**Single Semester Snobs, We Don’t Byte, Five Guys, KJBK**

**Mind\_Reader**

**TEST PLAN**

Date: 4/08/2023

**Table of Contents**

# 1 INTRODUCTION

1.1 OBJECTIVES

1.2 TEAM MEMBERS

# 2 SCOPE

# 3 ASSUMPTIONS / RISKS

3.1 ASSUMPTIONS

3.2 RISKS

# 4 TEST APPROACH

4.1 TEST AUTOMATION

4.2 TEST CASES (BLACK BOX)

4.3 TEST CASES (WHITE BOX)

5 TEST ENVIRONMENT

6 TEST SCHEDULE

# 

# **Introduction**

The Test Plan has been created to communicate the test approach to team members. It includes the objectives, scope, schedule, risks, and approach. This document will clearly identify what the test deliverables will be and what is deemed in and out of scope.

## **1.1 Objectives**

The current editor options available lack the level of accessibility that is required to allow students who are visually impaired to edit, write, and debug code adequately. This tool would extend Visual Studio Code’s existing accessibility options to allow everyone, primarily students K-12, to learn to program with Lego Mindstorms. Our goal is to provide an enhanced experience for students who are visually impaired that are transparent to sighted students. This allows everyone to use the same software solution, whether or not they are vision impaired.

Sprint 1 of the project will repair old extensions for the new environment, and deliver functions for voice control.

Sprint 2 of the project will deliver voice actions to VS code, and pull auto-complete from VSCode for Use in Audio Cues.

Sprint 3 of the project will deliver all the functional updates and make the extension available.

Sprint 4 of the project delivers bug removal and updates current functionalities. Autocomplete feature is accessible now.

Sprint 5 of the project delivers a complete voice hotkey and autocomplete function.

Sprint 6 of the project delivers the NLTK library for natural language processing, with a few more parameters to control the voice input.  
  
Sprint 7 of the project will release the extension completely with final testing.

## **1.2 Team Members**

| **Resource Name** | **Team Name** | **Year** | **Role** |
| --- | --- | --- | --- |
| Mason Bone | Single Semester Snobs | 2021 | Requirements Lead |
| Sophia Drewfs | Single Semester Snobs | 2021 | Design Lead |
| Jake Grossman | Single Semester Snobs | 2021 | Sponsor Lead |
| Josiah Moses | Single Semester Snobs | 2021 | Team Lead |
| Cal Wooten | Single Semester Snobs | 2021 | Testing Lead |
| Pedro Alvarez | We Don’t Byte | 2022 | Requirements Lead |
| Ryan Tolbert | We Don’t Byte | 2022 | Design Lead |
| Kendrick Johnson | We Don’t Byte | 2022 | Sponsor Lead |
| John Breaux | We Don’t Byte | 2022 | Requirements Lead |
| Thomas Lane | We Don’t Byte | 2022 | Team Lead |
| Haris Javed | Five Guys | 2022-23 | Requirements Lead |
| Jigme Sherpa | Five Guys | 2022-23 | Testing Lead |
| Zachary Chenausky | Five Guys | 2022-23 | Design Lead |
| Clay Lewis | Five Guys | 2022-23 | Team Lead |
| Saad Javed | Five Guys | 2022-23 | Sponsor Lead |
| Kyle Minter | KJBK | 2024-25 | Team Lead |
| Jacob Franks | KJBK | 2024-25 | Requirements Lead |
| Kennedy Boynton | KJBK | 2024-25 | Design Lead |
| Brett Berglund | KJBK | 2024-25 | Testing Lead |

# 

# **2 Scope**

At the end of Sprint 1, a user must be able to:

R1: Open an accessibility menu,

R2: change the theme,

R3: change the font size, and

H1: navigate the interface.

Access the old extensions without any bugs

R11: Use speech-to-function option

H12: access autocomplete options of partial functions

● Testing for usability, cohesion, and compatibility.

At the end of Sprint 2, a user must be able to:

H2: Navigate text editor,

H3: get input help,

H4: listen to cursor,

H5: listen to line context,

H6: get indentation level,

H7: be alerted of syntax errors,

H8: goto syntax errors.

