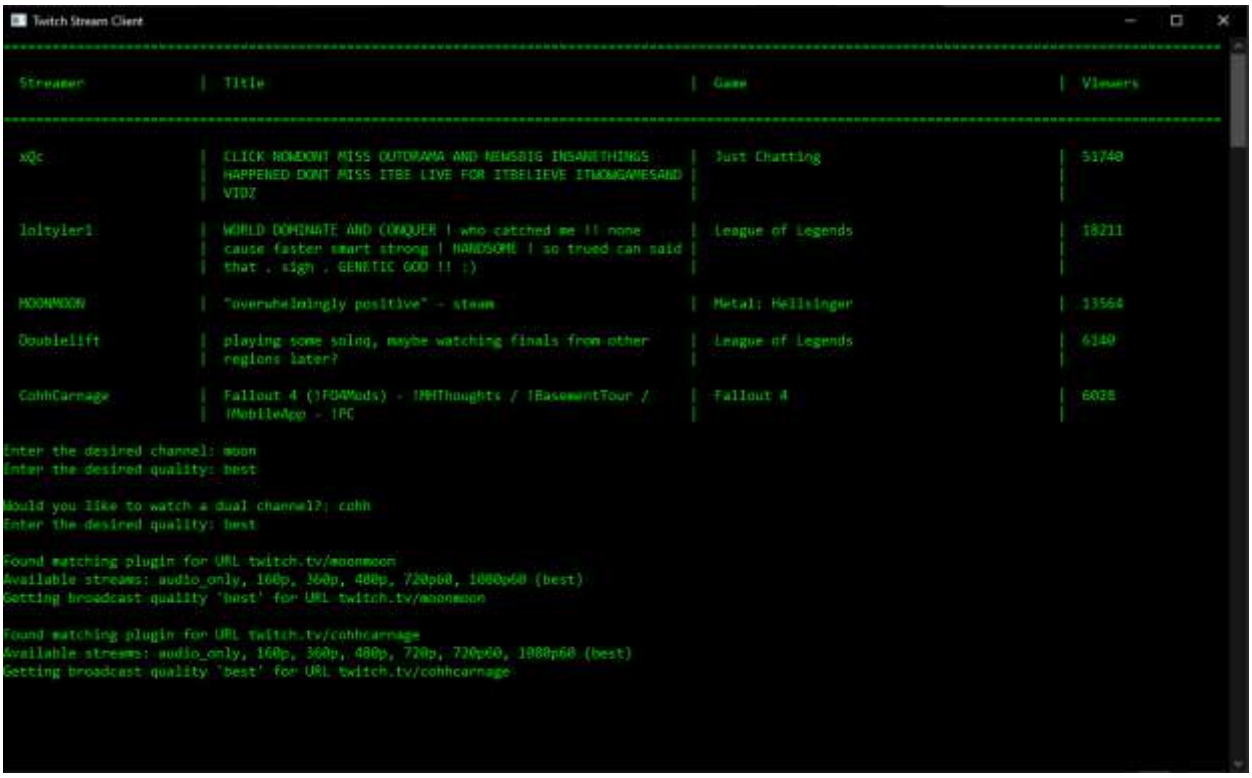
PYTHON PROJECTS

Daniel Tran

Table of Contents

Custom Twitch.tv Client



Project Description

Implements a customized client for viewing livestreams from Twitch.tv.

An instance of the client supports simultaneously viewing up to 4 livestreams with 3 viewing settings:

1. a single stream with its chatroom
2. two streams with their respective chatrooms
3. four streams

Furthermore, the project supports viewing multiple instances of a client over multiple monitors (up to 3 monitors). Therefore, **a maximum of 12 livestreams** can be viewed simultaneously.

Motivation

At the initial time of project conception, Twitch.tv did not offer a method to view multiple livestreams simultaneously.

Thus, one would have to commonly choose one of the following compromises:

1. Open multiple browser tabs but only focus on one, potentially missing entertaining segments of the other livestreams.
2. Open multiple browser tabs and manually organize the windows but also having to endure the tedium, especially as the number of livestreams scale.
3. Open only one browser tab but forgo all other livestreams, completely unaware of any entertaining segments of the other livestreams.

