Bookmarks Extended - Browser Extention

Synopsis

I started this project as a fork of https://github.com/KM-Bookmarks/bookm-parser but got rid of all of it because after instaling rust, your code didnt even work for my system for some reason (possibly the last real commit being 7 years ago had something to do with it...) Python for life 😎

This is a bookmarks extender extension. The idea being that it helps autocategorise newly added bookmarks by replacing the current menu with a new menu that uses AI to generate a few of the most likely categories at the top and a search through your folders button for more preciseness if the suggestions weren't good enough.

Table Of Contents

What am I trying to do exactly?

Background / where I wanted to go at the start

I thought bookmarks would only save 2 seconds of my time each search, slightly useful but most of my searches on the web were using a search engine anyways and saw no point to chaging a system that works for most part, However now that I am a student I am finding it no longer feasible to memorise all the websites for when I need them, I decided to spend a solid 6 hours just organising bookmarks into tabbed folders with subcategories and I now realise how difficult of a task it is to just categorise bookmarks as most of them are edge cases or would be appropriate to fit in multiple folders.

I am currently using NightTab which is a new tab replacer that is populated with bookmarks that you have to add manually. I have configured my nightTab to look really clean and modern and I have reached the point where I have about ~200 bookmarks in nightTab and ~700 bookmarks in firefox. Both of which can be exported, with firefox being in html and nightTab in json. Originally I wanted to just be able to merge the two

together and reinterporate them back into json so I could have all my bookmarks in one spot. However I am now realising the potential improvments through digging into the existing bookmark managing tools I currently use.

What I'm doing now

I have been working on the project for about a week and I have decided there are improvments to do on both of the bookmarking tools. Firstly I do not want to have to manually folder each item when I bookmark, I want to be prompted a few autosuggestions for where the bookmark could go for example, youtube music videos will be prompted into the music categories, the types of music the user is most likely to be listening to will appear at the top ect ect.

I also want to either fork nightTab or make my own newtab web extension that reads and exports to csv and has the ability to import bookmarks from firefox/google/browser and all the other types.

For the autosuggestions to happen I will need a categorisation model and im really not sure how fast this whole project will be... Might need my gamging pc to act as a remote sever to be able to do the machine learning calculations quickly but I will do everything in my power to optimise it with minimal compromise to model accuracy.

```
I never understood why people documented code until I went back to try continue old
projects after completley forgetting what I was trying to achieve. Please document
your code cuz its really hard to know what your trynna achieve or do with your
code. The ammount of repos ive seen with absolutley nothing just piss me off now.
```

How to use this repo in its curlerent unusable state.

So far the only bits that are working are the two convertion python files (browser bookmarks to csv and nighttab browser extention to csv). they are located in the src/components/... and are all contained in functions. for example

```
cd src/components
```

```
python3 nighttabjson_to_csv.py
parse('inputfilelocation+extention','outputfolder+extension')
```

What each file Is...

- main.py is the main python file, once the whole project is done, this should be the only nessecary file.
- all the other files ending in .py will be used in that main.py file, if one component their names may give hints to what needs debugging.
- the bookmarks.csv is the bookmarks.html cleaned and turned to csv in one.
- the night_tab.csv is the night_tab.json cleaned and turned to csv in one.

This *README.md* is so people like me and you can remember what the fuck this whole thing means when we move onto another project and forget about all of this.

Unessesary Added Complexities I decided I will be using an ai model to not only categorise the bookmarks and give appropriate quick suggestions but also to help generate favicons. I may also need a icon library such as fontawesome to use in combination.

My only worry currently is that it does not fit everything well enough, like youtube links will probably all come under visual / entertainment even if their content can be drastically different depending on the video

Website new tab override

documentation

Python script collection (conversions)

Steps

Tasks:

1. Python script to convert bookmarks.html to csv:
2. Python script to convert bookmarks.json to csv:
3. Python script to merge the bookmarks.csv files created in step 1 and step 2:
4. Python script to convert the merged CSV file (created in step 3) back to bookmarks.html:
5. Python script to convert the merged CSV file (created in step 3) back to bookmarks.json:

Cleaned CSV files to website

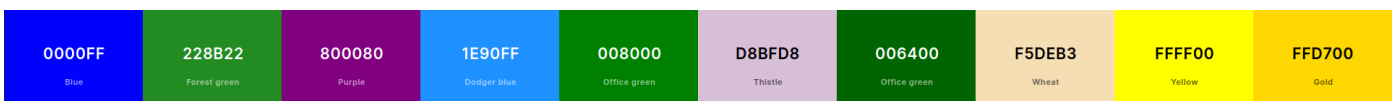
Turning the csv generated from the process before (firefox bookmarks to csv and nighttab bookmarks to csv) and using those combined csv files as data to flow into a webpage as their bookmarks under their categories.