H11: Move Cursor through hotkey

● Testing for ease of use and precision

At the end of Sprint 3, a user must be able to:

R4: Connect with LEGO SPIKE Prime Hub,

R5: Disconnect from LEGO SPIKE Prime Hub,

R6: Save files to LEGO SPIKE Prime Hub,

R7: Run filed from LEGO SPIKE Prime Hub,

R8: Delete Files from LEGO SPIKE Prime Hub,

R9: Stop execution, and open the extension in vscode and be able to program a lego mindstorm robot with

R12: partially use the voice hotkey

Use all the options available on the extension

At the end of Sprint 4, a user must be able to:

R4: Connect with LEGO SPIKE Prime Hub,

R5: Disconnect from LEGO SPIKE Prime Hub,

R6: Save files to LEGO SPIKE Prime Hub,

R7: Run filed from LEGO SPIKE Prime Hub,

R8: Delete Files from LEGO SPIKE Prime Hub,

R9: Stop execution, and open the extension in vscode and be able to program a lego mindstorm robot with

H12: access autocomplete options of partial functions

At the end of Sprint 5, a user must be able to:

R4: Connect with LEGO SPIKE Prime Hub,

R5: Disconnect from LEGO SPIKE Prime Hub,

R6: Save files to LEGO SPIKE Prime Hub,

R7: Run filed from LEGO SPIKE Prime Hub,

R8: Delete Files from LEGO SPIKE Prime Hub,

R9: Stop execution, and open the extension in vscode and be able to program a lego mindstorm robot with

Fully work with autocomplete functions

Partially work with voice hotkeys function

Be able to access Hover/Definition Provider

At the end of Sprint 6, a user must be able to:

R4: Connect with LEGO SPIKE Prime Hub,

R5: Disconnect from LEGO SPIKE Prime Hub,

R6: Save files to LEGO SPIKE Prime Hub,

R7: Run filed from LEGO SPIKE Prime Hub,

R8: Delete Files from LEGO SPIKE Prime Hub,

R9: Stop execution, and open the extension in vscode and be able to program a lego mindstorm robot with

Fully work with autocomplete functions

Fully work with voice hotkeys function

Be able to access Hover/Definition Provider

At the end of Sprint 7, a user must be able to:

R4: Connect with LEGO SPIKE Prime Hub,

R5: Disconnect from LEGO SPIKE Prime Hub,

R6: Save files to LEGO SPIKE Prime Hub,

R7: Run filed from LEGO SPIKE Prime Hub,

R8: Delete Files from LEGO SPIKE Prime Hub,

R9: Stop execution, and open the extension in vscode and be able to program a lego mindstorm robot with

Fully work with autocomplete functions

Fully work with voice hotkeys function

Be able to access Hover/Definition Provider

Access the release of the extension

● Testing for security and publication

## **3 Assumptions / Risks**

### 3.1 Assumptions

*This section lists assumptions that are made specific to this project.*

● The user will have VScode and a screen reader (NVDA, JAWS, or VoiceOver)

● The user will have a Lego Mindstorm EV3, Spike Prime, or Thinker robot

● The user will have a microphone and the system has a sound card.

3.2 Risks

The following risks have been identified and the appropriate action identified to mitigate their impact on the project. The impact (or severity) of the risk is based on how the project would be affected if the risk was triggered. The trigger is what milestone or event would cause the risk to become an issue to be dealt with.

| **#** | **Risk** | **Impact** | **Trigger** | **Mitigation Plan** |
| --- | --- | --- | --- | --- |
| 1 | If the user gets lost in the editor/wrong voice input and isn’t able to get to the desired place. | High | An accidental command/key  binding trigger | There will be key bindings to allow the user to get to a home page for the extension and allow the user to get back to where they need to go. |
| 2 | VScode’s compatibility with a public screen reader is less than advertised | Medium | If VScode reads a line improperly or doesn’t tell the user the correct information. | The extension will mitigate these issues by implementing functions optimizing content for the user’s selected screen-reader. |

# 

# **4 Test Approach**

The project is using an agile approach, with 3-week sprints. *Mention how you will conduct testing during the sprint in terms of the techniques you plan to do and when. Add a new subsection for each sprint.*

Sprint 4

Unit testing for the LEGO SPIKE Prime Hub control and parser.

Bug removal and updates current functionalities. Autocomplete feature is accessible now

Sprint 5

Functional testing of LEGO SPIKE Prime Hub connectivity and running of extension. Delivers a complete voice hotkey and autocomplete function.  
  
Sprint 6

Integration testing for the extension through the VScode Extension Development Host. Getting done with the final functionality.   
  
Sprint 7

Final testing of the extension and then releasing it.

## **4.1 Test Automation**

Text parsing features are tested using the Mocha testing framework for TypeScript. A GitHub action runs this test on Windows, MacOS, and Linux each time a commit or pull request is made to the repository.