In this regard, I conceived of the idea of a customized client, capable of viewing multiple livestreams simultaneously and being controlled effortlessly through hotkeys.

Design Goals

- The client must be capable of viewing multiple livestreams simultaneously.
- Hotkeys should be implemented to facilitate quick control of the client.

Benefits

- **Circumvents ads from Twitch.tv by leveraging the `streamlink` module**
- Integrates with my personal Twitch.tv account
- Minimizes footprint, compared to a web browser, by embedding the mpv media player into the Qt GUI
- Provides quick control via hotkeys, removing the need to resize or move the client by dragging
- Allows for the alerting of livestreams under special conditions, such as when viewer count hits a desired threshold, usually indicating that a special event is occurring

Languages

1. Python 3.8
2. AutoHotkey 1.1

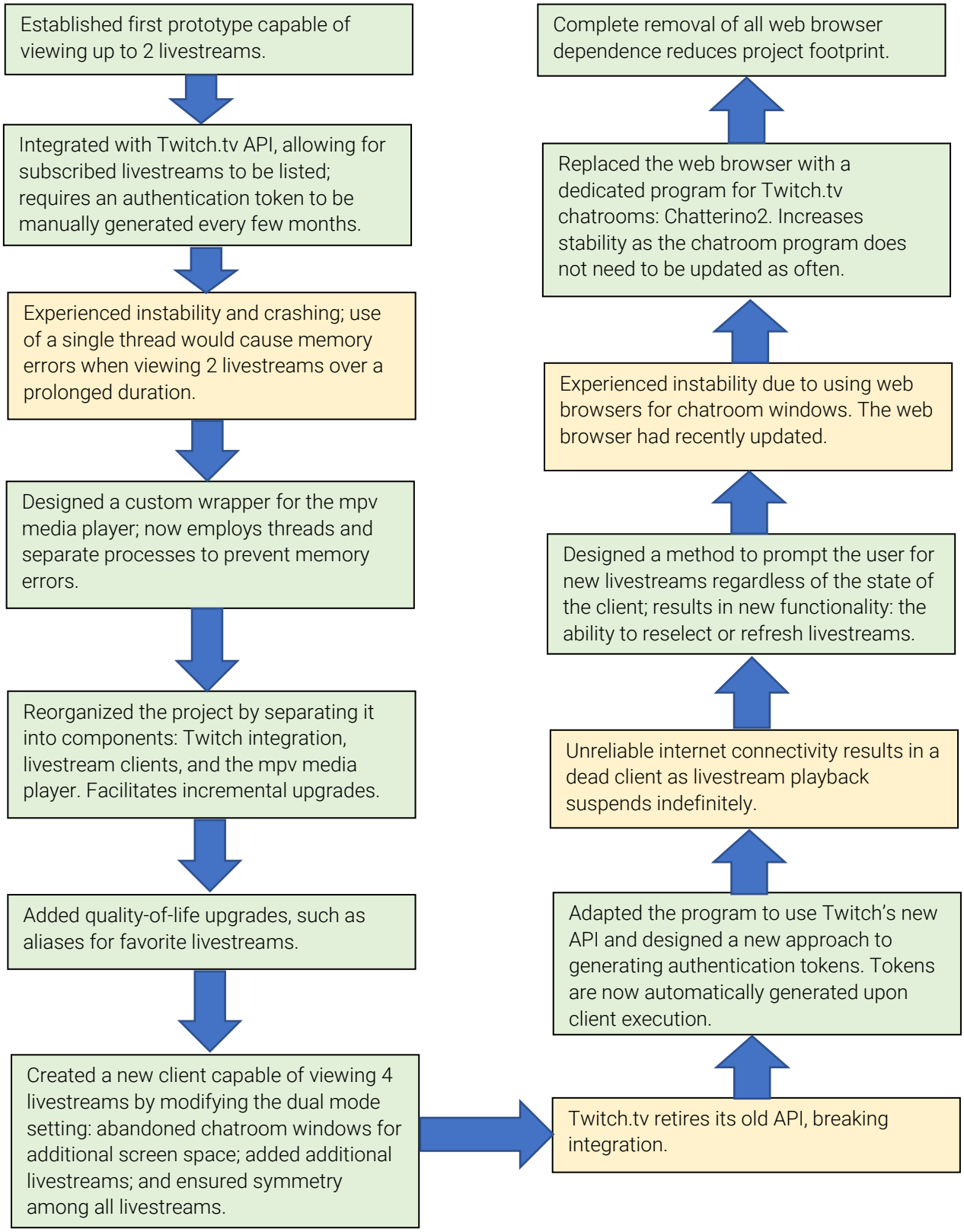
Python Dependencies

1. requests
2. streamlink
3. psutil
4. pyqt6
5. python-mpv
6. python-twitch-client

