Criterion B: Design

The Chrome Extension

Since the project is a chrome extension it should be explained how chrome extension function in relation to my project. Chrome extensions the basic anatomy of the following:

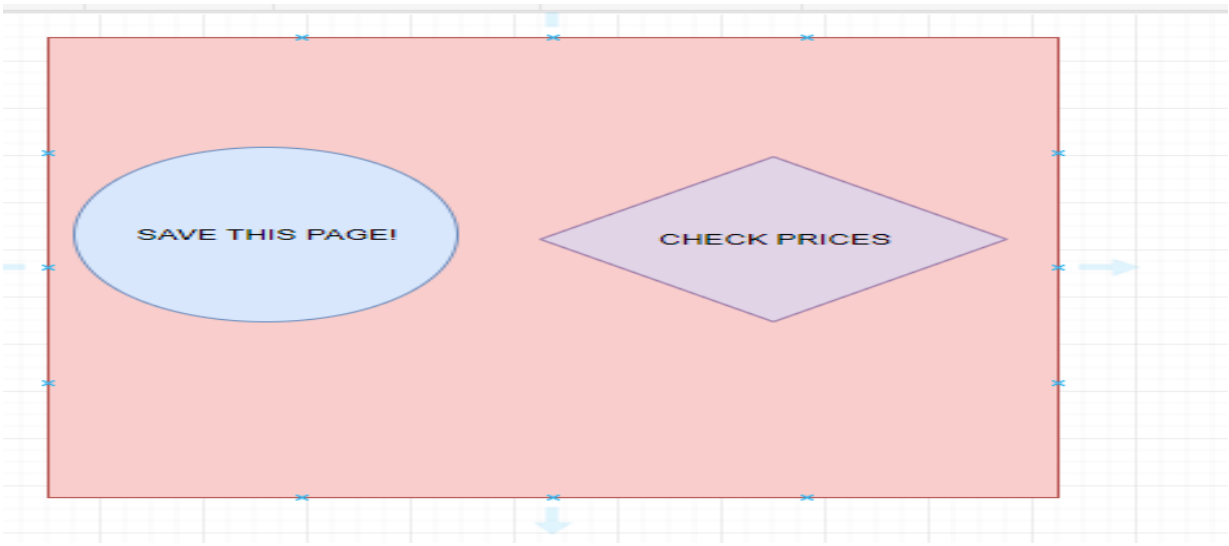
- A "manifest.json" file which specifies which provides the chrome browser with the following information:
    - Which other files are being used in the chrome extension.
    - Which images to use for icons.
    - The version number of the extension.
    - The name of the extension.
    - The permissions the extension will require.
- A background JavaScript file ("background.js") which runs javascript code in the background of the browser.
- A popup file (popup.html) which serves as the user interface for the chrome extension. When the user clicks on the icon of the extension the popup.html file is provided.
- A style sheet (style.css) file for the popup.html for changing the user interface.
- An external JavaScript file for the popup file ("popup.js") this provides interactive programming elements which can communicate with the background.js file.

Techniques That Will Be Used:

- **XMLHTTPRequests:** This is the default built-in object used in JavaScript to request HTML files on the web. This will be used to extract the HTML page off of the Amazon Website.
- **DOMParser:** This is another built-in object in JavaScript which allows for the parsing of HTML files. While normally used for parsing tags off of a local html file it can also be used to parse tags from a html file fetched from a request.
- **Chrome.storage.sync:** This is the Chrome Extension API for saving data within the browser across browsers. This will allow the application to save URLs of Amazon Pages and then access them from any chrome browser the user is logged into.
- **Document Object Model (DOM):** While technically used in the DOMParser already, this will be more explicitly referenced when adding functionality to the buttons for the user interface of the program. This will allow me to attached event listeners to HTML elements in order to actually have a working user interface.
- **chrome.tabs.getCurrent()**: This is another built-in chrome extension method for retrieving the URL of the current tab. This will be used to check if a user is actually on an Amazon web page.
- **chrome.browserAction.setBadgeText():** This is will be the method used to actually notify the user of new deals. This will display notifications via showing the number of new deals on the extension icon.

The User Interface:

In order to meet the first success criteria outlined in section A I will need a clearly designed application. This will be done by keeping the design as simple as possible in order to make sure the client can operate the extension.



