HWA Project: Supermarket Inventory System

By George Ryalls

My Approach

- Reviewed the specification and broke it down into the related sections to determine what the MVP was.
- Created my Jira Board with the relevant epics and user stories for me to get the minimum viable product(MVP).
- Fully focused on creating the MVP before adding anything extra as new features could be added to the product retrospectively.
- Made sure I had a proper understanding of the technologies used.

Sprint Plan

- Using my initial ERD I could determine what the structure of my project would be.
- From that I could develop my backlog and set out the tasks into a sprint.
- My backlog was broken down into 4 epics; Front-end, back-end, testing and documentation.
- These epics included the relevant user stories and tasks that needed to be completed in order to fulfill them.
- All items from my backlog were added to the sprint as they were all required to meet the MVP.

Technologies

Spring:

- Spring is an application framework and inversion of control container for Java. There are numerous different modules of Spring and for this project I used Spring data, web and boot. It allows the use of beans, which are a managed object, meaning the creation, management and destruction of the object is controlled by Spring hence why Spring is known as an inversion of control container.

HTML:

- HTML is the standard markup language for documents to be displayed in a browser. It is usually used in conjunction with CSS and JavaScript to get the most out of it. It is used to determine the structure and general contents of the webpage It works off a tag-based system that are read and interpret by a browser.

CSS

- CSS is used to determine how a webpage looks and is styled. It can be used inline within the HTML document to style element, or through an external style sheet which is linked to the HTML page. Doing it through an external sheet allows you to have standard styling across your different HTML pages. Another way to implement CSS is through a framework such as bootstrap which allows you to import it into your HTML document and use its pre-determined styles for elements such as buttons.

JavaScript:

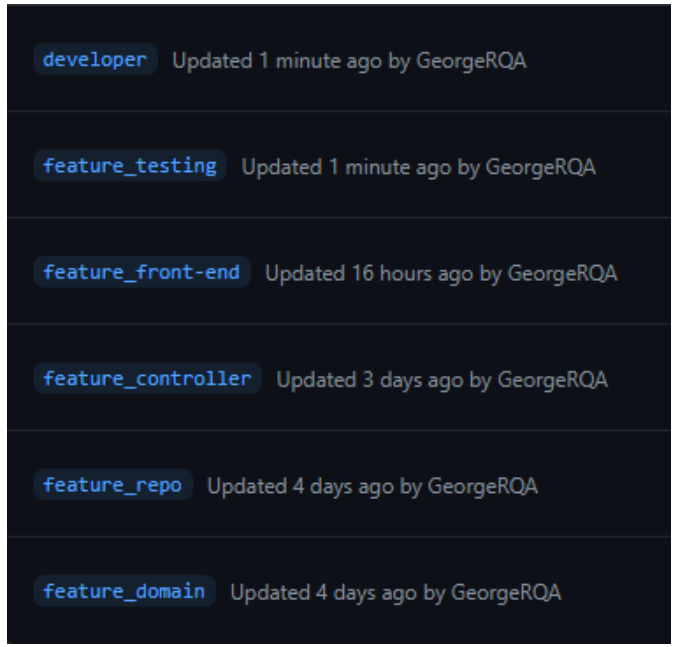
- JavaScript is a scripting language that can give webpages functionality. These can include things such as search bars, buttons and other functionality you would commonly see on a website.

Selenium

- Selenium is a tool used to automate web browsers and can be used in order to test browsers. It is a Web driver-based testing medium which is a piece of software that manages communication between your code and browser. The web driver used in this project was for Chrome, so all the testing was done in a chrome browser. Selenium allows you to select certain elements from a browser such as a button through its ID and then assert whether clicking that button has the expected outcome, such as taking you to a new webpage.

Version Control

- Version control was managed using Git and GitHub.
- A structure of main > developer > feature_(feature name) was used in order to ensure only working code was pushed to the main branch.



Testing

- Achieved an overall coverage of 50.9% in src/main/java so far.
- The classes tested were the Service and controller classes.
- The coverage can be increased considerably by testing the update, read and delete methods in the service and controller.

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
HWAProject	59.0 %	193	134	327
src/main/java	50.9 %	88	85	173
src/test/java	68.2 %	105	49	154

Test Examples

Service Create Test

```
17
18  @SpringBootTest
19  public class ServiceTest {
20
21      @Autowired
22      private SupermarketService service;
23
24      @MockBean
25      private SupermarketRepo repo;
26
27      @Test
28      void testCreate() {
29          Supermarket apple = new Supermarket("Apple", 0.49, 10);
30          Supermarket addedApple = new Supermarket(1L, "Apple", 0.49, 10);
31
32          Mockito.when(this.repo.save(apple)).thenReturn(addedApple);
33
34          assertThat(this.service.create(apple)).isEqualTo(addedApple);
35
36          Mockito.verify(this.repo, Mockito.times(1)).save(apple);
37      }
```