External Dependencies

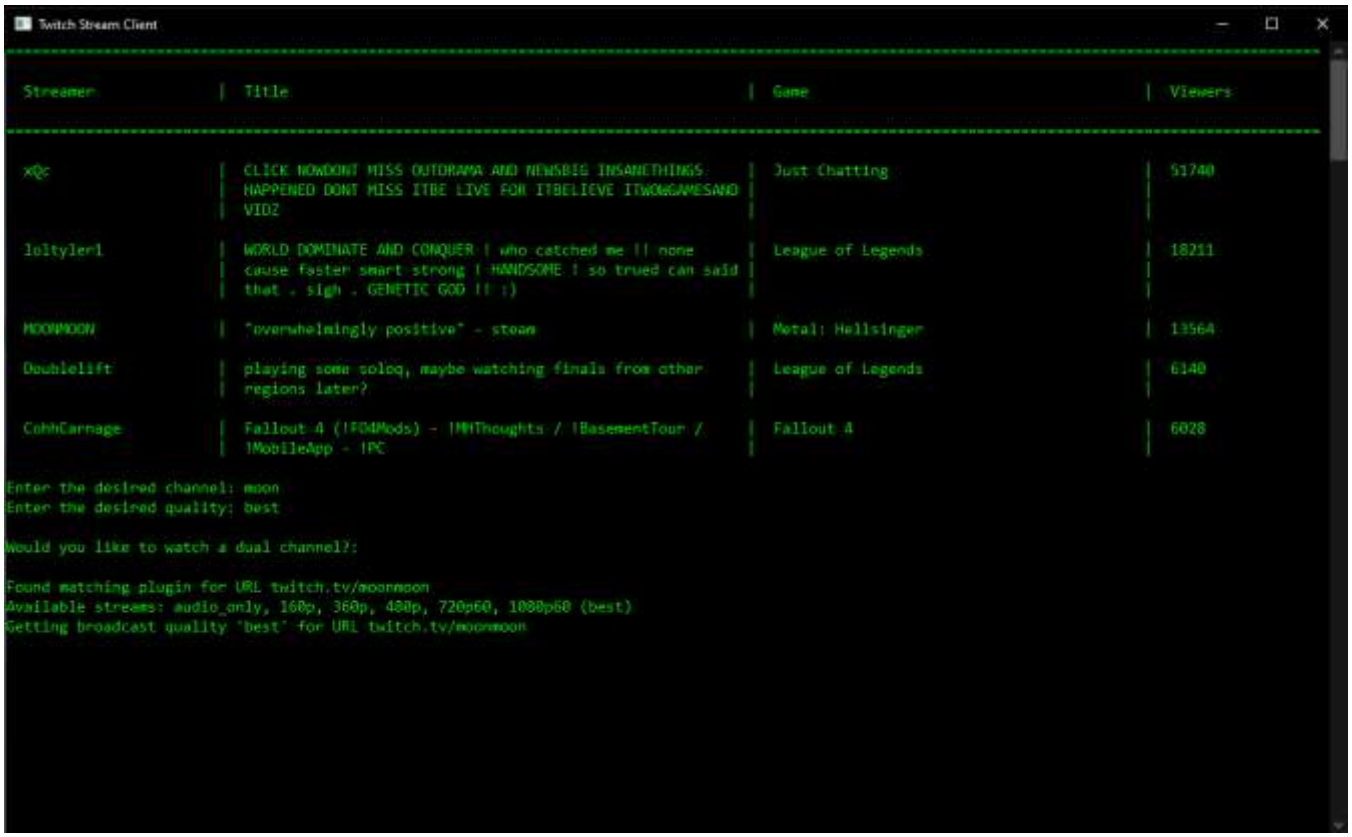
1. Libmpv
2. Chatterino 2

Notable Milestones & Roadblocks



Features

1. Selection of a desired or subscribed streamer



The project employs the API of Twitch.tv to list online streamers in which my personal account is subscribed to. The streamers are listed within the command prompt and the user is asked