## 

## **4.2 Test Cases (Black Box)**

### **4.2.1 Feature R-1 Open accessibility menu**

| **Test**  **Case ID** | **Description** | **Requirements Trace** | **Directions** | | **Expected Output** |
| --- | --- | --- | --- | --- | --- |
| TC-1101 | User Right Clicks on active editor | R-1 | 1.  2.  3. | The user right clicks on the mouse in an open editor. The user scrolls down to the Mind\_Reader sub menu and highlights it.  The | The user should be able to increase the font/editor scale through this menu as well as change the theme of the developer |
| TC-1102 | User presses  Shift+F10  To open context menu | R-1 | 1. The user presses Shift+F10 and opens the context menu 2. The user uses arrow keys and goes down to the mind reader submenu 3. The user enters the submenu where all accessibility options will be | | The user should be able to increase the font/editor scale through this menu as well as change the theme of the developer |
| TC-1103 | R1: Open Accessibility  Menu | H-1 | 1. The user presses the menu hotkey to open the accessibility menu in the left pane. | | The accessibility menu is displayed and gains focus |
|  |  |  |  |  |  |

### 

### **4.2.2 Feature R-2 Change Theme**

| **Test**  **Case ID** | **Description** | **Requirements**  **Trace** | **Directions** | **Expected Output** |
| --- | --- | --- | --- | --- |
| TC-1201 | The user opens change theme with the accessibility  menu | R-1  R-2 | 1. The user opens the accessibility menu. 2. The user clicks the change theme item in the menu and opens the theme menu in the command palette. 3. The user then changes to the theme of their choice | The built in VSCode command palette menu should open with options for the themes. |
| TC-1202 | The user opens the change theme | R-1  R-2 | 1. The user uses Ctrl+K and Ctrl+T in sequence to open the Change theme menu in VSCode | The VSCode command palette menu should open with options for the themes. |

### **4.2.3 Feature R-3 Change Font Size and Scale**

| **Test**  **Case ID** | **Description** | **Requirement s Trace** | **Directions** | | **Expected Output** |
| --- | --- | --- | --- | --- | --- |
| TC-1301 | The user opens the access editor context menu and  selects Increase, Decrease, or  Reset Font Size | R-1  R-3 | 1. 2. | The user opens the accessibility menu through the editor  context menu  The user then chooses between  Increase, Decrease, or Reset  Font Size | Depending on the command entered the developer should change the Font Scaling by the users requested input |
| TC-1302 | The user opens the command palette menu and selects Increase,  Decrease, or  Reset Font Size | R-1  R-3 | 1.  2.  3. | The user opens the command palette by using the hotkey  Ctrl+Shift+P  The user then can type  Increase, Decrease, or Reset  Font Size  The user can choose between these | Depending on the command entered the developer should change the Font Scaling by the users requested input |
| TC-1303 | The user opens the access editor context menu and  selects Increase,  Decrease, or  Reset  Developer Scale | R-1  R-3 | 1.      2. | The user opens the accessibility menu through the editor context menu  The user then chooses between  Increase, Decrease, or Reset  Developer Scale | Depending on the command entered the developer should change the Developer Scaling by the users requested input |
| TC-1304 | The user opens the command palette menu and selects Increase,  Decrease, or  Reset  Developer Scale | R-1  R-3 | 1.  2.  3. | The user opens the command palette by using the hotkey  Ctrl+Shift+P  The user then can type  Increase, Decrease, or Reset  Developer Scale  The user can choose between these | Depending on the command entered the developer should change the Developer Scale by the users requested input |

### **4.2.4 Feature R4 Connect LEGO SPIKE Prime Hub**

| **Test**  **Case ID** | **Description** | **Requirement**  **Trace** | **Directions** | **Expected Output** |
| --- | --- | --- | --- | --- |
| TC-1401 | The user runs  the “Connect LEGO SPIKE Prime Hub” command to connect a LEGO SPIKE Prime Hub. |  | 1. Run “Connect LEGO SPIKE Prime Hub”  2. Select an open serial port from the quickpick list | The LEGO SPIKE Prime Hub is connected and a connection message is output. |

### 

### 

### **4.2.5 Feature R5 Disconnect LEGO SPIKE Prime Hub**

| **Test**  **Case ID** | **Description** | **Requirement**  **Trace** | **Directions** | **Expected Output** |
| --- | --- | --- | --- | --- |
| TC-1501 | The user runs the  “Disconnect LEGO SPIKE Prime Hub” command to connect a LEGO SPIKE Prime Hub. | R-4 | 1. Run “Disconnect LEGO SPIKE Prime Hub” | The LEGO SPIKE Prime Hub is disconnected and a disconnection message is output. |