Controller Create Test

```
@Test
void testCreate() throws Exception {

    Supermarket apple = new Supermarket("apple", 0.99, 10);

    String appleAsJSON = this.mapper.writeValueAsString(apple);

    RequestBuilder mockRequest = post("/create").contentType(MediaType.APPLICATION_JSON).content(appleAsJSON);

    Supermarket savedApple = new Supermarket(2L, "apple", 0.99, 10);

    String savedAppleAsJSON = this.mapper.writeValueAsString(savedApple);

    ResultMatcher matchStatus = status().isCreated();

    ResultMatcher matchBody = content().json(savedAppleAsJSON);

    this.mockMVC.perform(mockRequest).andExpect(matchStatus).andExpect(matchBody);
}
```

HWA Demonstration

As a user I want to be able to add a new item to the system so that I can update my inventory.

CRUD Funtionality

55 ~~HWAP-10~~

As a user I want to be able to view all the entries in the database so that I can keep track of what products are in there.

CRUD Funtionality

55 ~~HWAP-11~~

As a user I want to be able to update an entry in the database so that I can keep the information up to date.

CRUD Funtionality

55 ~~HWAP-12~~

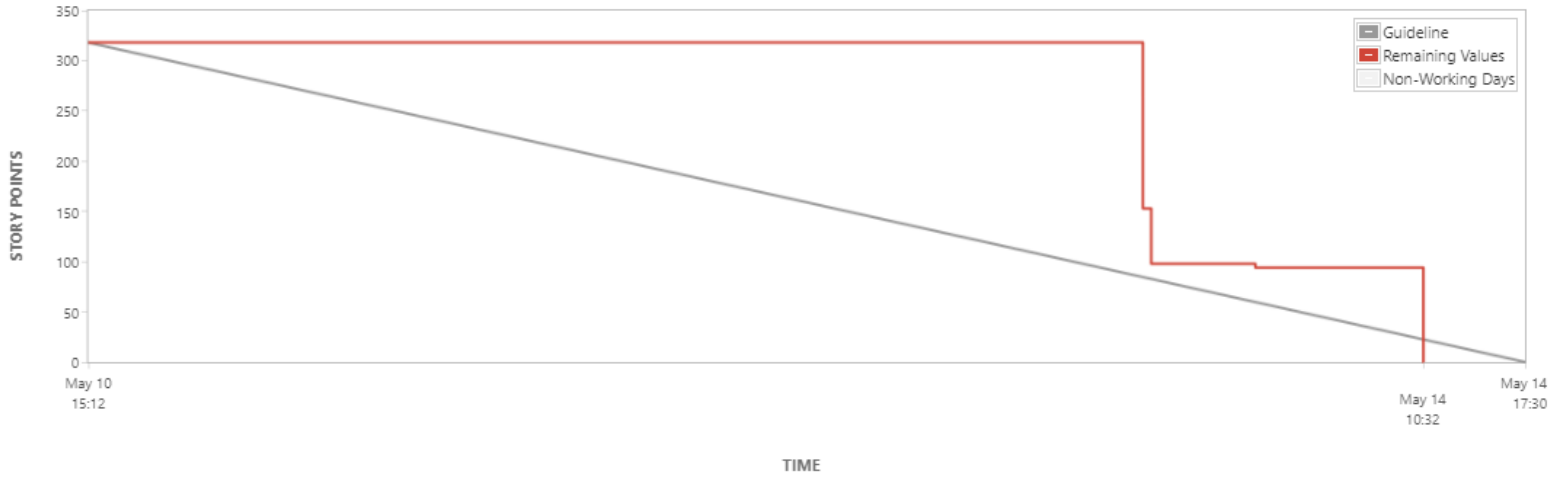
As a user I want to be able to delete an entry in the database so that I can remove ones that are no longer required.

CRUD Funtionality

55 ~~HWAP-13~~

Sprint Review

- As can be seen by my burndown chart no issues were left uncompleted. This sprint went better than my previous one as I managed to successfully add all the user stories prior to starting the sprint as opposed to retrospectively adding them. Additionally this time I was a lot better at moving topics into in progress and then over to complete which meant the sprint wasn't all completed in one steep jump.



Sprint Retrospective

- Completed all user stories on the sprint so nothing got returned to the backlog.
- Effectively split the relevant user stories up into different epics as well as assigning them tasks to make them easier to manage.
- Was able to use it as an overview of what needed to be completed for the project.
- I think although it was improved on, I still need to be more active in how I assign tasks as in progress or complete as I was occasionally still moving things a while after they were done as opposed to the moment they were complete.

Conclusion

- Overall, I believe I managed to deliver the MVP.
- I think there's room for improvement to be made with regards to my project management that would allow me to complete task more effectively.
- I would have liked to be able to add a modal to update items instead of being taken to a new page however, I didn't have the time to implement it and focused on delivering the MVP.
- There is also considerable room for my testing to be expanded in order to push the coverage closer to 80% in addition to expanding my Selenium coverage.