, and user interactions  
● Setting up Git and GitHub account to store and track your enhancements of the prototype   
● Explaining the Java concepts used in the project   
● Discussing the generic features of the product:  
● There will be an admin to manage the website. An administrator login will be required to access the admin page.

**The admin should be able to change his password if he wants, he should be able to:**

● Manage the products in the store including categorizing them  
● Browse the list of users who have signed up and be able to search users  
● See purchase reports filtered by date and category

**Software Requirements**

● IntelliJ: An IDE to code for the application   
● Java: A programming language to develop the web pages, databases, and others  
● SQL: To create tables for admin, categories, and other specifics  
● Maven: To create a web-enabled Maven project  
● Git: To connect and push files from the local system to GitHub   
● GitHub: To store the application code and track its versions   
● Scrum: jira   
● Search and Sort techniques: Data structures used for the project   
● Specification document: Any open-source document or Google Docs

|  |  |
| --- | --- |
| IntelliJ |  |
| Java |  |
| Maven |  |
| Git |  |
| GitHub |  |
| Scrum | JiRA |
| Search and Sort techniques | List, arrays |
| Technologies | SpringBoot  JPA  Hibernate  H2 DataBase  SpringBoot Security |

2. Dependencies in Maven POM.XML: SportyFinalProject

* Java 11(I have used Java 11)
* spring-boot-starter-data-jpa
* spring-boot-starter-web
* h2
* lombok
* spring-boot-starter-tomcat
* springfox-swagger2
* springfox-swagger-ui
* spring-boot-starter-security
* spring-boot-maven-plugin
* maven-war-plugin

**https://github.com/malinimurthy123/sportyShoes/tree/master**

**3 . Flow chart User**

**3.1 User Flow Chart**

***.***

Start

HomePage is displayed for any user with login option on the same page.

**­­­**

NO

Register First

IF

Registered User

Login page

The User is taken to homepage with add to cart option.

YES

Further pages with options like checkout, payment gateway and payment confirmation.

END

**3.1 Admin Flow Chart**

Start

LOGIN

The Admin is taken to homepage with options like:

**­­**

END

Setup Products

Setup Product Categories

Change Password

Purchase Report

Browse Members

Admin can add products form here

Admin can add categories form here

Users report can be seen here

Registered users can be seen here

Admin can change password from here and redirect to dashboard

**4. Sprint planning**

**4.1 This section tells how the workload was sliced into smaller part and was completed.**

**For effective pace and efficiency a trello board was managed as shown below.**

**It was divided into three parts**

* 1. **Task to do.**
  2. **In Progress**
  3. **Completed in Sprint 1**

**The timebox for each sprint i.e sprint 1 and sprint 2 was 1 week .**

**Sprint 1**

**The sprint 1 overall working flow of the application was done.**

**Number and duration of sprints required –**

**Total Sprints - 3**

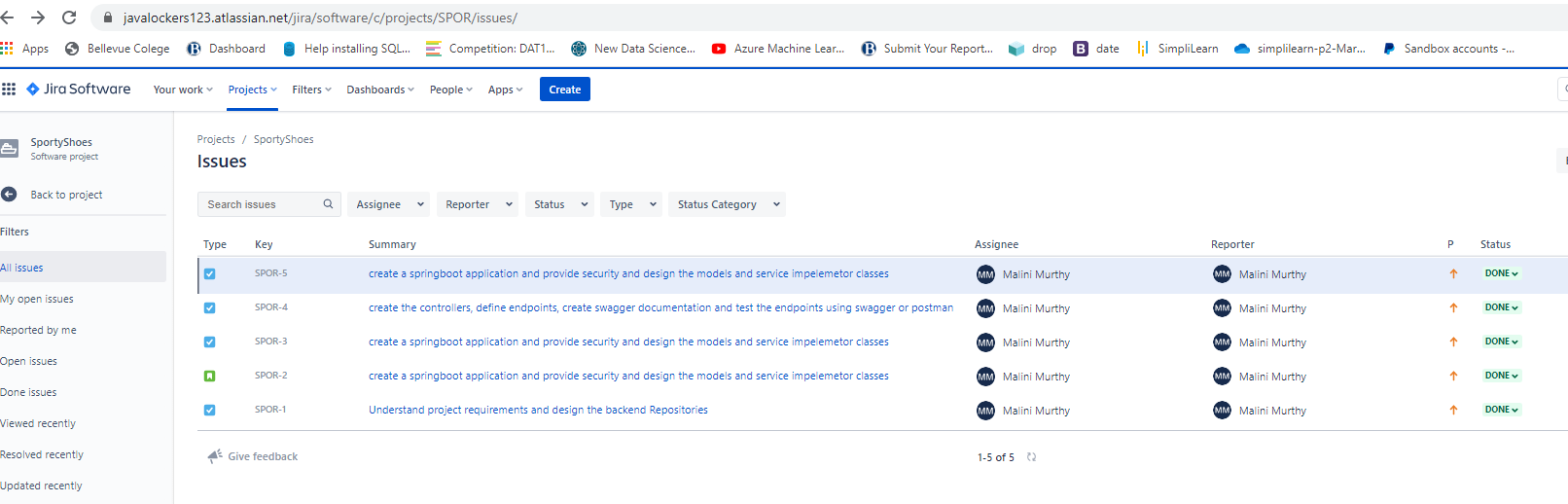
Sprint 1- Week 1

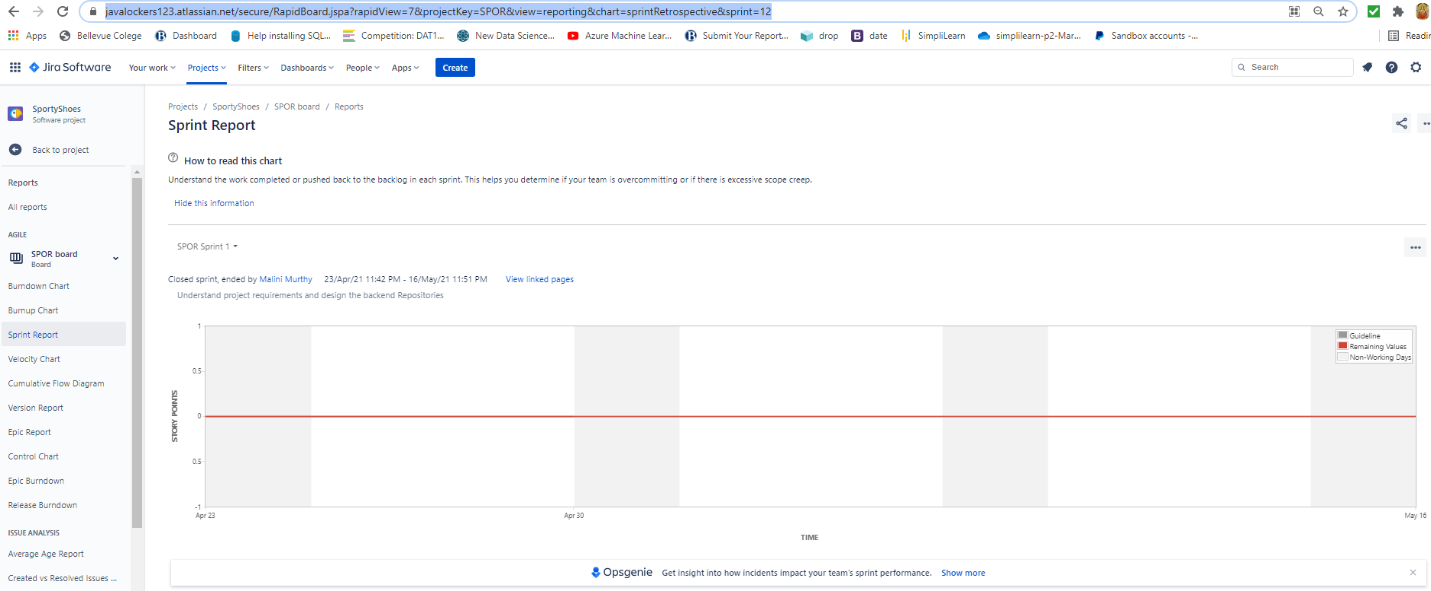
Sprint 2- Week 2

Sprint 3- Week 3

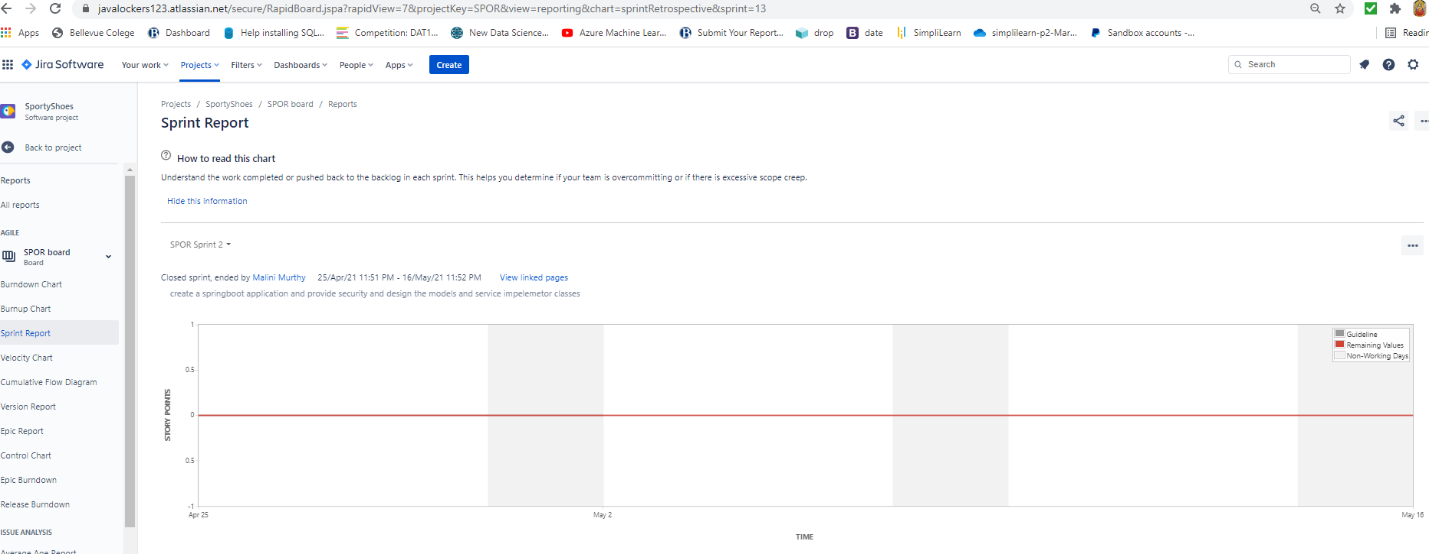
**Sprint 1-** Understand project requirements and design the backend Repositories

[**https://javalockers123.atlassian.net/secure/RapidBoard.jspa?rapidView=7&projectKey=SPOR&view=reporting&chart=sprintRetrospective&sprint=12**](https://javalockers123.atlassian.net/secure/RapidBoard.jspa?rapidView=7&projectKey=SPOR&view=reporting&chart=sprintRetrospective&sprint=12)