to select valid livestream(s). A valid livestream is any livestream that is currently online on Twitch.tv.

2. Single Livestream Mode

An instance of the client in viewing one livestream with its corresponding chatroom.

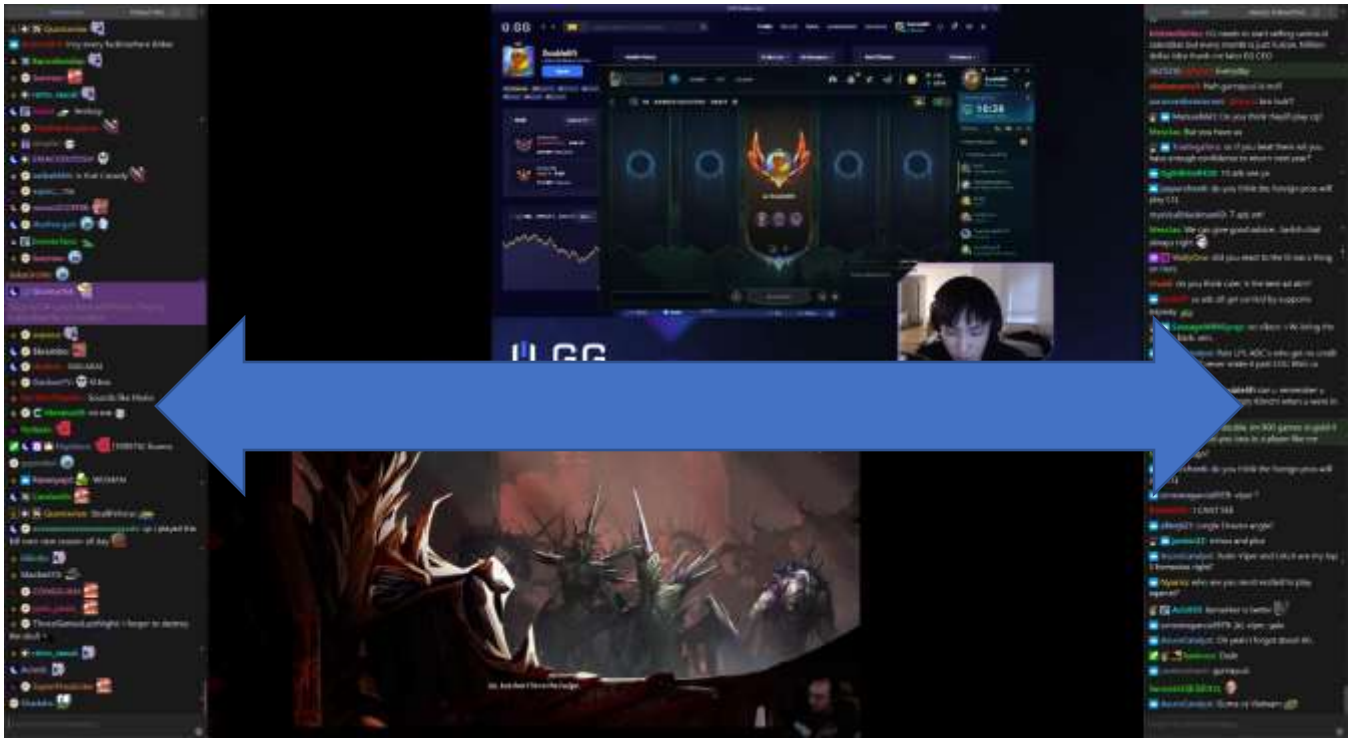
3. Dual Livestream Mode

An instance of the client in viewing two livestreams with their corresponding chatrooms.

