Sporty Shoes

A SpringBoot project,
Using H2 database
And Maven
So as to be
fully standalone.

1) Project Statement:

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

2) Sprint Planning:

A. Product Backlog:

i. **Plan Flow** of the Back-End, Front-End and Object Persistence.
   a. Plan server-side dependency Spring modules, DB, interfaces, JPA, CRUD, Entities, session management etc.
   b. Plan client-side dependency CSS, HTML, JSP, JS.
   c. Plant smart ease of living features.

ii. Start coding **classes** and **interfaces**.
   a. Data members for users, products, transactions, cart, etc and Data Structures required, persistence required and also start coding them.
   b. Different types Annotation mapping, session management, ModelAndViews, repository interface using CRUD/JPA, etc. for the portal
   c. Start coding different annotated HttpServlets and Maintain Entities, Data Structures, add random values for testing.
   d. Start coding in JS, JSP, HTML and CSS for client-side view.
   e. Run and Debug all the code and reduce waste codes.
   f. Test and Iterate code till a certain polish is accomplished.

iii. Run, debug, test and reiterate.
   a. Run and Debug all the code and reduce waste code, data members and structures.
   b. Test and reiterate code till a certain polish is accomplished.

iv. Test Run App.
   a. **Run** and **Test** Application.

B. Sprint Table:

Sprint No.	Tasks	Estimation	Status
1.	**Plan Flow** of the Back-End, Front-End and Object Persistence.	4 hours	Pending
2.	Start coding **classes** and **interfaces**. –- **PART I**	10 hours	Pending
3.	Start coding **classes** and **interfaces**. – **PART II**	10 hours	Pending
4.	**Run**, **debug**, **test** and **reiterate.**	4 hours	Pending
6.	**Test** Run App.	2 hours	Pending

C. Sprints:

i. **Plan Flow** of the Back-End, Front-End and Object Persistence.

a. Plan server-side dependency Spring modules, DB, interfaces, JPA, CRUD, Entities, session management etc.:

In this, we plan the dependencies, flow, components, requirements, logical issues, etc to required to make the SportShoes portal or atleast enough to start building it. Here we decided to use spring modules: SpringBoot, devtools, JPA, Hibernate, H2 database, MySQL database. We decided that even though the normal use would be H2 database to make the App truly standalone but the code for connecting to MySQL can be done if needed as it is not much different. Hibernate and JPA are required to persist objects as entities as required. Interfaces like CRUDRepository and JPARepository make things easy by providing the basic SQL queries for getting List<Object> and making the life of programmers easy. We also plan all the Entities i.e. object classes whose objects are to be persisted or stored in the database. We also require to choose one of the many session management tools. We ended chosing HttpSession as the method for maintaining session.

b. Plan client-side dependency CSS, HTML, JSP, JS.:

JS, CSS, JSP and ModelAndViews are the best tools for the job as we plan to offload some view calculation job to client using JS lowering lines of code and also making the website seem to load faster. Here we plan the JSONObjects and JSONArrays to send from the server-side and start with building view on the client-side using the above mentioned JSP, JS, CSS. Model and Views and HttpSession are used on the server-side. We must carefully think about the processing and data we send to the client some data is better of pre-processed as that is more secure than handing down sensitive data. We must also plan additional options for Admin login and make sure on the serverside that normal user cannot access the servlet. Adding more options in the visible DOM for Admin login so that they can access more data was done using JS and data sent from server-side such that even if source code was to be inspected normal user cannot access the page and will be redirected to Home Page.

c. Plant smart ease of living features.:

In this we plan for any kinds of ease of living features that could be added to make portal better. Such as adding products to cart using CartManager, making cart versatile by adding option to remove individual items from the cart, buying all items in the cart using one click. Also, to make user focus out less and enjoy the seemless experience we collect most of the personal data such as address, card-details early on during the registration process so that the Buy option just buys the item.

