Comprehensive Test Plan

Overall Test Plan

Since our project uses multiple technologies across multiple frameworks, we will be testing each component of our product individually as well as testing the finished product. The parts of our project include our web extension, REST API, and machine learning component. We will test the web extension to make sure that it can send the currently viewed websites DOM to the REST API as well as take the response from the API and pass it to the operating systems voice synthesis system. The extension will be user tested as well as unit tested using Selenium. Our testing for the REST API will revolve around ensuring speed and zero loss of data. Unit tests for this function will need to check the speed the DOM file is sent and received followed by an integrity test of the data being received. When Sending the newly processed data from the ML side, a similar test will be created to test speed and accuracy of speech text. For the machine learning component, we will test the model with test data to ensure that our model reaches an acceptable level of accuracy. Unit tests will be used to test the interfaces of the component. And end-to-end testing will be used to verify that the entire system behaves as specified.

Test Case Descriptions

WE1.1 **Web Extension Test 1**
WE1.2 This test will ensure that the extension can collect the DOM on page load
WE1.3 This test will load a web page and ensure that the extension waits for the full page to be loaded before getting the DOM of the given website so that any Javascript that modifies the page on load, has ran.
WE1.4 Inputs: The inputs for this test will be a website URL
WE1.5 Outputs: The output for the test will the correct DOM of the given URL
WE1.6 Normal
WE1.7 Whitebox
WE1.8 Functional
WE1.9 Unit Test

WE2.1 **Web Extension Test 2**
WE2.2 This test will ensure that the extension can call the REST API
WE2.3 This test will take the DOM collected from WE1 and make a POST REST call to our API
WE2.4 Inputs: The inputs for this test will be the DOM collected from WE1 and our API URL
WE2.5 Outputs: A successful HTTP status code returned from the API and a message from the API
WE2.6 Normal
WE2.7 Whitebox
WE2.8 Functional
WE2.9 Unit Test

WE3.1 **Web Extension Test 3**
WE3.2 This test will ensure that the extension can use the SpeechSynthesis API
WE3.3 This test will ensure that the browser supports the SpeechSynthesis API and any function of that API we need to use
WE3.4 Inputs: A constant string
WE3.5 Outputs: No errors from the browser and output from the speaker
WE3.6 Normal
WE3.7 Whitebox
WE3.8 Functional
WE3.9 Unit Test

WE4.1 **Web Extension Test 4**
WE4.2 This test will ensure that the extension can take the results from the API and pass them to the SpeechSynthesis API (Requires WE2 and WE3)

WE4.3 This test will take the output from WE2 and pass it to the SpeechSynthesis API so that the transcript of the page out read to the user
WE4.4 Inputs: Output from WE2
WE4.5 Outputs: Speech synthesis of the transcript of the page
WE4.6 Normal
WE4.7 Whitebox
WE4.8 Functional
WE4.9 Unit Test

WE5.1 **Web Extension Test 5**
WE5.2 This test will ensure that the user can change certain settings about the extension
WE5.3 This test will make sure the user can change certain settings (including the type of voice the user wants to use and rate of speed) and that they are persisted
WE5.4 Inputs: User interaction
WE5.5 Outputs: Saved settings
WE5.6 Normal
WE5.7 Whitebox
WE5.8 Functional
WE5.9 Unit Test

WE5.1 **Web Extension Test 6**
WE5.2 This test will ensure that all the test cases pass for every browser supported
WE5.3 This extension will be using the WebExtension standard which allows for our extension to be used across browsers in one codebase, some APIs may not be supported across browsers or function differently.
WE5.4 Inputs: Extension installed on a base set of browsers (Chrome, Firefox, Edge)
WE5.5 Outputs: All previous test cases passed
WE5.6 Normal
WE5.7 Whitebox
WE5.8 Functional
WE5.9 Unit Test

AP1.1 **REST API Test 1**
AP1.2 This Test will test accuracy of the REST API's transfer of the DOM from the Extension to the ML.
AP1.3 The file being sent from the Extension will be compared against the file received by the ML, checking for any corrupted data.
AP1.4 Inputs: The DOM file pulled from the Extension when a page is first loaded and the DOM received on the ML side.

