Documentation of flow of the project SportyShoes site :

Document and project specification-

Project name- SportyShoes

Developer name- Rahul Rana

GitHub Account name- Rahulpersie66

Github repostitry link- https://github.com/Rahulpersie66/Project--sporty-Shoes-using-springboot.git

IDE- Eclipse IDE used for creation of thisSprigBoot project with the help of SpringStarter project which behind the scene use spring.io

Tool – Maven used for download dependency

Plug-in – Maven compiler plugin and maven war (webapp archive) plugin

Dependency – MySql connector 8(mysql-connector-java), springfox-boot-starter, springboot-starter-jpa, spring-boot-devtools, spring-boot-configuration-tools

Programming language- JAVA is used

Database name- fun

Table names- user_registered, admin_table, products_sporty_shoes, ordered_booked

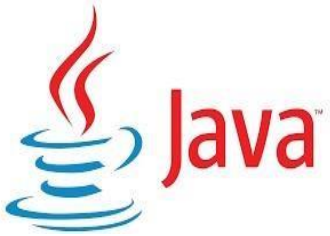
Searching/Updating/Inserting – used SpringBoot Jpa query language to add,update and search data in database

Output of the project – Swagger2 is used for display output or can use Postman to see the output

Server- Any server(because spring have inbuild server)

For dependency file- pom.xml for downloading the dependency

MySql connecting done in file- application.properties


Java


eclipse


GitHub


git


Maven™


MySQL®

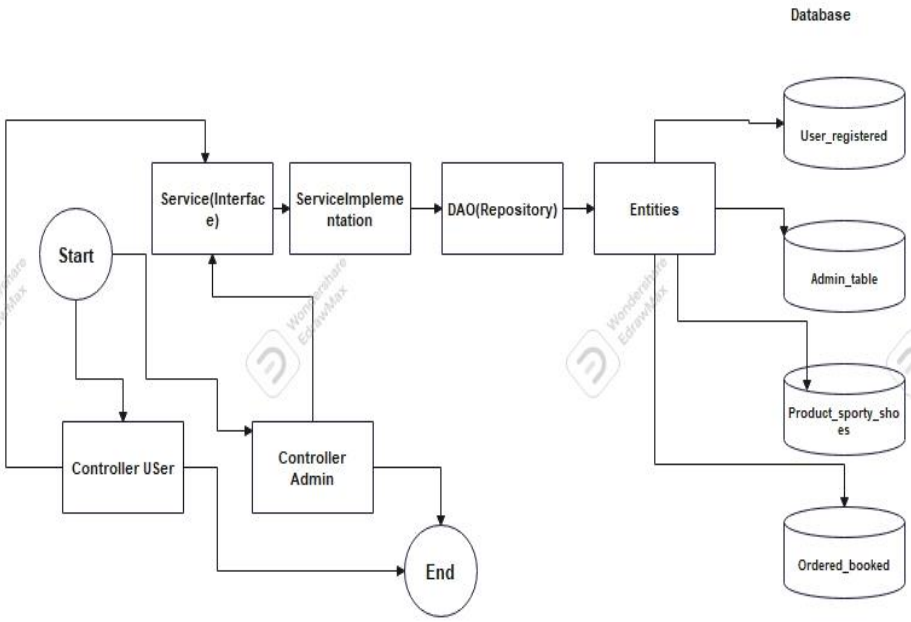

spring®


spring boot


Swagger™
Supported by SMARTBEAR


POSTMAN

Data flow diagram according to algorithm-



Algorithm -

Step -1 Create a Spring boot project using spring starter project which uses spring.io behind the scene for spring boot project creation.

Step -2 Add required dependency to **pom.xml** file and set **application.properties** file in **src/main/resource** for database configuration and swagger 2 enable.

Step -3 In the main driver class present in the pkg:**src/main/java/com.spring.boot.sportyshoes** SportyshoesApplication.java add print statement "welcome to sportyshoes" that will be print on the console.