For the website I want a focus to be on integrating the python script into it and being able to edit how folders look and interact. With a focus on the data handling and how users can import more data if they wish and formatting data themselves as they please.

Possible API integrations

- https://github.com/RealFaviconGenerator/rfg-api
- https://realfavicongenerator.net/
- https://github.com/domantasm96/URL-categorization-using-machine-learning
- Machine Learning model to categorise a url into its category

Color Theory



(Generated From Coolors)

```
The background color is #0000FF (blue)
The background color is #228B22 (forest green)
The background color is #800080 (purple)
The background color is #1E90FF (dodger blue)
The background color is #008000 (green)
```

```
The background color is #D8BFD8 (thistle)
The background color is #006400 (dark green)
The background color is #F5DEB3 (wheat)
The background color is #FFFF00 (yellow)
The background color is #FFD700 (gold)
```

(Generated From ChatGPT)

Steps

Steps required to create a new tab web extension:

1. Plan your extension: Define the purpose and functionality of your extension. Determine what it will do and what features you need to build.
2. Set up your development environment: Install the necessary software tools, such as a code editor and browser extensions development kit.
3. Create a new extension: Create a new folder for your extension and add a manifest file that specifies the name, version, permissions, and other details of your extension.
4. Design the user interface: Create HTML, CSS, and JavaScript files to define the user interface and functionality of your extension.
5. Set up event listeners: Define event listeners in your extension's JavaScript code to handle user interactions with the user interface.
6. Implement the functionality: Write the JavaScript code to implement the functionality of your extension.
7. Test your extension: Test your extension on different devices and browsers to ensure that it works as intended.
8. Publish your extension: Publish your extension on the Chrome Web Store, Firefox Add-ons, or other extension marketplaces, or distribute it directly to users.
9. Maintain and update your extension: Monitor your extension's performance, fix any bugs or issues that arise, and update it with new features and improvements as needed.

Machine Learning Model

steps

To categorize URLs into broad and specialized categories, you can use a supervised machine learning algorithm such as a classification algorithm. Here are the steps you can follow:

1. Data Cleaning:
    - Clean the URLs to remove any irrelevant information (e.g. query strings).
    - Extract features from the URLs that can be used for classification. For example, you can extract the domain name, the length of the URL, the presence of certain keywords in the URL, etc.
    - You can also use external data sources to enrich your dataset, such as adding metadata about the URLs (e.g. from an API that provides information about web pages).
2. Labeling:
    - Label each URL in your dataset with the corresponding broad and specialized categories. You can do this manually or use an existing dataset as a reference.
3. Training:
    - Split your dataset into training and test sets.
    - Train a supervised classification algorithm (e.g. logistic regression, random forest, support vector machine, etc.) on the training set using the features and labels you extracted in steps 1 and 2.
    - Evaluate the performance of the trained algorithm on the test set. You can use metrics such as accuracy, precision, recall, and F1-score to evaluate the performance.
4. Prediction:
    - Use the trained algorithm to predict the categories of new URLs.
    - You can also use the feature importance scores to identify the most important features for classification.
5. Refinement:
    - Refine the algorithm by tuning the hyperparameters and retraining on the entire dataset.

- You can also consider using techniques such as cross-validation and ensemble methods to improve the performance of the algorithm.

6. Deployment:
    - Deploy the algorithm to categorize new URLs in real-time.

and its basically as easy as that...

Big List :)

1. Python script to convert bookmarks.html to csv:

    a. such as Beautiful Soup and CSV.

    b. Open the bookmarks.html file using Beautiful Soup and parse the data.

    c. Create a CSV file and write the parsed data into it.

    d. Save the CSV file and close it.

