POKÉMON WEB APPLICATION

Philip Sephton

INTRODUCTION

I am Philip Sephton, who is currently training with QA consulting on their software specialist course.

When I first looked at the specification I was unsure what I should prioritise so I made a separate document which included all the tasks that I needed to complete to get a minimum viable project. This acted as a checklist I can refer to while completing the assessment.

After identifying the minimum viable project I was the able to create a template for what my page would look like, as well as complete my Jira boards which used in conjunction with my document to prioritise tasks that needed completing.

CONSULTANT JOURNEY

During this project I have had to learn new technologies to help me implement my design.

These technologies where Html, CSS, JavaScript and Spring.

Among these technologies I had the most difficulty understanding and applying JavaScript, mostly because I was struggling understanding how to put what I had imagined in my head into practice, mostly having a button for every entry to update or delete.

To overcome this I spent many hours practicing JavaScript online using the online resource edabit, as well as the mobile resource Codecademy Go, while practicing JavaScript I realised that I could call html instructions inside Javascript which would allow me to create these buttons.

CI

Applying version control was fairly straightforward throughout the project, during my previous project I didn’t implement the version control as well as I would of liked, this helped me during this project as it allowed me to remember what I did wrong and how to improve going forward.

During the project I created a dev branch which when I added new features and bug fixes, I made in their own feature or bug fix branch, which then got pushed up to the dev branch when the where completed.

During the project I did also push straight onto my dev branch, this was to update the readme when I found formatting issue, although this isn’t best practice I did it to save time, if I was updating code I wouldn’t push straight up to the dev branch as it could break y application.

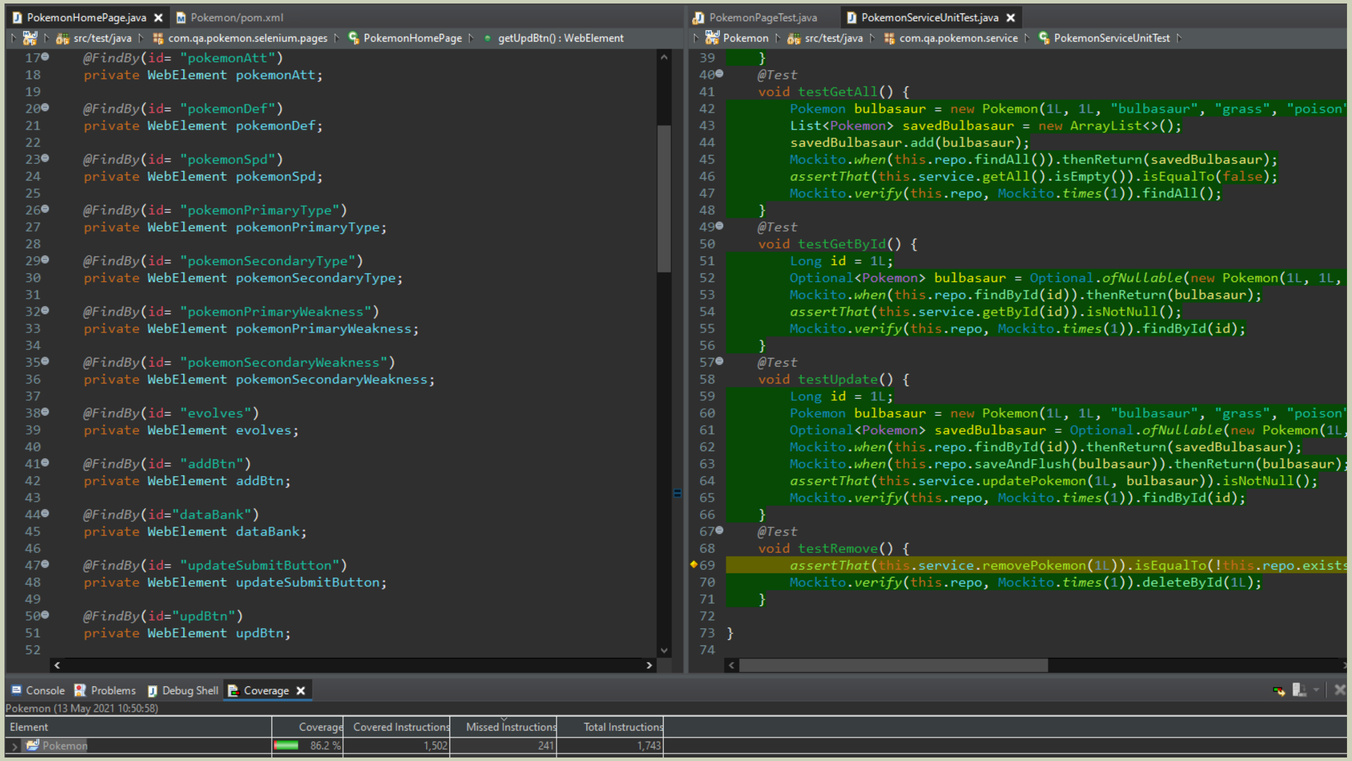
TESTING

During my project I tried to test as much of my project as I could to get a coverage of 80% which was required in the product specification.