Step -4 Now create 5 different packages for 5 different operation- table formation(entities), for crud operation(daos), for define services(service interface), for working of services(serviceImpls) and for sending commands(controllers).

Step -5 In entities define all table that are going to be stored in the database using @Entity @Table(name) and define their primary key if needed.

Step -6 Now create Repository for each table by extending repository to JPA Repository for implementing basic operation such as add, delete , findall and find one. This class will be marked as @Repository.

Step -7 Now create Services for each table and define services required by both controller admin and user. Remember to make this interface and just write each services definition only.

Step -8 In next package ServiceImpls- implementation each service in details that are required by both the controller and this class will be marked as @Service. And in this class each method of service will be overrriden and proper definition of their working will be there.

Step -9 In the last package Controller what services require by user or client will be define here alongside with their mapping and will call the respective service from serviceImpl class. This class will be marked as @RestController and mapping of url to services will be done here.

Step -10 Now run the driver class and just use swagger2 to perform various function or can use postman to perform each function at url:**http://localhost/8080/home** with mapping joined to port number:8080 (8080 is default port number).

Step by Step process from sprint planning to product release-

Sprint 1-

Sprint Planning –In this Sprint, all tables that are storing data are defined and connected to the database using spring boot application properties and repository class is define for each table to perform basic crud operation.

Objective of sprint 1-

-to construct a table in MySql using Annotation way and mark each table @Entity

-Now connect the each table Repository class by extending it to JPA Repository and provide class name of table class alongside with primary key datatype. And mark these class as @Repository.

```
@Repository
public interface AdminDao extends JpaRepository<AdminTable, Integer>
```

Process & development work-

Each table –**Admin_table, products_sporty_shoes,user_registered,ordered_booked** will be define alongside with their column name and their datatype and define setter and getter method with constructor for each field.

- It will create table with respective table name in mysql alongside their column .

Sprint retrospective-

That how to implement basic crud operation of repository in the Service interface and how to make them work according to the request by the client.

Sprint 2-

Sprint planning- In this phase of sprint 2, I build the service implementation and service interface. In service interface class define all the services that can be required by the client on web server and their definition in the service Implementation class where well explain definition of each service is defined alongside comments what each service do. And make this class @Service to help springboot to recognize it's a service classs.

Sprint Objective-

-Build interface for all 4 different table along side with service definition.

-In next class **ServiceImpl implemnets Service** which override all service methods and well define them. This class is marked as @Service.

-In serviceImpls is designed according to client request on the web server and will perform basic crud operation with the help of repository class.

Sprint Retrospective-

In this sprint 2, the project is still missing controller class which will perform or send request to perform various services on demand of the client request. So here are two actors- User and admin, so two controllers need to define perform various operation.

Process and development work-

In **service** interface simple definition of serbices are defined . e.g-

```
public interface AdminService {

	//sign in the admin and search if is admin
	public Boolean signInAdmin(String adminName,String password);

	//if admin name is correct
	public Boolean checkAdminName(String adminName,String password);

	//add admin in the table-- AdminTable
	public AdminTable addAdmin(String adminName,String password);

	//to change the password of admin using - adminname
	public Boolean changePassword(String adminName, String password);
}
```