4. Volume Control of Individual Livestreams (Any Mode)

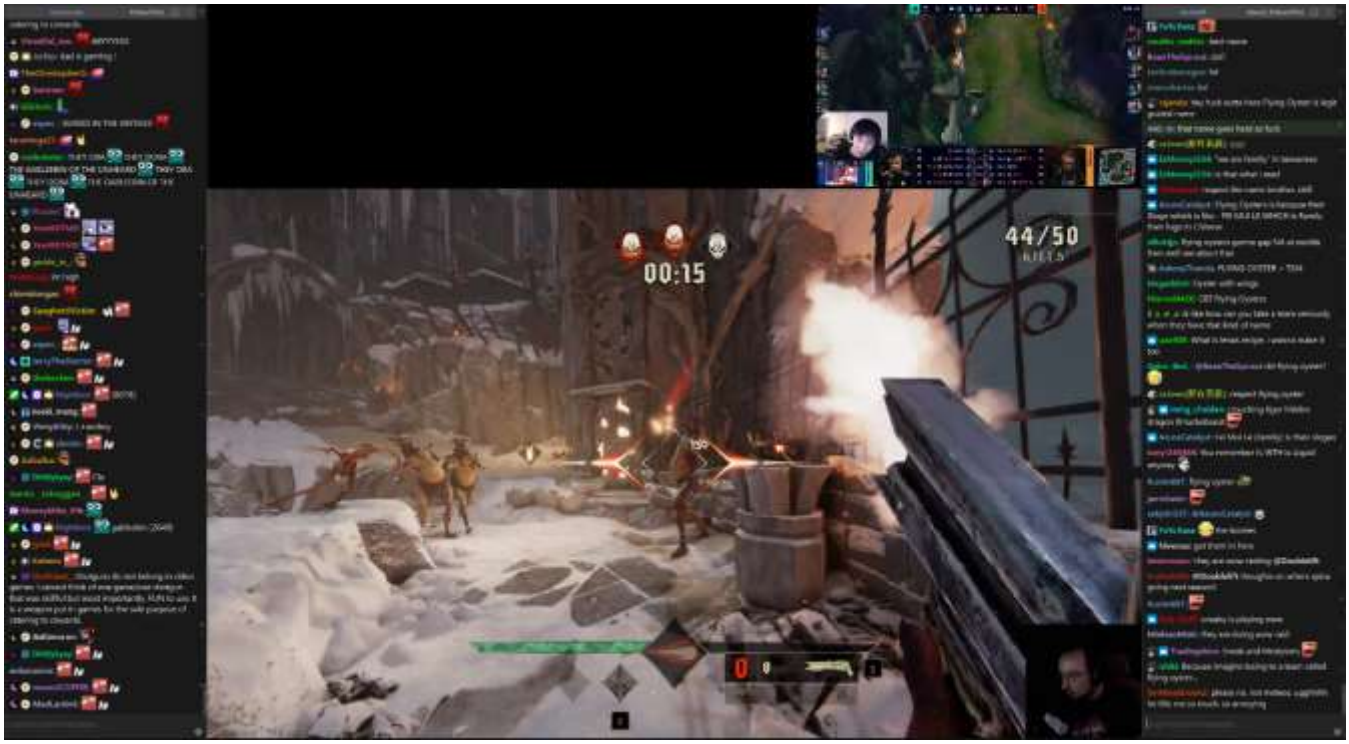
The volume of any livestream can be individually controlled regardless of the number of livestreams being viewed.

5. Swapping of Chatroom Windows (Dual Livestream)



The chatroom windows can be swapped during the dual livestream mode.

6. Zooming of Livestreams (Dual Livestream)



Clicking on a livestream will enlarge it and shrink the other livestream during the dual livestream mode.

7. Flipping of Livestreams and Chatroom Windows (Dual Livestream)

The livestreams and chatroom windows can be flipped during the dual livestream mode.