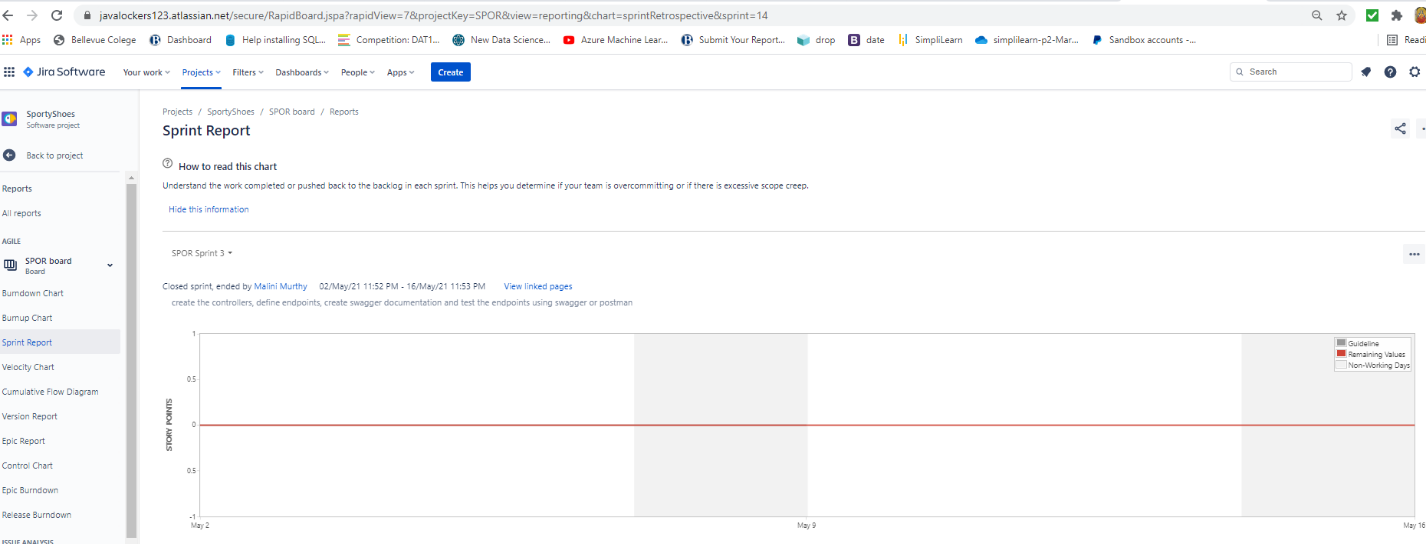
****

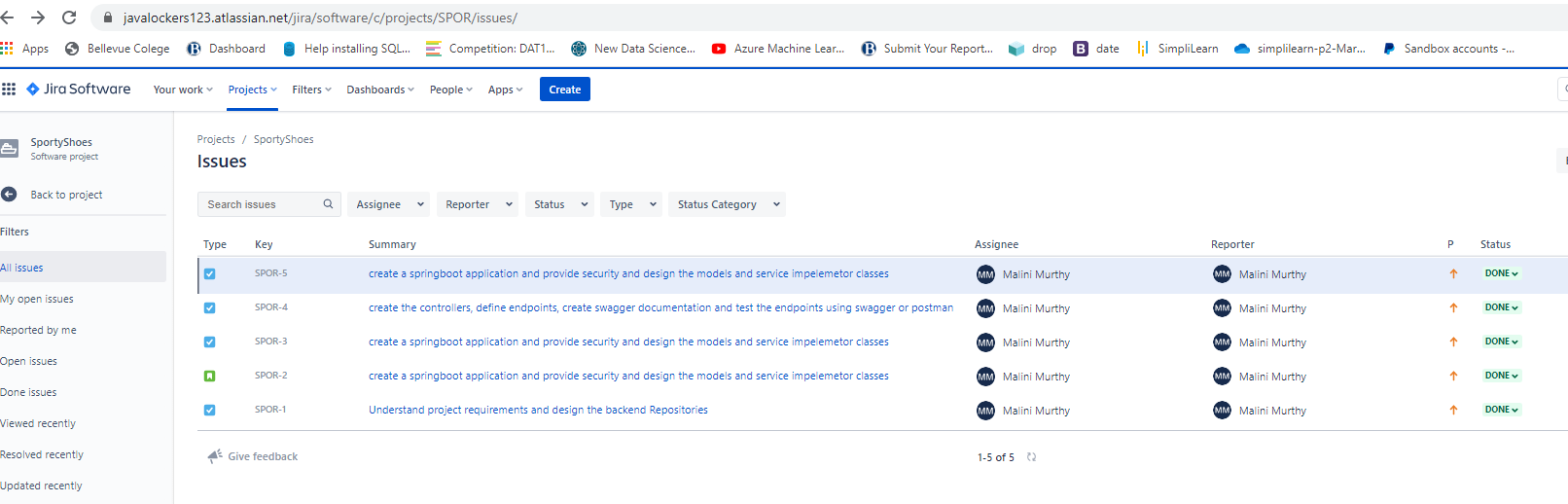
****

**Sprint 2 -** create a springboot application and provide security and design the models and service impelemetor classes