AP1.5 Outputs: Percentage of accuracy based on two files.
AP1.6 Normal
AP1.7 Blackbox
AP1.8 Performance test
AP1.9 Unit Test

AP2.1 **REST API Test 2**
AP2.2 This Test will test accuracy of the REST API's transfer of the Speech Text from the ML to the Extension.
AP2.3 The file being sent from the ML will be compared against the file received by the Extension, checking for any corrupted data.
AP2.4 Inputs: The speech text file created by the ML, the file received on the Extension side.
AP2.5 Outputs: Percentage of accuracy based on two files.
AP2.6 Normal
AP2.7 Blackbox
AP2.8 Performance test
AP2.9 Unit Test

AP3.1 **REST API Test 3**
AP3.2 This Test will test the speed of the REST API's transfer of the DOM from the Extension to the ML.
AP3.3 The file being sent from the Extension will be timed from time sent to time received by ML.
AP3.4 Inputs: The DOM file of the page pulled from the extension.
AP3.5 Outputs: Time from file being sent to file received.
AP3.6 Normal
AP3.7 Blackbox
AP3.8 Performance Test
AP3.9 Unit Test

AP4.1 **REST API Test 4**
AP4.2 This Test will test the speed of the REST API's transfer of the Speech Text from the ML to the Extension.
AP4.3 The speech text file being sent from the ML will be timed from time sent to time received by Extension.
AP4.4 Inputs: The Speech Text created by the ML component.
AP4.5 Outputs: Time from file being sent to file received.
AP4.6 Normal
AP4.7 Blackbox

AP4.8 Performance Test
AP4.9 Unit Test

ML1.1 **Machine Learning Test 1**
ML1.2 This test will ensure that the model can correctly classify data instances.
ML1.3 This test will send test data to the model for classification.
ML1.4 Inputs: The inputs for this test will be test data instances.
ML1.5 Outputs: The classification of the data instances will be output by the model.
ML1.6 Normal
ML1.7 Blackbox
ML1.8 Functional
ML1.9 Unit Test

ML2.1 **Machine Learning Test 2**
ML2.2 This test will ensure that the interface to the machine learning component of the system correctly responds to a request from the REST API component..
ML2.3 This test will send data from the REST API component to the machine learning component and the machine learning component will respond to the REST API.
ML2.4 Inputs: The inputs for this test will be a test data instance.
ML2.5 Outputs: The machine learning model component will respond to a request from the REST API component..
ML2.6 Normal
ML2.7 Whitebox
ML2.8 Functional
ML2.9 Integration Test

ML3.1 **Machine Learning Test 3**
ML3.2 This test will ensure that the machine learning component classifies data in a reasonable time.
ML3.3 This test will send test data to the model for classification and time how long it takes to receive the results from that data.
ML3.4 Inputs: The inputs for this test will be test data instances.
ML3.5 Outputs: The classification of the data instances will be output by the model.
ML3.6 Normal
ML3.7 Blackbox
ML3.8 Performance
ML3.9 Unit Test

Test Case Matrix

	Normal/ Abnormal	Blackbox/ Whitebox	Functional/ Performance	Unit/ Integration
WE1	Normal	Whitebox	Functional	Unit
WE2	Normal	Whitebox	Functional	Unit
WE3	Normal	Whitebox	Functional	Unit
WE4	Normal	Whitebox	Functional	Unit
WE5	Normal	Whitebox	Functional	Unit
WE6	Normal	Whitebox	Functional	Unit
AP1	Normal	Blackbox	Performance	Unit
AP2	Normal	Blackbox	Performance	Unit
AP3	Normal	Blackbox	Performance	Unit
AP4	Normal	Blackbox	Performance	Unit
ML1	Normal	Blackbox	Functional	Unit
ML2	Normal	Whitebox	Functional	Integration
ML3	Normal	Blackbox	Performance	Unit