**Grasp Principles Used**

ConsumIt features low coupling in the core data model, as is typical with consumer-facing systems. The model only has one many-to-many relationship meaning all three of our core objects are mostly independent of each other. As of this iteration, we have fairly low cohesion on the front-end. That will be a goal for the next iteration. We definitely practice the Information Expert principle; the Consumption object is the information expert as it contains a User, a Consumable, and the amount of time it took for the user to consume that consumable. It knows the majority of the important knowledge. We practice the Creator principal through our REST API. All objects are created through that layer. We practice the Controller principal through the front-end by communicating from our front-end to our data-layer through a careful javascript environment.

popup.js

saveLinks(clickEvent)

Selects the current window and calls saveLink(currentUrl, callback)

saveLink(url, callback)

The callback is optional. Stores the url given.

showLinks(clickEvent)

Modifies the popup window to include a list of all the saved urls

hideLinks(clickEvent)

Returns the popup window to its initial state

clearConsumables()

Clears all saved consumables

stripFragment()

Removes the fragment from a url. Used so google.com/#VGS is the same as google.com/

eventListeners

saveLink - When the save link element is clicked the saveLinks function is called

viewLink - When the view link element is clicked the showLinks functions is called

clearLink - When the clear link element is clicked the clearConsumables function is called

clickMenu.js

This is the script that runs in the background while the extension is active. It has two uses currently. It contains many of the same functions as popup.js

1. Right click menu

Modifies the right click menu in Chrome to include a consume later button when a link is clicked. The consume later button calls the saveLink function from above with the url of the link that was clicked

2. Timer script

This script runs in the background. Whenever a page saved as a consumable is navigated to the script will start a timer for that page. When the user leaves the page (through switching tabs, going to a new page, closing Chrome, etc) the timer will stop and report how long was spent on the page (in the background console). The time is currently reset after each session. This is not involved in the use cases being presented currently, so it is not yet complete.

Future changes

The time spent on each page will be stored permanently and once the user marks a page as complete it will report that time to the server.

The user will be able to mark a page as complete.

There will be a consumable management page which allows the user to easily delete and organize consumables.

There will be an estimated time for each page which reports the average time it took to consume a link.

There will be a timeout to check if the user has left a page open and is no longer active.

The user will be able to specify an amount of time the have and the extension will suggest articles around that time.

There will be some default value for pages that have not yet been reported by other users.