During my attempt writing testing code for my front end I was successful writing code to cover my controller and service, during writing selenium tests for my front end, when writing the tests I realised that the tests ran faster than my page could load causing them to fail most of the time.

To overcome this I had to explicitly wait the tests until all of the desired elements had loaded into the webpage.

TESTING CONT



PokemonHomePage.java | Pokemon/pom.xml

src/test/java > com.qa.pokemon.selenium.pages > PokemonHomePage > getUpdBtn() : WebElement

```java
    @FindBy(id= "pokemonAtt")
    private WebElement pokemonAtt;

    @FindBy(id= "pokemonDef")
    private WebElement pokemonDef;

    @FindBy(id= "pokemonSpd")
    private WebElement pokemonSpd;

    @FindBy(id= "pokemonPrimaryType")
    private WebElement pokemonPrimaryType;

    @FindBy(id= "pokemonSecondaryType")
    private WebElement pokemonSecondaryType;

    @FindBy(id= "pokemonPrimaryWeakness")
    private WebElement pokemonPrimaryWeakness;

    @FindBy(id= "pokemonSecondaryWeakness")
    private WebElement pokemonSecondaryWeakness;

    @FindBy(id= "evolves")
    private WebElement evolves;

    @FindBy(id= "addBtn")
    private WebElement addBtn;

    @FindBy(id="dataBank")
    private WebElement dataBank;

    @FindBy(id= "updateSubmitButton")
    private WebElement updateSubmitButton;

    @FindBy(id="updBtn")
    private WebElement updBtn;

```

PokemonPageTest.java | PokemonServiceUnitTest.java

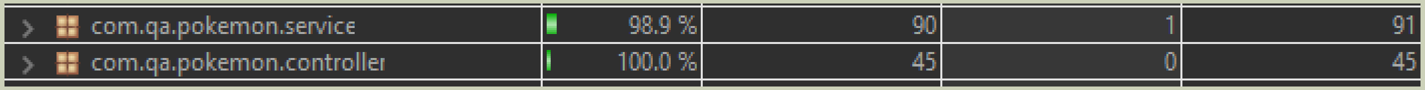
Pokemon > src/test/java > com.qa.pokemon.service > PokemonServiceUnitTest

```java
    }
    @Test
    void testGetAll() {
        Pokemon bulbasaur = new Pokemon(1L, 1L, "bulbasaur", "grass", "poison"
        List<Pokemon> savedBulbasaur = new ArrayList<>();
        savedBulbasaur.add(bulbasaur);
        Mockito.when(this.repo.findAll()).thenReturn(savedBulbasaur);
        assertThat(this.service.getAll().isEmpty()).isEqualTo(false);
        Mockito.verify(this.repo, Mockito.times(1)).findAll();
    }
    @Test
    void testGetById() {
        Long id = 1L;
        Optional<Pokemon> bulbasaur = Optional.ofNullable(new Pokemon(1L, 1L,
        Mockito.when(this.repo.findById(id)).thenReturn(bulbasaur);
        assertThat(this.service.getById(id)).isNotNull();
        Mockito.verify(this.repo, Mockito.times(1)).findById(id);
    }
    @Test
    void testUpdate() {
        Long id = 1L;
        Pokemon bulbasaur = new Pokemon(1L, 1L, "bulbasaur", "grass", "poison"
        Optional<Pokemon> savedBulbasaur = Optional.ofNullable(new Pokemon(1L,
        Mockito.when(this.repo.findById(id)).thenReturn(savedBulbasaur);
        Mockito.when(this.repo.saveAndFlush(bulbasaur)).thenReturn(bulbasaur);
        assertThat(this.service.updatePokemon(1L, bulbasaur)).isNotNull();
        Mockito.verify(this.repo, Mockito.times(1)).findById(id);
    }
    @Test
    void testRemove() {
        assertThat(this.service.removePokemon(1L)).isEqualTo(!this.repo.exists
        Mockito.verify(this.repo, Mockito.times(1)).deleteById(1L);
    }

}

