USER SETUP/ SET UP STEPS	**Notes**	**completion**
Run through the Setup Setups and get your project ready to begin work.		
Review the Resources outlined below - be sure to have relevant documentation and references open while you develop!		
Build & Run your Cypress Tests.		
Setup Steps		
1. Using your current Music Library Project, create a separate branch for Testing development.	The music library code had some added pop ups and stuff that made it difficult to test, went back to pre-punch list changes to do testing on	done
If your Music Library project is not able to Add, Search AND Delete, you may choose to use the starter code library.		
(IDEALLY, You should use YOUR project, NOT the starter code.)		
Note: Both the front and back ends are included as they will need to be in sync. Even if you have a working backend but not a working frontend, please use the starter for this project.		
Optional Starter Code Project:		
Music Library Starter		
2. Launch your Music Library Backend Server	needed to do pipenv install first then pipenv shell then flask run	done
3. Install the Cypress testing tool on your front end project.	this hung up first time around, then went by smoothly second time trying to install	done
4. Launch your Music Library Frontend Application	npm start	done
5. In a separate/split terminal Launch the Cypress Testing Tool	npx cypress open	done
6. Create regular commits using a clear and consistent style.	goal was 10 and switched to DEV instead of main at 9	done

TASKS/ USER STORIES	**Notes**	
Main Stories		
(5 points): As a developer, I want to make good, consistent commits.	**should be at**	**done**
(2.5 points): As a developer, I will create a test spec file for Cypress. I can name it music.cy.js or use one of the component names.	**used musci.cy.js**	**done**
(2.5 points): As a developer, I will create a describe block to group my tests, it will have a description that accurately categorizes all of the tests contained within it.	**used describe block for all tests contained**	**done**
(5 points): As a developer, I will add Cypress specific selectors (data-cy, data-test, and or data-testid). One for each of the fields in my “Add” Song form. I will create a test (#1) for this form that adds data to each field and clicks on the “submit” button.	**had to undo the add song confirmation pop up window, add force form to populate on approval to get testing to work.**	**done**
(10 points): As a developer, I will test (#2) the Filter/Search functionality.	**cy.get('select[data-cy="filter-selector"]').focus().select('genre'); cy.get('input[data-cy="filter-input"]').focus().type('Hip-Hop');**	**done**
Using the designed search/filter function to display the added record,		
Hint: You may need to add the .wait(nnn) command to create a delay of a specified number of milliseconds.		
Navigate to the search results in the table		
Assert that the record/data exists.		

(10 points): As a developer, I will create another test (#3) that will delete the record that I just added. I will need to use appropriate commands to navigate the duplicate elements of my table to find the correct row and then trigger the “delete” button or capability currently coded in the application.	in this I had to use a reload. NOTE: need to figure out how to get the changes from testing wiped, currently manually undoing them	done
Bonus Stories		
(2.5 points): As a developer, I will create a test that verifies my Edit page (or modal) by changing the contents of the title field.	edit the title of the song that is on the 9th row': same as above had to manually undo testing, need to figure that out….probably would have come up in next sections if I continued on bonuses.	done
Hint: use the chained .clear() command for retyping in a field with existing data.		
(2.5 points): As a developer, I will create a custom command that creates a record and then refactor the add and filter tests to use the custom command.		
(2.5 points): As a developer, I will create a custom command that “deletes” a record and then refactor the delete and filter tests to use the custom command.		
(2.5 points): As a developer, I will add an API test to verify my add posts and assert that the data has been added.		
END RESULT	Notes	completion