### **4.2.6 Feature R6 Save files to LEGO SPIKE Prime Hub**

| **Test**  **Case ID** | **Description** | **Requirement**  **Trace** | **Directions** | **Expected Output** |
| --- | --- | --- | --- | --- |
| TC-1601 | The user uploads the current files to a program slot on the LEGO SPIKE Prime Hub. | R-4 | 1. Run “Upload current file”  2. Pick a program slot from the quicklist | The current file is uploaded to the LEGO SPIKE Prime Hub on the specified slot and a message is output. |

### 

### **4.2.7 Feature R7 Run files from LEGO SPIKE Prime Hub**

| **Test**  **Case ID** | **Description** | **Requirement**  **Trace** | **Directions** | **Expected Output** |
| --- | --- | --- | --- | --- |
| TC-1701 | The user runs  a file on the LEGO SPIKE Prime Hub | R-4 | 1. Run “Run Program”  2. Pick a program slot from the quicklist | The selected program is run. |
| TC-1702 | The user runs a file with an  error on the LEGO SPIKE Prime Hub | R-4 | 1. Run “Run Program” 2. Pick a program slot from the quicklist | The selected program is run, and the errors are output to the “SPIKE Prime Output” panel. |

### **4.2.8 Feature R8 Delete files from LEGO SPIKE Prime Hub**

| **Test**  **Case ID** | **Description** | **Requirement**  **Trace** | **Directions** | **Expected Output** |
| --- | --- | --- | --- | --- |
| TC-1801 | The user deletes a file from a program slot on the LEGO SPIKE Prime Hub. | R-4 | 1. Run “Delete Program”  2. Pick a program slot from the quicklist | The selected program is removed from the slot on the LEGO SPIKE Prime Hub. |

### **4.2.9 Feature R11 gets Speech-to-function**

| **Test**  **Case ID** | **Description** | **Requirement**  **Trace** | **Directions** | **Expected Output** |
| --- | --- | --- | --- | --- |
| TC-1901 | The user will be able to use their voice on the LEGO SPIKE Prime Hub to work on functions. | R-11 | 1. Run “speech-to-text”  2. Give permission for microphone access. | The selected option will let you write a function for the LEGO SPIKE Prime Hub through speaking. |

### 

### **4.3.0 Feature R12 gets activates voice hotkey**

| **Test**  **Case ID** | **Description** | **Requirement**  **Trace** | **Directions** | **Expected Output** |
| --- | --- | --- | --- | --- |
| TC-2001 | The user will be able to use their voice on the LEGO SPIKE Prime Hub to work for hotkeys. | R-11 | 1. Run “voice activation Hotkey”  2. Give the permission for microphone access. | The selected option will let you perform a hotkey function for the LEGO SPIKE Prime Hub through speaking. |

## **4.3.1 Feature H9 Get Number Of Spaces**

| **Test**  **Case ID** | **Description** | **Requirement**  **Trace** | **Directions** | **Expected Output** |
| --- | --- | --- | --- | --- |
| TC-3009 | The user uses get spaces shortcut with a document open | H-9 | 1. User presses Ctrl+Shift+G  2. User then presses Space | The number of leading spaces is printed to the screen and read aloud |
| TC-3010 | The user uses get spaces shortcut without a document open | H-9 | 1. User presses Ctrl+Shift+G  2. User then presses Space | “No document is currently active” is printed and read aloud |

## 

## **4.3.2 Feature H10 Get Line Number**

| **Test**  **Case ID** | **Description** | **Requirement**  **Trace** | **Directions** | **Expected Output** |
| --- | --- | --- | --- | --- |
| TC-3101 | The user uses get line number shortcut with a document open | H-10 | 1.User presses Ctrl + Shift + /  2. User then presses L | The line number containing the cursor is printed to the screen and read aloud |
| TC-3102 | The user uses get line number shortcut without a document open | H-10 | 1.User presses Ctrl + Shift + /  2. User then presses L | “No document is currently active” is printed and read aloud |
| TC-3103 | The user uses get line number shortcut on a line >1,000 | H-10 | 1. User presses Ctrl + Shift + /  2. The user then presses L | The entire line number containing the cursor is printed to the screen and read aloud |
| TC-3104 | The user uses getline number shortcut on a line < 500 | H-10 | 1.User presses Ctrl + Shift + /  2. The user then presses L | The line number containing the cursor is printed to the screen and read aloud |

