ASSESSMENT 3 – ITERATION 2

1 WHERE?

*UML Class Diagram of Starting System*

A diagram of a political party

Description automatically generated

2 WHAT?

Goal for the iteration is to **complete at least the first six tasks** to create a subset of the full New Zealand election results page. I want the product of this iteration to at least meet the unit testing specifications, and compliance with relevant code style.

1. **Create Election**
2. **Display Election**
3. **Add Party to Election**
4. **Display Parties in Election**
5. **Add Electorates to Election**
6. **Display Electorates**
7. Set Electorate winner.
8. Display Electorate winners

These tasks will involve studying and manipulating an html, CSS, and JavaScript templates to match a given image.

Planned Task Sequence

|  |  |  |  |
| --- | --- | --- | --- |
| **TASK SEQUENCE** | **DETAILS** | **ESTIMATE** | **ACTUAL** |
| PLANNING | * Create the necessary Jasmine Spec Runner tests for Election.js, Electorate.js, and Party.js. * Style the page to reflect the desired output via TableStyle.css | 20 minutes | 5 minutes |
| ANALYSIS | * Look at the class diagram to review what aspects need to be tested for the .js classes. * Compare the desired output with the actual output from iteration 1. Researching the | 20 minutes | 3 hours |
| DESIGN | * Update the class diagram based on any insights from the analysis * List all the aspects in need of testing for the Spec Runner coding * Identify specific CSS design elements in need of adjustment, removal or addition. | 20 minutes | 30 minutes |
| CODING | * Coding the Spec Runner to test the aspects identified for testing. * Coding in the necessary css elements | 2 hours | 3 hours |
| TESTING | * Run the full code through the suite of style checking sites. * Ensure that the Spec Runner code reflect the description | 1 hour | 4 hours |
| - | - | 4 hours | 10 hours  35 minutes |

Estimated /Actual Timings for Each Task &Expected Product Output.

|  |  |  |  |
| --- | --- | --- | --- |
| TASK | ESTIMATE | PRODUCT | ACTUAL |
| Create spec runner unit tests: | No estimate | -ElectorateSpec.js  -ElectorateTableArraySpec.js  -PartySpec.js  -PartyTableArraySpec.js | 7 hours |
| Table formatting | No estimate | -table text color to blue  -thick table border  -‘vertical’column to be centered  -table header border removed | 3 hours |
| TOTAL HOURS | No estimate |  | 10 hours |

3 HOW?

**Unit Test**

I wrote up a small number of tests using Jasmine. These test the relevant attributes and methods of the Election.js, Party.js, and Electorate.js classes.

Some of the relevant attributes being tested were the names and type of the class. The other function of the tests was for ‘lint’ checking of the input to the attributes and methods (to table functions) of the classes.

**Feature Planning**

The primary feature of the iteration 2 is to display at least a table of the running parties of the New Zealand Election for 2023. There probably should be 72 (65 general and 7 Māori) electorates in the table, so we maybe need to adjust based on that.