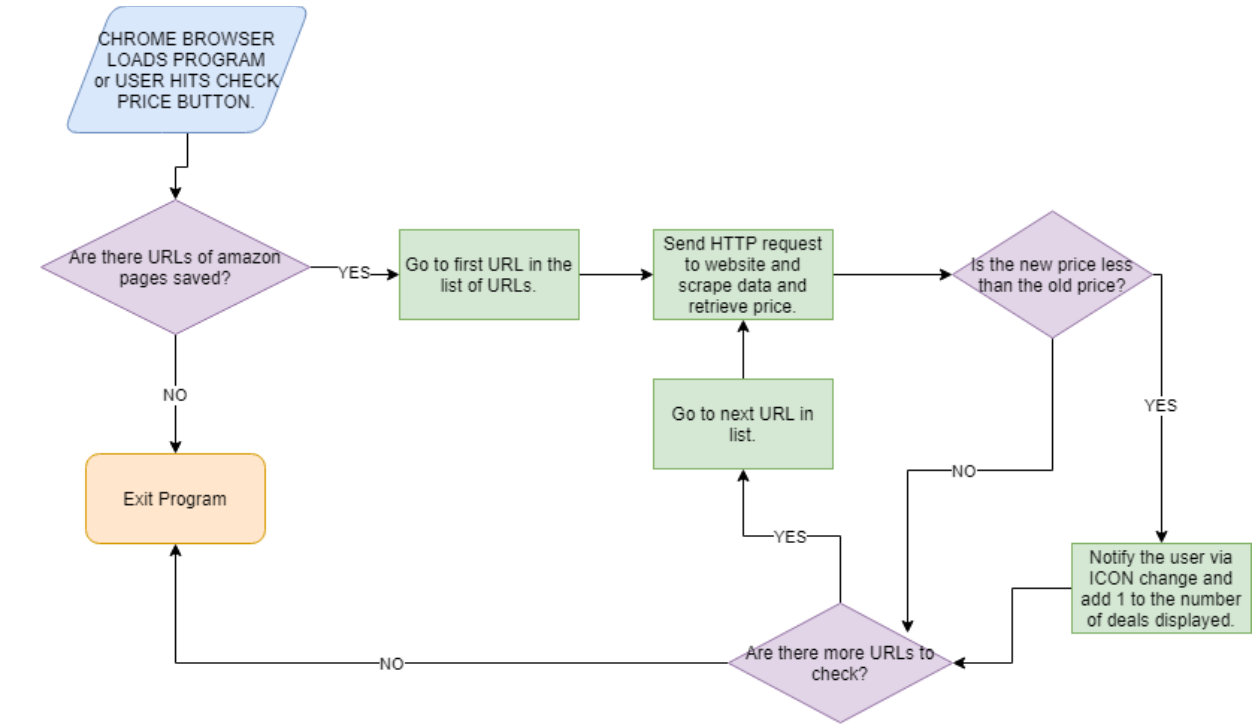
The Algorithms

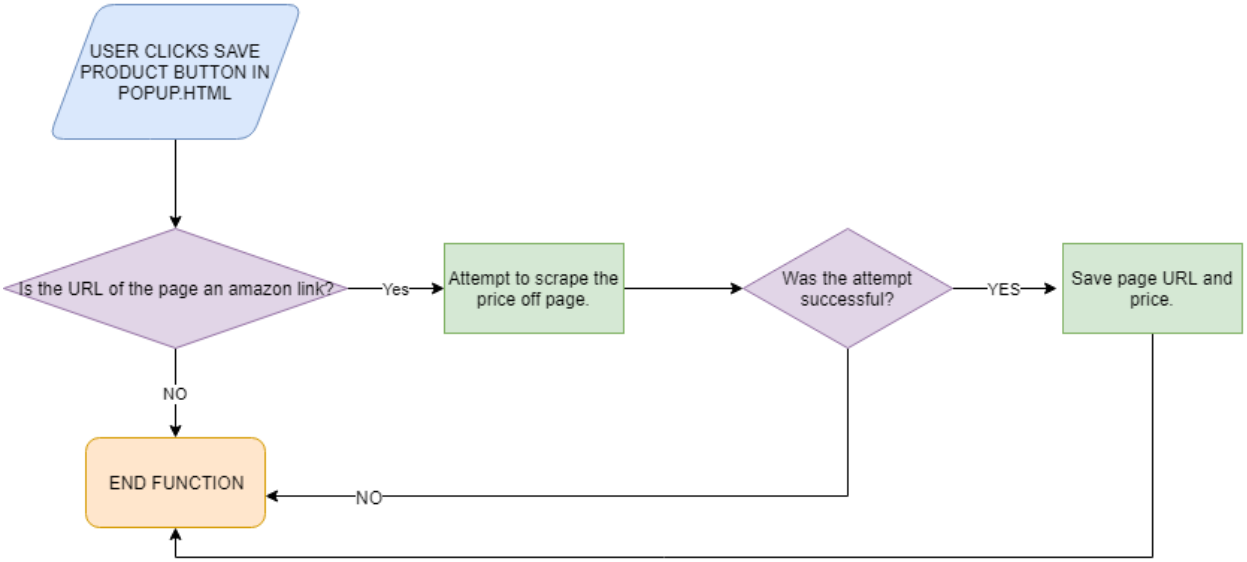
There are two main logical portions of the chrome extension which I will be building in JavaScript. The user interface will be done in HTML/CSS which doesn't use an algorithm to display information. As such the only main application of a flowchart would be the JavaScript portion of the extension.