## **4.3.3 Feature H12 Autocomplete function**

| **Test**  **Case ID** | **Description** | **Requirement**  **Trace** | **Directions** | **Expected Output** |
| --- | --- | --- | --- | --- |
| TC-3106 | Using the hotkey after typing a partial function or variable will speak the Intellisense autocomplete options | H-12 | 1. User goes to settings  2. Clicks on extension  3. Click on Mind Reader  4. Find the Autocomplete Option.  5. Reload VSCode | The outputs will have the option to autocomplete and the user will have a chance to change the outcome. |

## **4.3.4 Feature H13 Hover/Definition Provider**

| **Test**  **Case ID** | **Description** | **Requirement**  **Trace** | **Directions** | **Expected Output** |
| --- | --- | --- | --- | --- |
| TC-3108 | The hover provider is one of the key features of vs-code to get quick definitions of code components such as functions, variables, keywords, etc. | H-13 | 1. User goes to settings  2. Clicks on extension  3. Click on Mind Reader  4. Find the extension Option.  5. Reload VSCode | we will try to make this feature accessible to completely blind people. Instead of having to hover over certain texts, the users can ask for definitions using texts.  . |

## **4.3.5 Feature R10 Line Highlighter**

| **Test**  **Case ID** | **Description** | **Requirement**  **Trace** | **Directions** | **Expected Output** |
| --- | --- | --- | --- | --- |
| TC-3201 | The highlighter highlights the line containing the cursor | R-10 | 1. Go to Settings (File -> Preferences -> Settings)  2. Click on Extensions  3. Click on Mind Reader  4. Find the Line Highlighter Options  5. Reload VSCode | The line number containing the cursor is highlighted |
| TC-3202 | The highlighter can highlight multiple lines | R-10 | 1. Go to Settings (File -> Preferences -> Settings)  2. Click on Extensions  3. Click on Mind Reader  4. Find the Line Highlighter Options  5. Reload VSCode | Each line clicked will be highlighted |
| TC-3203 | The user can change the highlight color | R-10 | 1. Go to Settings (File -> Preferences -> Settings)  2. Click on Extensions  3. Click on Mind Reader  4. Change the background color field  5. Reload VSCode | The highlight changed to the value entered by the user |
| TC-3204 | The user can change the text color within the highlight | R-10 | 1. Go to Settings (File -> Preferences -> Settings)  2. Click on Extensions  3. Click on Mind Reader  4. Change the text color field  5. Reload VSCode | Only the text color contained within the highlight is changed |
| TC-3205 | The user can change the text style (bold, italic, underlined) | R-10 | 1. Go to Settings (File -> Preferences -> Settings)  2. Click on Extensions  3. Click on Mind Reader  4. Change the font style/weight, text decoration  5. Reload VSCode | The text style changes to what is selected (bold, italic, or underlined) |
| TC-3206 | The user can edit the border of the highlighter | R-10 | 1. Go to Settings (File -> Preferences -> Settings)  2. Click on Extensions  3. Click on Mind Reader  4. Change the border color/style/width  5. Reload VSCode | Only the border is updated with the new configuration |
| TC-3207 | The user can edit the outline of the highlighter | R-10 | 1. Go to Settings (File -> Preferences -> Settings)  2. Click on Extensions  3. Click on Mind Reader  4. Change the outline color/style/width  5. Reload VSCode | Only the outline is updated with the new configuration |
| TC-3208 | No text color is selected in the highlighter | R-10 | 1. Go to Settings (File -> Preferences -> Settings)  2. Click on Extensions  3. Click on Mind Reader  4. Locate the textColor field to ensure nothing is listed.  5. Reload VSCode | The default text color set by VSCode is used if no custom color is set. |
| TC-3209 | Enable/Disable the line highlighter | R-10 | 1. Go to Settings (File -> Preferences -> Settings)  2. Click on Extensions  3. Click on Mind Reader  4. Tick/Untick the isEnabled box  5. Reload VSCode | The line highlighter is enabled or disabled depending on if the box is ticked or not. |