2. Python script to convert bookmarks.json to csv:

    a. Import the necessary libraries, such as JSON and CSV.

    b. Open the bookmarks.json file and load the data into a Python object.

    c. Create a CSV file and write the data from the Python object into it.

    d. Save the CSV file and close it.

3. Python script to merge the bookmarks.csv files created in step 1 and step 2:

    a. Import the necessary libraries, such as CSV.

    b. Open the bookmarks.html CSV and bookmarks.json CSV files and read their data into Python objects.

    c. Combine the data from the two Python objects into a single Python object.

    d. Create a new CSV file and write the merged data into it.

    e. Save the CSV file and close it.

4. Python script to convert the merged CSV file (created in step 3) back to bookmarks.html:

   a. Import the necessary libraries, such as CSV and BeautifulSoup.

   b. Open the merged CSV file and read the data into a Python object.

   c. Create a new bookmarks.html file using BeautifulSoup and write the data from the Python object into it.

   d. Save the bookmarks.html file and close it.

5. Python script to convert the merged CSV file (created in step 3) back to bookmarks.json:

   a. Import the necessary libraries, such as CSV and JSON.

   b. Open the merged CSV file and read the data into a Python object.

   c. Convert the Python object into a JSON object.

   d. Create a new bookmarks.json file and write the JSON object into it.

   e. Save the bookmarks.json file and close it.

6. Machine Learning Model

   b. Clean the URLs to remove any irrelevant information (e.g. query strings).

   c. Extract features from the URLs that can be used for classification. For example, you can extract the domain name, the length of the URL, the presence of certain keywords in the URL, etc.

   d. You can also use external data sources to enrich your dataset, such as adding metadata about the URLs (e.g. from an API that provides information about web pages).