*Desired Output*

A list of candidates

Description automatically generated

Figure -Desired Output

4 EVALUATION!

**Style Compliance**

Jshint

*Election.js*

*A screen shot of a computer

Description automatically generated*

Figure -JSHint: Election.js

*Party.js*

*A black rectangular object with a black border

Description automatically generated*

Figure -JSHint: Party.js

*Electorate.js*

*A black rectangular object with a black border

Description automatically generated*

Figure -JSHint: Electorate.js

*View.js*

*A black rectangular object with a black border

Description automatically generated*

Figure -JSHint: View.js

W3c html style validators

*index.html*

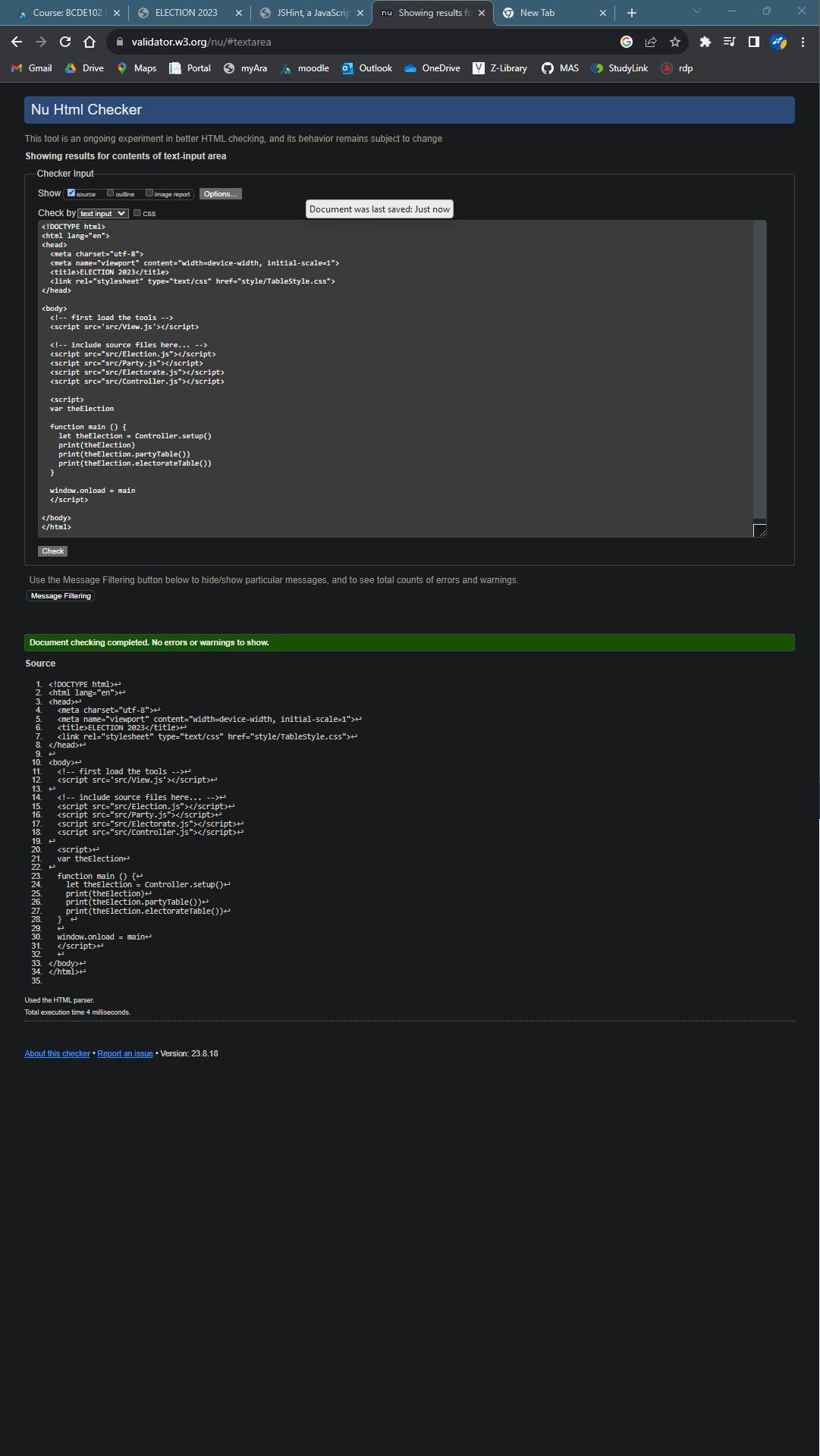
**

Figure -Nu Html Checker: index.html

**How Did The Plan Fare?**

*Unit Test Results*

A screenshot of a computer

Description automatically generated

Figure -Jasmine Unit Tests

*Page Display*

A screenshot of a computer

Description automatically generated

Figure -Actual Output

**Deviations From Iteration Plan**

There were no significant deviations from the original iteration plan.

**Review**

|  |  |
| --- | --- |
| *What worked* | *What could be done differently* |
| **Code Patterns**  Generally speaking, adapting code patterns for tables from a different source was quite straightforward. | **Save Time Debugging**  Immediately run any .js code through a code visualiser to identify any simple mistakes in terms of mistyped variables, forgotten function parenthesis etc… before I waste any time attempting to debug the code.  This was a particular issue when writing the code for the Jasmine Spec Runner units. |
| **CSS Styling Troubleshooting**  There were specific css stylings that were difficult to figure out initially, but with some careful analysis of the code the styling issues were resolved.  A prime example this is of the NEWLINE constant inside of the print function of View.js  Another is the table ‘vertical centering’ of the inner table border, the td is considering cells as their own unit. So separating the td allows me to centre align the vertical column border. | - |

*Discussion of Performance*

It is difficult to estimate the amount of time needed for every task and having the presence of mind to record the time spent doing those tasks.