**4.5.1 Feature R14, R15, R16, R17, R18, R19, R20 Audio Flags**

| **Test**  **Case ID** | **Description** | **Requirement**  **Trace** | **Directions** | **Expected Output** |
| --- | --- | --- | --- | --- |
| TC-4001 | The user can add an audio flag to a line and choose a sound. | R14: Add Audio Flag | 1. Move the cursor to any line in the document  2. Click “Add Audio Flag”  3. Choose a sound for the flag from the pop up prompt | The line selected has a flag added to it with the selected sound. |
| TC-4002 | The user cannot add an audio flag to a line in which one is already present. | R14: Add Audio Flag | 1. Move the cursor to any line in the document  2. Click “Add Audio Flag”  3. Choose a sound for the flag from the pop up prompt.  4. Without moving the cursor, click “Add Audio Flag.” | An error message should pop up informing the user that there is already an audio flag on the current line. |
| TC-4003 | The user can preview audio flag sounds when adding a flag to a document. | R14: Add Audio Flag | 1. Move the cursor to any line in the document.  2. Click "Add Audio Flag"  3. Use arrow keys to highlight each option in the dropdown prompt.  4. Click the preview sound button in the dropdown menu. | The highlighted sound option should play upon the preview sound button being clicked. |
| TC-4004 | The user can delete an audio flag from a line in a document. | R16: Delete Audio Flag | 1. Move the cursor to a line with an audio flag on it.  2. Click “Delete Audio Flag.” | The audio flag will be removed from the current line in the document. |
| TC-4005 | The user cannot delete an audio flag from a line in which there is not one present. | R16: Delete Audio Flag | 1. Move the cursor to any line in the document.  2.Click "Delete Audio Flag." | An error message should pop up informing the user that there is no preexisting audio flag to delete from the file. |
| TC-4006 | Audio flags save and load with their respective file. | R17: Saving Audio Flags | 1. Move the cursor to any line in the document.  2. Click “Add Audio Flag.”  3. Choose a sound for the flag from the pop up prompt.  4. Save the document.  5. Close the document.  6. Reopen the document | The audio flag will be added, saved, and loaded onto the selected line in the document. |
| TC-4007 | Deleted audio flags will be removed from the document. | R17: Saving Audio Flags | 1. Move the cursor to line with an audio flag.  2. Click “Delete Audio Flag.”  3. Save the document.  4. Close the document.  5. Reopen the document. | The deleted audio flag should be removed from the document and not reappear upon reopen. |
| TC-4008 | Jump to pre pre-existing flag from the current line. | R15: Move to Audio Flag | 1. Move the cursor to line without an audio flag on it.  2. Click “Move to Audio Flag.” | The cursor should jump to the next flag down in the file. |
| TC-4009 | The audio flag does not play sound while the screen reader is playing | R19: Audio Flag Compatibility With Screen Reader | 1. Move the cursor to line without an audio flag on it.  2. Click “Move to Audio Flag.” | The audio for the flag should not play. |
| TC-4010 | Hotkey for adding audio flags to files. | H13: Place Audio Flag Hotkey | 1. Move the cursor to any line in the document  2. Click “Ctrl+Shift+8” to add an audio flag.  3. Choose a sound for the flag from the pop up prompt | The line selected has a flag added to it with the selected sound. |
| TC-4011 | Hotkey for moving the cursor to a line with audio flags | H14: Jump to Audio Flag Hotkey | 1. Move the cursor to line without an audio flag on it.  2. Click “Ctrl+Shift+7” to move the cursor to an audio flag. | The cursor should jump to the next flag down in the file. |
| TC-4012 | Hotkey for deleting audio flags from files. | H15: Delete Audio Flag Hotkey | 1. Move the cursor to a line with an audio flag on it.  2. Click “Ctrl+Shift+9” to delete an audio flag. | The audio flag will be removed from the current line in the document. |
| TC-4013 | The user will be able to search for text within the subset of audio flags | R18: Search Audio Flags | 1. Click "Search audio flags"  2. Type in text, matching a line with an audio flag on it, into the field | The text will be found on a line that is flagged and it will be highlighted. |
| TC-4014 | The user will not be able to search for text not within the subset of audio flags | R18: Search Audio Flags | 1. Click "Search audio flags"  2. Type in text, that does not match any lines with audio flags on them, into the field | The text will not be found and no text will be highlighted. |
| TC-4015 | User will be able to search for audio flags while ignoring case sensitivity | R18: Search Audio Flags | 1. Click “Search audio flags”  2. Type in text, that matching a line with an audio flag on it, but an opposite case, into the field | The search function disregards case sensitivity to make finding desired text more intuitive. |
| TC-4016 | User can now search for audio and have it scroll to the highlighted text | R18: Search Audio Flags | 1. Click “Search audio flags”  2. Type in desired text into the field | If there are no highlighted pieces of text on the screen, it will scroll to the nearest match. |
| TC-4017 | Users can change the highlighter color of matched text when searching for audio flags via the JSON settings file. | R18: Search Audio Flags | 1. Click “Search audio flags”  2. Type in text matching a flagged line’s text (take note of the highlight color)  3. Change the searchHighligher color settings in the JSON file.  4. Save the JSON file.  5. Return to the file containing audio flags. 6. Click “Search audio flags”  7. Type in text matching a flag line’s text. | The color of the highlighter should be changed to match the user settings in the JSON file. |
| TC-4018 | Users can change the highlighter color of matched text when searching for audio flags via the UI view. | R18: Search Audio Flags | 1. Click “Search audio flags”  2. Type in text matching a flagged line’s text (take note of the highlight color)  3. Change the searchHighligher color settings in the UI view.  4. Return to the file containing audio flags. 5. Click “Search audio flags”  6. Type in text matching a flag line’s text. | The color of the highlighter should be changed to match the user settings in the UI view. |
| TC-4019 | Users can now move the cursor to searched text. | R15: Move to Audio Flag | 1. Click "Search audio flags"  2. Type in text matching a flagged line's text  3. Click the enter key to jump to a line with the matching text. | The search popup should not close and will remain open for continued searches |
| TC-4020 | Users can preview the sound associated with a preexisting audio flag. | R19: Audio Flag Compatibility With Screen Reader | 1. Move the cursor to a line with a preexisting audio flag on it.  2. Click "Play Line Audio" | The sound associated with the preexisting audio flag should play upon the button being clicked. |
| TC-4021 | Users can preview the sound associated with a preexisting audio flag. | H16: Play Line Audio | 1. Move the cursor to a line with a preexisting audio flag on it.  2. Click "Ctrl+Shift+6" | The sound associated with the preexisting audio flag should play upon the button being clicked. |
| TC-4022 | Users can preview the sound cue for a line | R19: Audio Flag Compatibility With Screen Reader | 1. Move the cursor to a line without a preexisting audio flag on it.  2. Click "Play Line Audio" | The sound context for the line will be played upon the button being clicked. |
| TC-4023 | Users can preview the sound cue for a line | H16: Play Line Audio | 1. Move the cursor to a line without a preexisting audio flag on it.  2. Click "Ctrl+Shift+6" | The sound context for the line will be played upon the button being clicked. |
| TC-4024 | R18, R10: Compatibility between line highlighter with audio flag search highlighter | 1.Workspace is active  2. A text document, with audio flags  3. Search audio flag window is open, with typed search criteria | 1. Drag cursor and highlight text on the same line as the text highlighted by the search function. | The line highlighter takes priority over the search highlighter. Other line highlighter functions work with the user highlighted text. |