For admins accessing the site let them view most of the users with their personal data and purchase history along with date and time of purchases made. Also, admins have option to login as normal users so that they can browse portal with users-like experience which is actually just limiting admin to a normal user.

ii. Start coding **classes** and **interfaces**.

a. Data members for users, products, transactions, cart, etc and Data Structures required, persistence required and also start coding them.:

We need to start coding with all the data members, data structures, persistence of an object, classes in mind. Users, Products, Transactions and Carts are the classes which need to be managed using data structures, Hibernate and JPA entities, etc. We decided to add Users and Products to SQL database while using a java data structure to maintain transactions and user's product carts in memory using userIDs and productIDs to map them. This saves on data as Users and Products are mapped rather than hard coding all. i.e. loose coupling is achieved hence, the preference for Spring.

b. Different types Annotation mapping, session management, ModelAndViews, repository interface using CRUD/JPA, etc. for the portal:

In this, we make all the different mappings to functions of different servlets using annotations. We use ModelAndViews, CRUD/JPA repository interfaces, JSONArrays and JSONObjects, HashMaps, ArrayLists and iterator interface to inter play with data and map user to different servlet pages as annotated. CRUD/JPA methods are configured so as to save, update and find entities for Login and Registration. If credentials entered by a client matches of that in repository HttpSession is used to track the user as they move through different sections of the portal. The session is managed unitl they log out by accessing LogOut section.

c. Start coding different annotated HttpServlets and Maintain Entities, Data Structures, add random values for testing.

In this, we start with the rest of the HttpServlet functions that provide ViewAndModel. Here we think carefully about the data sent to the user not being sensitive. We use the aforementioned JSONArrays and JSONObjects and use HashMap to send the JSON objects to client and the client can access them using ${objname} notation with objname being the same as the one configured on the server-side. This also means that changing JSONArray can break the output hence strategically using JSONObjects in their stead, we reduce the risk. We also add Random values to the SQL database so that we can start with the testing of the project as we proceed so we don't waste our time writing false code. We must also refine all the functions that navigate user through the portal. We also decided to maintain user carts and user transactions in java collections which are not backed up in the DB.

d. Start coding in JS, JSP, HTML and CSS for client-side view:

Now we have to code for the client-side view using JSP, CSS and JS majorly. We use JS to Reduce server-side load and also html code. Making the experience feel lightning fast as less data is sent for the same view. We use JS to also make options, popups, disclaimers, etc. show up as needed using server-side HashMap<String, JSONObject> or HashMap<String, JSONOArray> to send and JS innerHTML properties for controlling HTML tags so that view can be calculated rather than typed however clients that don't allow running of .js scripts will face view not working. A CSS is made with all the pages in mind and is linked to all the JSPs. i.e. it is plug and play. We could have made different CSS for different pages, but once CSS is downloaded it can be used throughout the portal rather than getting different CSS for different Pages. Hence, saving on client data sent as well as client data required.

e. Run and Debug all the code and reduce waste codes:

In this session, we are almost completely done with the major coding part of the SpringBoot application and we only need to minorly adjust the code to make the functioning of the app smooth. We also actively look for any redundant code and make sure we move data we always calculate and mark it with some reference and keep it in memory. This helps save server-side resources and we don't keep calculate things we keep needing again and again. We might also have skipped over some bugs such as a certain one we faced where adding products to cart functioned and so did removing the products would work fine too; on the surface that is. CartMaker which uses data manipulated by CartAdder and CartRemover to send JSON objects and arrays to load the view on client-side using JS would have its functionality to add products broken when products were removed and hence the view on the client-side would not load. The object was unusable by CartAdder after being used by CartRemover due to using String.split() to calculate the document view. It was solved by reiterating the method to not break functionality after being used over and over again by CartAdder and CartRemover than being accessed by CartMaker to build client-side view objects.

f. Test and Iterate code till a certain polish is accomplished:

Testing makes sure that none of the bugs make it out to the finished version of the application hence is very important step. The testing here is done manually which is still better than doing none. All of the outlying border conditions that could be thought of by me and my team including and consisting entirely of me set out to test the application on these border conditions and any inconsistency was ironed out before finalizing the product publishing.

Sprint No.	Tasks	Elapsed	Status
1.	**Plan Flow** of the Back-End, Front-End and Object Persistence.	3 hours	Done
2.	Start coding **classes** and **interfaces**. –- **PART I**	11 hours	Done
3.	Start coding **classes** and **interfaces**. – **PART II**	13 hours	Done
4.	**Run**, **debug**, **test** and **reiterate.**	5 hours	Done
6.	**Test** Run App.	2 hours	Done

3) Working of the App:

4)