The extension has two main functions:

- Saving Amazon product pages with the price.
- Check Amazon product pages for a new price and comparing it with the old price.

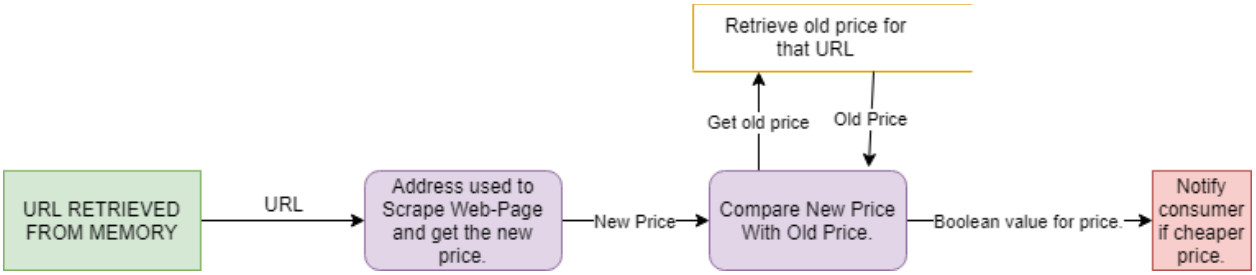
The flowcharts below showcase the algorithms that will be used in those two functions.


CHROME BROWSER LOADS PROGRAM or USER HITS CHECK PRICE BUTTON.
Are there URLs of amazon pages saved?
YES
Go to first URL in the list of URLs.
Send HTTP request to website and scrape data and retrieve price.
Is the new price less than the old price?
YES
NO
Exit Program
Go to next URL in list.
NO
YES
Notify the user via ICON change and add 1 to the number of deals displayed.
Are there more URLs to check?
NO


USER CLICKS SAVE PRODUCT BUTTON IN POPUP.HTML
Is the URL of the page an amazon link?
Yes
Attempt to scrape the price off page.
Was the attempt successful?
YES
Save page URL and price.
NO
END FUNCTION
NO

Data Flow Diagram.

I have also charted the flow of data. For this program that is going from the URL of the product page to getting a boolean value if the price has changed. The program gets the URL from memory via chrome.storage and then scrapes the webpage using the URL and getting the new price which is then compared with the old price retrieved from memory. Finally the program checks to see if the new price is greater or less than the new price and sends a boolean value to the output.



Testing Success Criteria From Criteria A

Success Criteria	Test
The client can save products from viewing an Amazon page.	The client can easily find the save button on the chrome extension. Tested in Evaluation.
The application save save the product page URLs across browsers as well as previously found price.	Tested by loading extension on the same account across multiple computers using the same browser.
The product can scrape the price of the product off of an Amazon product page.	Price scrapped is printed in console then compared with the price on the actual web page.
The application can notify the client of price changes in desired products.	The icon of the chrome extension can change.
The application can check product price on chrome browser start up.	The price of the product is printed in the console on extension loading as to insure that the price is actually checked.
The application can compare the new price with the old price.	The outcome is printed from the test. Either, “too expensive” or “too cheap”.