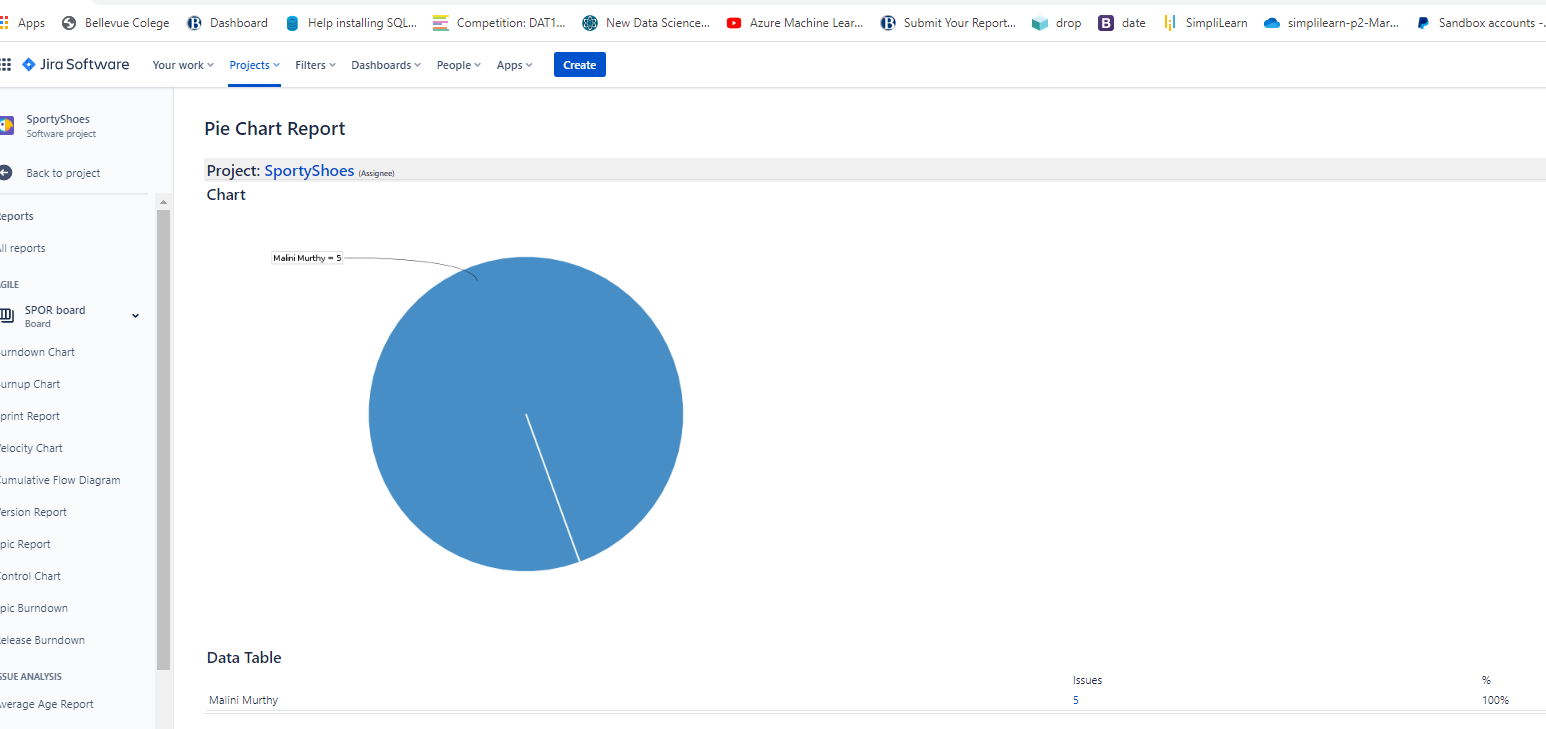


Sprint 3- create the controllers, define endpoints, create swagger documentation and test the endpoints using swagger or postman