## **4.4 Test Cases (White Box)**

| **Test Case ID** | **Description** | **Direction/goals** | **Expected Output** |
| --- | --- | --- | --- |
| TC-2201 | The user will be able to use their voice on the LEGO SPIKE Prime Hub to work on functions. | This should be able to verify the speech and configure the function | The expected output should be that the function is getting written through speaking. Currently, the program is having some issues with accessing voice. |
| TC-2202 | The user will be able to use their voice on the LEGO SPIKE Prime Hub to work for hotkeys. | This should be able to verify the voice and configure the hotkeys | The expected output should be that the following hotkey is accessed through speaking. Currently, the program is still having some issues with accessing voice. |

### 

### **4.4.1 Lexer**

| **Test Case ID** | **Input** | **Expected Output** |
| --- | --- | --- |
| TC-2001 | Empty Input | No Token |
| TC-2002 | Undefined | No Token |
| TC-2003 | Whitespace only | No Token |
| TC-2004 | Comment only | COMMENT Token containing contents of comment |
| TC-2005 | Single Statement | STATEMENT Token containing contents of statement |
| TC-2006 | getIndent() accuracy for spaces and tabs | Tokens with an appropriate level of indent |
| TC-2007 | Control Statement (for, while, try, etc.) | New token with next level of indent |
| TC-2008 | Restart() | Lexer is refreshed with new input |

### **4.4.2 LexNode**

| **Test Case ID** | **Input** | **Expected Output** |
| --- | --- | --- |
| TC-2101 | No Input | No Token |
| TC-2102 | Single Empty Line | No Token |

# 

# **5 Test Environment**

The majority of our testing will be completed in the Extension Development Host provided by VScode. This is a built-in function in VScode to allow developers to test their extensions without requiring publication. It does this by opening a new instance of VScode specifically made for testing your extension and loading your extension into it for testing. The integration testing with Mocha is performed in the Extension Development Host as well.