```

Console | Problems | Debug Shell | Coverage

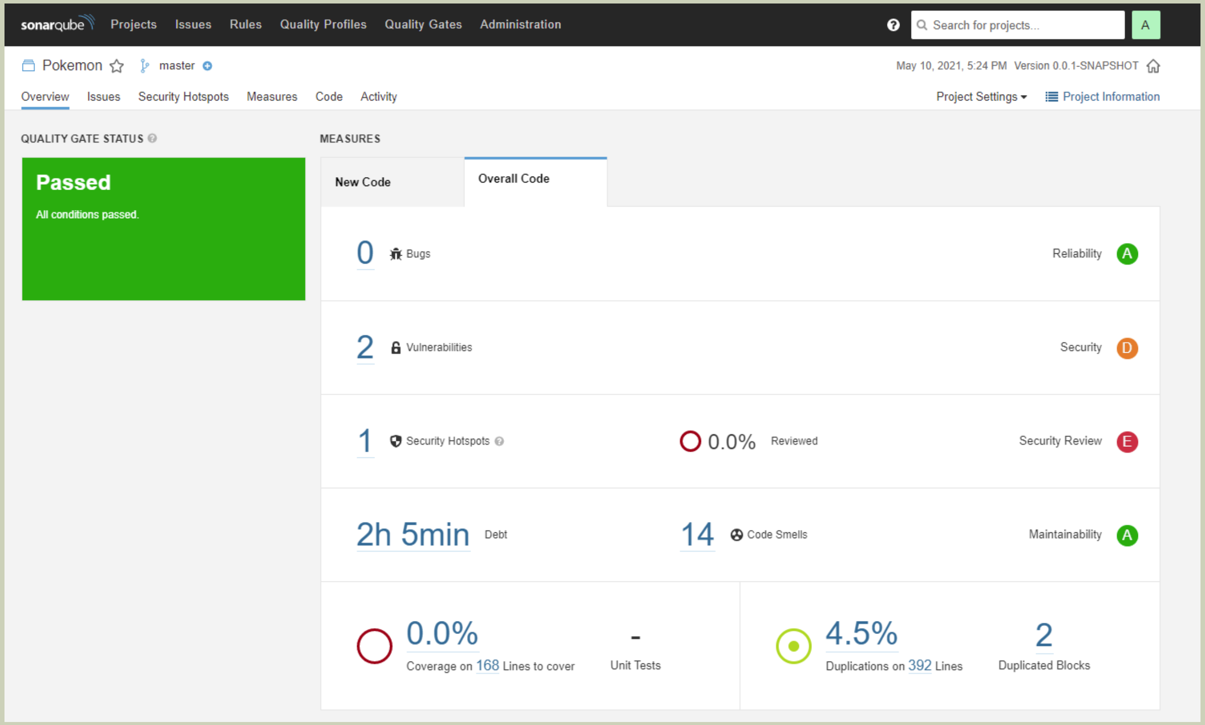
Pokemon (13 May 2021 10:50:58)

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
Pokemon	86.2 %	1,502	241	1,743



com.qa.pokemon.service	98.9 %	90	1	91
com.qa.pokemon.controller	100.0 %	45	0	45

TESTING CONT



QUALITY GATE STATUS

Passed

All conditions passed.

MEASURES

New Code | Overall Code

0 Bugs		Reliability	A
2 Vulnerabilities		Security	D
1 Security Hotspots	0.0% Reviewed	Security Review	E
2h 5min Debt	14 Code Smells	Maintainability	A

0.0% Coverage on 168 Lines to cover — - Unit Tests

4.5% Duplications on 392 Lines — 2 Duplicated Blocks

DEMONSTRATION

USER STORY:

- As a user, I must be able to view all entries in the database on the website, so that I can read all of the information.

USER STORY:

- As a user, I would like to view entries in a formatted manner, so then it is easier for me to read the information displayed.

DEMONSTRATION CONT

USER STORY:

- As a user, I must be able to add entries to the database, so then I can compare them with other entries in the database.

USER STORY:

- As a user, I must be able to update entries in the database, so that I can change the details of certain entries if they are incorrect or the details change.

USER STORY:

- As a user, I must be able to delete entries in the database, so then I can remove any entry I don’t need or any duplicates.

SPRINT REVIEW

During my initial sprint on the project I was able to achieve all my goals of delivering a minimum viable project according to the specification outlined.

Before I started my sprint I thought of other features that I could implement into my project to make it more visually appealing, these included having the data printed out as cards rather than a table so that users be able to identify each separate entity, as well as having a quantity counter that whenever a new entry was added to the database the quantity counter would increase, to show tha amount of entries in the database.

After I had created the minimum viable project I realised that I had sufficient time to complete one of these features, These two features weren't integral to the program so I decided to include the card feature to make the website more visually appealing, as the counter feature is not as important as making the website look cleaner, if the product owner wants can always implement this feature in an upcoming sprint.

SPRINT RETROSPECTIVE

On my initial sprint I found creating all the controller, service and domain for the backend as well as creating the modal for creating entries in the front end went successfully, the main thing that I struggled with implementing the update and remove buttons on the front end, I then spent a bit too much time trying to think of how to implement the methods which in turn made me create a method to delete the last entry in the database, looking back at the project now I think there are other ways I could of tried to implement the method that I didn’t think of originally, I also struggled with the update function as after I had created it some of the values were returning null, this was due to different names associated with the values.

In future I would consider looking at all the methods I could create at the beginning and assess what I will have to learn for the project as well as being consistent with my naming conventions, so then I can mitigate the time debugging and unable to progress.

CONCLUSION

While under going the projects I have learnt many valuable skills most importantly trouble shooting my code after I had made a mistake and repairing it quickly, I have also learnt how to prioritise my time doing what I can do with the knowledge I had and coming back to the function that was giving me the most problems.

After completing this project I realised that I may not always have the knowledge to achieve what I would like to do for any future projects, so I have gathered a collection of resources which I can use to reference, learn from and understand what would be the best approach.