**5 Solution Overview**

**Platform requirement**

**The platform used for this project is the InteliJ IDE, H2 DataBase, SpringBoot,**

**Since the project of this phase is going to be an input for next phase4, the project in this phase 3 involves no UI interface but only backend development with endpoints being tested using postman and swagger documentation. Please refer the screen shots in the screenshots document**

**Endpoints Developed for the SportyShoes**

1. Application is provided Security using WebSecurityConfigurerAdapter of the springboot

Following imports are needed to provide the security for the application:

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.core.userdetails.UserDetailsService;  
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;  
import org.springframework.security.crypto.password.PasswordEncoder;

The models are specified under the models package

Admin

Cartitem

Category

Products

Purchaseitem

Users

The repository has below repositories that extends JpaRepository

Adminrepository

Cartitemrepository

Categoryrepository

Productsrepository

Purchaseitemrepository

Users

The Service package has implemented classes. These classes have methods that perform the CRUD(create, read, update, delete) and Search Operations and these classes implement their respective interfaces

AdminCrudController

CategoryCrudServiceImpl

ProductCrudServiceImpl

PurchaseItemServiceImpl

UserCrudServiceImpl

**The Endpoints are defined in the classes under the controller package**

**AdminCrudController**

1. @PostMapping("/admin") – Create Admin
2. @GetMapping("/admin/{id}") – Search Admin
3. @PutMapping("/admin") – Update Password

**ProductCrudController**

1. @PostMapping("/product") – Create Product
2. @PutMapping("/product") – Update Product
3. @GetMapping("/product/{id}") – Search Product by Category and Date
4. @DeleteMapping("/product/{id}")- Search Product by ProductID
5. @GetMapping("/allproducts")- Search all Products

**PurchaseItemCrudController**

1. @PostMapping("/purchaseitem") – Purchase an item, which is the sporty shoes
2. @GetMapping("/allpurchases") – Search all Purchases

**UserCrudController**

1. @PostMapping("/user") – Create User
2. @PutMapping("/user")- Update an User
3. @GetMapping("/user/{id}")- Search an User by UserId
4. @DeleteMapping("/user/{id}") – Delete User
5. @GetMapping("/allusers")- Search all Users

SportyDashBoardController

1. @GetMapping("/dashboard") – this is the dashboard welcome screen which will be an UI next phase

The Swagger Documentation has been developed and is available on the below URL:

<http://localhost:7000/swagger-ui.html#/user45crud45controller>

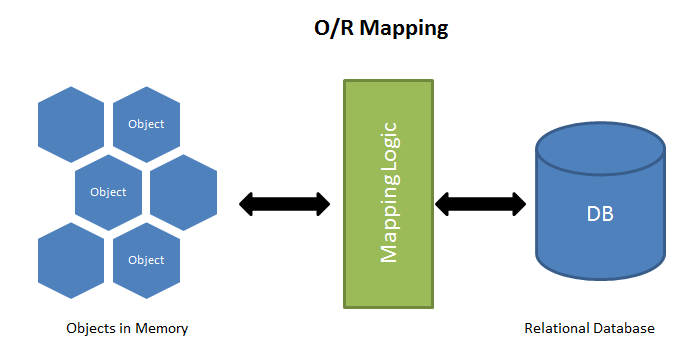
Swagger documentation is created in the class sportyshoesconfigbean

import springfox.documentation.spring.web.plugins.Docket;  
import springfox.documentation.swagger2.annotations.EnableSwagger2;

Swagger documentation is created by –

return new Docket(DocumentationType.*SWAGGER\_2*)

DataBase and Mapping:



**Time Line for development Modules that gave rise to Sporty Shoes POC required for phase 3**

1. Getting Started

Index Page

Static Components

2. My Account

My Account Page

JPA

Hibernate

MySQL

Security

New User

Login

3. Admin Portal

Entity Diagram

Login

New Shoes

View Shoes List

4. Shoes store

Shoes shelf - Product

Shoes Detail - Product

Update Shoes - Product