# **6 Test Schedule**

| **Task Name (sample is below, focus on sprint 1 to start)** | **Start** | **Finish** | **Effort (1-5)** | **Comments** |
| --- | --- | --- | --- | --- |
| *Test Planning* | 10/14//21 | 11/13/21 | 2 |  |
| *Review Requirements documents* | 10/17/21 | 11/13/21 | 2 |  |
| *Create initial test estimates* | 10/18/21 | 11/1/21 | 3 |  |
| *Hotkey Testing* | 10/21/21 | 11/13/21 | 4 | Manual testings required |
| *First deploy to QA test environment* | 10/21/21 | 10/26/21 | 3 |  |
| *Functional testing – Sprint 1* | 10/18/21 | 10/26/21 | 5 | Testing of all functions will be retesting to ensure old code doesn’t conflict. |
| *Iteration 2 deploy to QA test environment* | 10/26/21 | 11/13/21 | 3 |  |
| *Functional testing – Sprint 2* | 10/26/21 | 11/13/21 | 5 | Testing of all functions will be retesting to ensure old code doesn’t conflict. |
| *System testing* | 11/1/21 | 11/13/21 | 4 | With the VScode extension development host this can be done with any test. |
| *Regression testing* | 11/1/21 | 11/13/21 | 4 | With every new hotkey and function other features will need to be tested for accuracy. |
| *Usability Testing* | 11/1/21 | 11/23.21 | 3 | User tests will be conducted |
| *Resolution of final defects and final build testing* | 11/20/21 | 11/27/21 | 4 |  |
| *Deploy to Staging environment* | 11/27/21 | 11/30/21 | 3 |  |
| *Performance testing* | 11/30/21 | 12/1/21 | 5 |  |

| *Release to Production* | 12/1/21 | 12/1/21 | 3 |  |
| --- | --- | --- | --- | --- |
| *Functional testing - Sprint 1 -We Don’t Byte* | 3/20/22 | 3/26/22 | 2 | Testing getLeadingSpaces function |
| *Functional testing - Sprint 2 -We Don’t Byte* | 4/3/22 | 4/9/22 | 2 | Testing getLineNumber |
| *Functional testing - Sprint 3 -We Don’t Byte* | 4/25/22 | 5/7/22 | 4 | Testing Highlight function and customizations |
| *Regression and Integration testing - Sprint 3*  *- We Don’t Byte* | 4/25/22 | 5/10/22 | 3 | Test all existing machinery (Lexer, Parser, commands) for regressions. Ensure the project is just as, or more, stable than when we got it. |
|  |  |  |  |  |
| *Functional testing - Sprint 1 -Five Guys* | 10/25/22 | 11/1/22 | 3 | Testing Speech-to-function, Listing autocomplete options |
| *Functional testing - Sprint 2 -Five Guys* | 11/2/22 | 11/19/22 | 3 | Testing Move Cursor |
| *Functional testing - Sprint 3 -Five Guys* | 11/20/22 | 12/6/22 | 4 | Testing Voice Hotkeys |
| *Performance Testing - Five Guys* | 12/4/22 | 12/6/22 | 4 |  |
| *Regression and Integration testing - Sprint 3*  *- Five Guys* | 12/4/22 | 12/8/22 | 4 | Test all existing work for regressions. Ensure the project is just as, or more, stable than when we got it. |
| *Functional testing – Sprint 4 – Five Guys* | 20/1/23 | 18/2/23 | 3 | Tested all the previous functions, worked on more features for voice hotkeys, and auto-complete. |
| *Regression & integration testing - Sprint 5 - Five Guys* | 19/2/23 | 11/3/23 | 3 | Tested all the previous functions. Autocomplete and voice hotkeys are working fine. Definition/hover is currently accessible for usage. |
| *Performance Testing - Sprint 6 - Five Guys* | 11/3/23 | 4/8/23 | 3 | NLTK Library added, with a few more updates on the extension. |
| *Resolution of final defects and final build testing*  *Sprint 6 - Five Guys* | 4/9/23 | 5/1/23 | 4 |  |
| *Deploy to Staging environment*  *Sprint 6 - Five Guys* | 5/1/23 | 5/6/23 | 5 |  |
| *Functional Testing*  *Hotkey Testing*  *Sprint 2 - KJBK* | 10/31/24 | 11/9/23 | 3 | Testing audio flags functionality and ensuring stability under different conditions. |
| *Regression and Integration Testing - Sprint 6 - Team KJBK* | 3/17/2025 | 3/31/2025 | 4 | Testing all previous group features in coordination and interaction with new features. |