7. Website

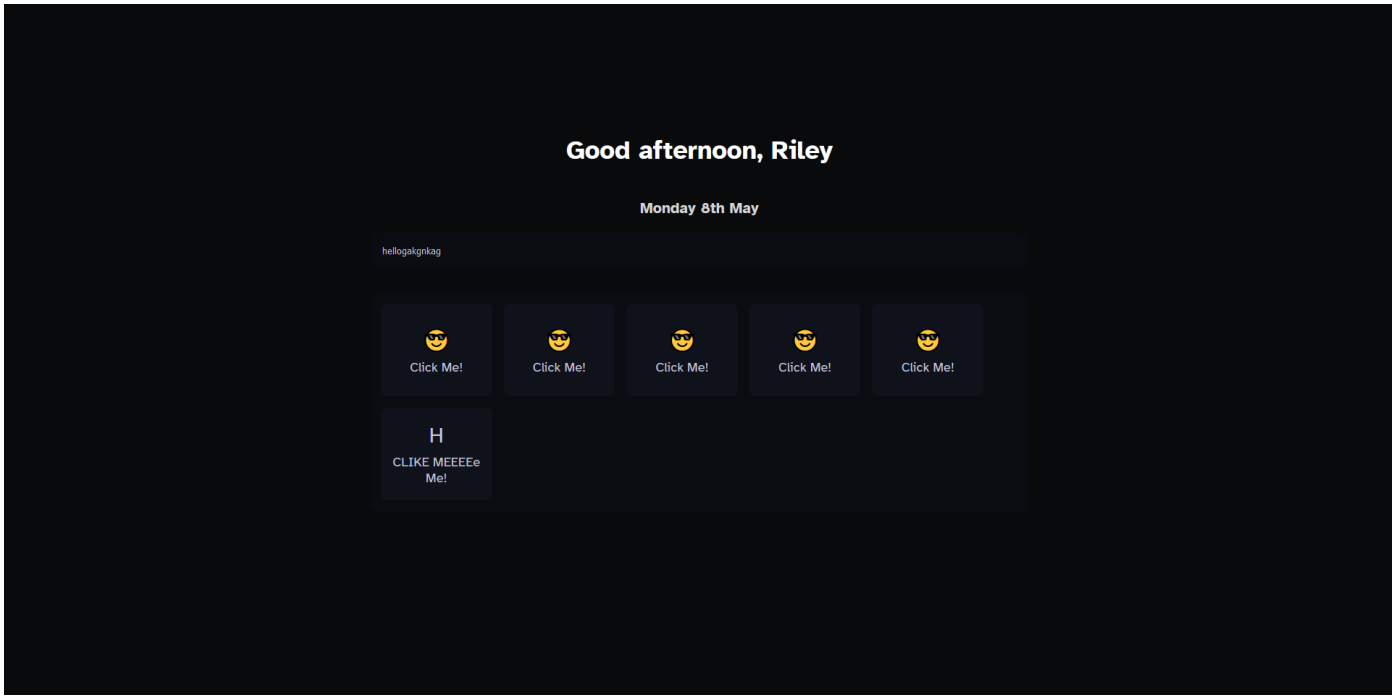
The Project, Disected & timed out

Task	Estimated Hours	Assignee	Status
Yeah, This Might Take Me A Solid Year To Finish			
1. python script that converts Bookmark.html > bookmark.csv	~7 hours	Riley Smith	✔
2. python script that converts Bookmark.json > bookmark.csv	~8 hours	Riley Smith	✔
3. python script that merges the previous two csv files into one	~1 hours	Riley Smith	Not Started
4. Reverse step 1.	~3 hours	Riley Smith	Not Started
5. Reverse step 2.	~2 hours	Riley Smith	Not Started
6. Machine Learning Model 1	~12 hours	Riley Smith	Not Started
6. Machine Learning Model 2	~12 hours	Riley Smith	Not Started
6. Machine Learning Model 3	~12 hours	Riley Smith	Not Started
7.Basic website - **First Release**	~2 hours	Riley Smith	Not Started
8.Website improvements - **Second Release**	>7 hours	Riley Smith	Not Started

PROJECT LOG

this is a log for my progress, intended for just me but can be useful if you want to contribute or just know how the project is going and where its headed. time format dd/mm/yy because im no psyco.

12/5/23



Overview

This seems like a good spot to finish up for the night. I made the website look nicer and cleaned up the readme so it looks a bit easier to read. Next time I will probably be setting up react as it has been all fine and dandy making the website look nice but its going to need linked lists and data structures and folders and that seems near impossible to do in vanilla js and I know what im doing with react so it shouldnt be too hard.

Next Steps

- Implement React
- Date and time fetcher
- connect the config file to css variables + some config for date time formats and such?
- Implement Foldered data from a csv
- Searchbar? - at least decide what you will be doing when it comes to searching for links and keywords and such
- Does the json to csv need fixed or was csv not the best idea for foldered data in the first place? offer alternative solutions

14/5/23

Overview

Big one is that I have gone from the boilerplate from the previous log that was written in plain html, js and css into react which should handle procedurally generating the bookmarks and it integrates with node a bit nicer than plain html, css js.

Converted from .conf to .yml just because I like yaml more and It looks like nested items get more readable (https://yaml.org/spec/1.2.2/#example-compact-nested-mapping) which I want because I want to use yaml instead of csv as the final export. It is a smaller file than json and html so it should mean the bookmarks load faster anyways so I see no issue with converting... yet...

css colour variables are now stored in the .yml (YAML) file, have not been connected with css variables yet.