After defining function need to create another class which extends the **Service** class and override all methods in details as required by controller. e.g-

```
@Service
public class AdminServiceImpl implements AdminService {

  @Autowired
  private AdminDao adminDao;



  //Check if Admin is valid or not
  @Override
  public Boolean signInAdmin(String adminName, String password) {
    boolean flag=false;
```

```
    Set<AdminTable> allAdmins=new LinkedHashSet<AdminTable>(adminDao.findAll());
    for(AdminTable admin:allAdmins)
      if(admin.getAdminUsername().equals(adminName) &&
admin.getPassword().equals(password))
        flag=true;
    return flag;
  }


  //check adminName is Right
  @Override
  public Boolean checkAdminName(String adminName, String password) {
    boolean flag=false;
    Set<AdminTable> allAdmins=new LinkedHashSet<AdminTable>(adminDao.findAll());
    for(AdminTable admin:allAdmins)
      if(admin.getAdminUsername().equals(adminName))
        flag=true;
    return flag;
  }


  //add the admin details in admin Table
  @Override
  public AdminTable addAdmin(String adminName, String password) {
    AdminTable admin=new AdminTable(adminName,password);
    adminDao.save(admin);
    return admin;
  }

  //to change the password using adminName
  @Override
  public Boolean changePassword(String adminName, String password) {
    boolean flag=false;
    Set<AdminTable> allAdmins=new LinkedHashSet<AdminTable>(adminDao.findAll());

    for(AdminTable admin:allAdmins)
      if(admin.getAdminUsername().equals(adminName))
      {
        AdminTable newAdmin=new
AdminTable(admin.getAdminId(),admin.getAdminUsername(),password);
        adminDao.save(newAdmin);
        flag=true;
      }

    return flag;
```

```
  }



}
```

And this class will be marked as @Service.

Sprint 3

Sprint Objective-

-Finally we will create controller class to control services which need to be called and which need to be displayed according to client requirement.

-for different request demand by client respective service is called.

-here mapping of each service is done according to client request

-if it's a display request use @GetMapping

-if it's a add request i.e signUp ,purchase product use @PostMapping

-if it's a update request i.e update productname, vendor name,edit account details use @PutMapping

-if its a delete request i.e delete a product from product table use @DeleteMapping

-all these above request require url alongside with them.

Sprint Retrospective-

In this sprint 3, use of swagger 2 enable and how to work on swagger2 api , add request body and work with various mapping such as @GetMapping @PostMapping @PutMapping and @DeleteMapping and how to work on Postman and save each request response.

Process and development work-

For signIn display string message welcome you are signed in and for signOut display message you are signed out and also if someone purchase the product how to reflect the changes of product quantity in the product table and purchased item order should be placed in ordered _Booked table.

And , mark this class as @RestController to help springboot to recognize it as controller class and then this class suppose call signIn function it will call the class like this.

```
@RestController
public class AdminController {

  @Autowired
  private AdminServiceImpl adminServiceImpl;
  @Autowired
  private UserServiceImpl userServiceImpl;
  @Autowired
  private ProductServiceImpl productServiceImpl;
  @Autowired
  private OrderServiceImpl orderServiceImpl;

    //to signIn user admin -- if admin is valid Admin
    @GetMapping("/auth/signIn/{adminname}/{password}")
    public String signInAdmin(@PathVariable String adminname,@PathVariable String
password)
    {
      if(adminServiceImpl.signInAdmin(adminname, password))
        return "You are valid ADMIN- "+adminname+" !! Welcome to Admin page.";
      else if(adminServiceImpl.checkAdminName(adminname, password))
        return "Admin your password is wrong!!";
      else
        return "Not a valid admin";
    }
  }
```

-if admin id and password is valid the msg will be printed: You are valid admin.

-if not valid admin : then not a valid admin

-if admin id is correct but password wrong then: Admin your password is wrong!!

Similarly all request of client is define in the controller class alongside their mapping with their respective url to call them.

Conclusion-

This SpringBoot project uses the swaggger2 or postman to perform various function and then reflect the same in the database. That you can seen in the Screenshots of project file.

Can enhance project by-

→Making it more interactive by using BootStrap.

→ Provide more option for user to perform as multiple item shopping.

→Providing a real payment or transaction method for purchasing the item.

→Join delivery of item option and how much time will item will be received.

→can include option pay at delivery.

➔ Its url based app so make it interactive by clicking on option you go to page for shooping.

Unique selling points-

→You can purchase item directly by going to Url .

→For admin it is easy to use as it reflects database data on the screen and directly can call each function using swagger2.

→ Until user purchase item, order is not placed and no deduction of quantity from project.