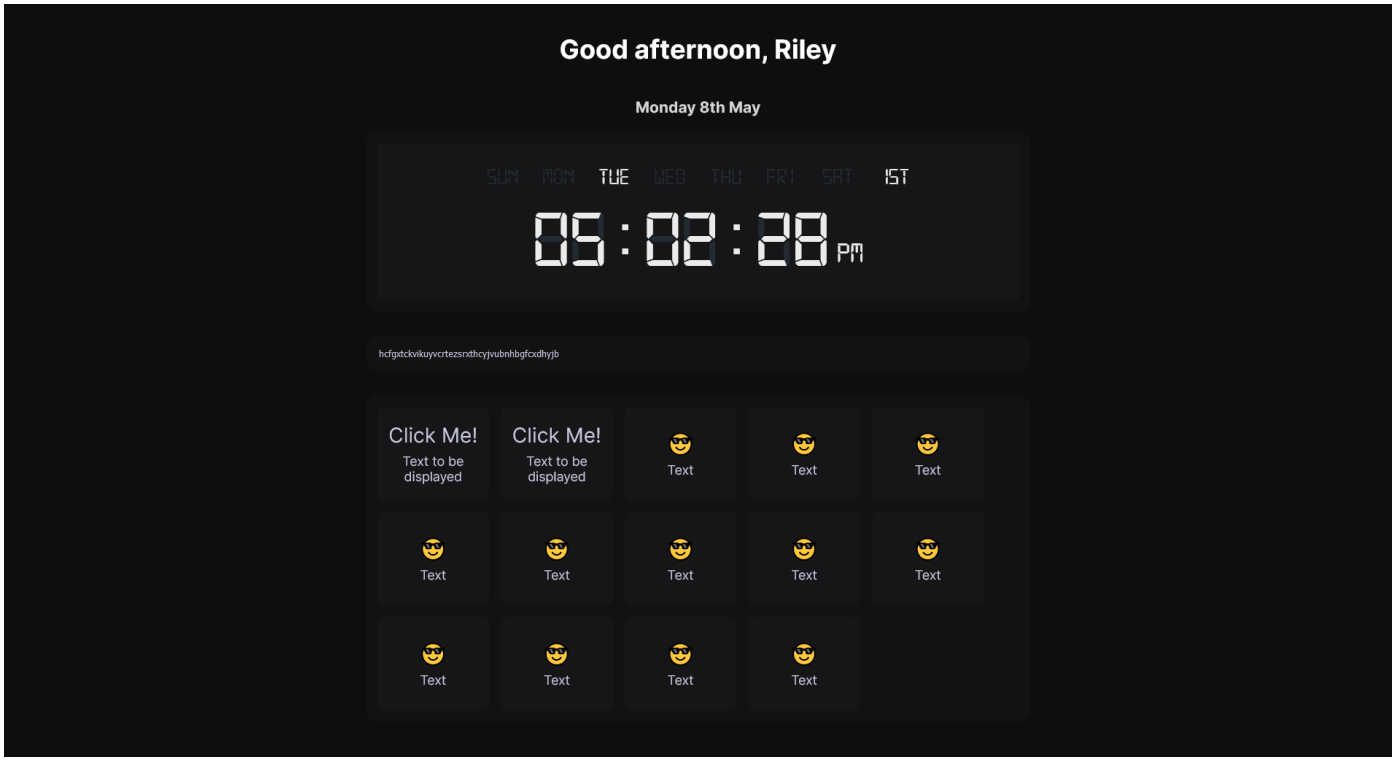
website now has its own seperated documentation

Next Steps

- Remake orginal website in nextjs
- Date and time fetcher
- connect the config file to css variables + some config for date time formats and such?
- Implement Foldered data from a csv
- Searchbar? - at least decide what you will be doing when it comes to searching for links and keywords and such
- Does the json to csv need fixed or was csv not the best idea for foldered data in the first place? offer alternative solutions

16/5/23

Overview




```
+ Next JS App Converted
+ Clock Added
+ Backend called
```

Useful Youtube tutorials used

- Fireship - Next.js 13 - The Basics https://www.youtube.com/watch?v=__mSgDEOyv8&t=248s
- Alex Eagleson - How to Build Scalable Architecture for your Next.js Project https://www.youtube.com/watch?v=Iu5aZDqZt8E&t=3214s
- Alex seems very competent and good at setting up projects for future proofing and git management, the video is just outdated and some of the imports broke nextjs.

Took inspiration from: https://github.com/sergiss/react-clock once everything up and running will add some bool config options such as 24h clock, date format ect...

Next Steps

- **git autorunner script so I cant upload to github without code working properly, USE *HUSKY***
- Learn how pocketbase works and integrates with users. Accounts, Config save, Bookmarks backups ect.

- connect the config file to css variables + some config for date time formats and such?
- Implement Foldered data from a csv
- Searchbar? - at least decide what you will be doing when it comes to searching for links and keywords and such
- Does the json to csv need fixed or was csv not the best idea for foldered data in the first place? offer alternative solutions

20/5/23

Overview

Not alot done today, fixed a bug in which the clock component wasnt loading clientside.

22/5/23

Plan for today

Today I will be setting up Husky with my nextjs app. this will only allow working code to be pushed to the git repo

Overview

...

Next Steps

...