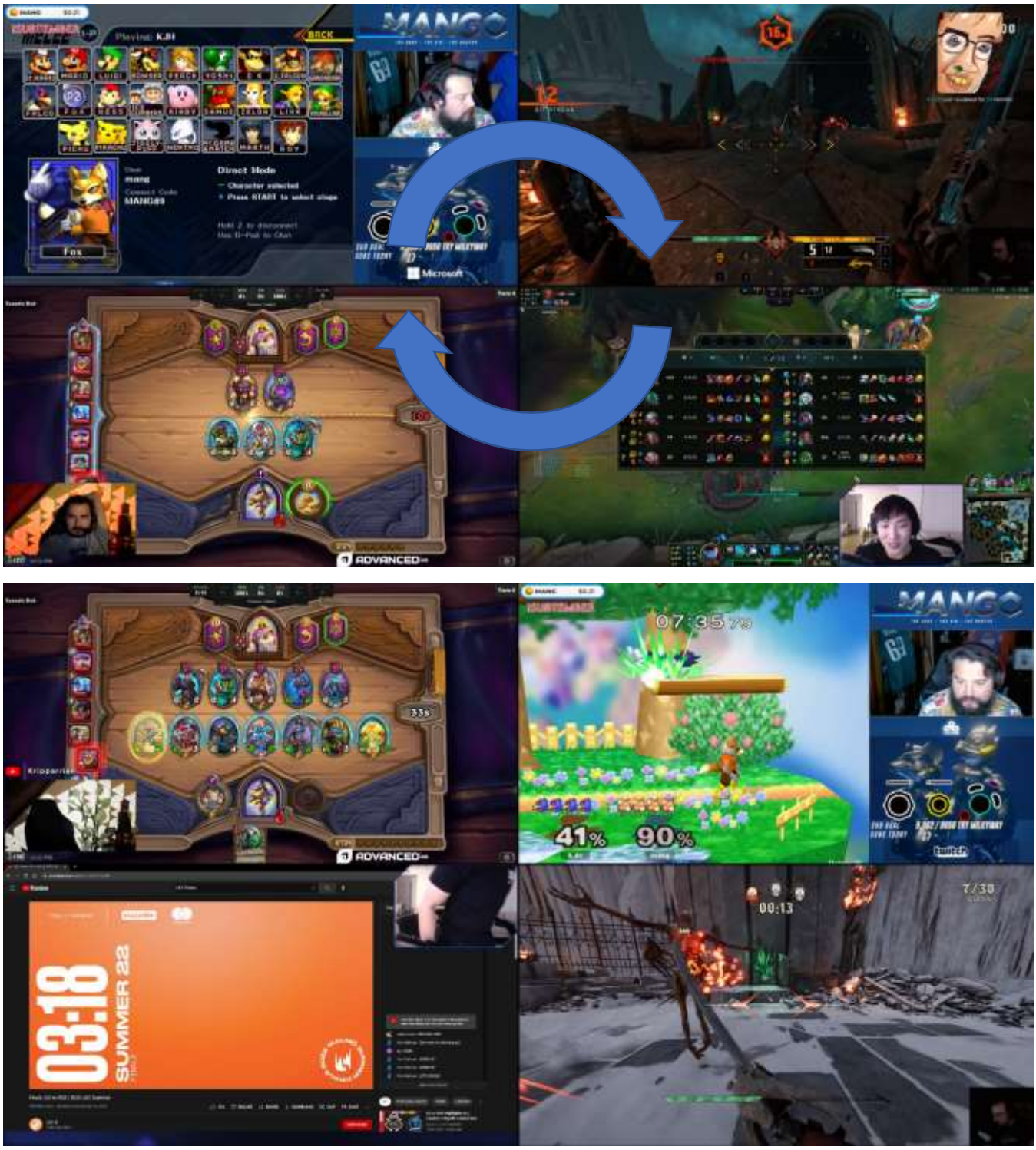
8. Inversion of Livestreams and Chatroom Windows (Dual Livestream)

The livestreams and chatroom windows can be inverted during the dual livestream mode.

9. Quad Livestream Mode

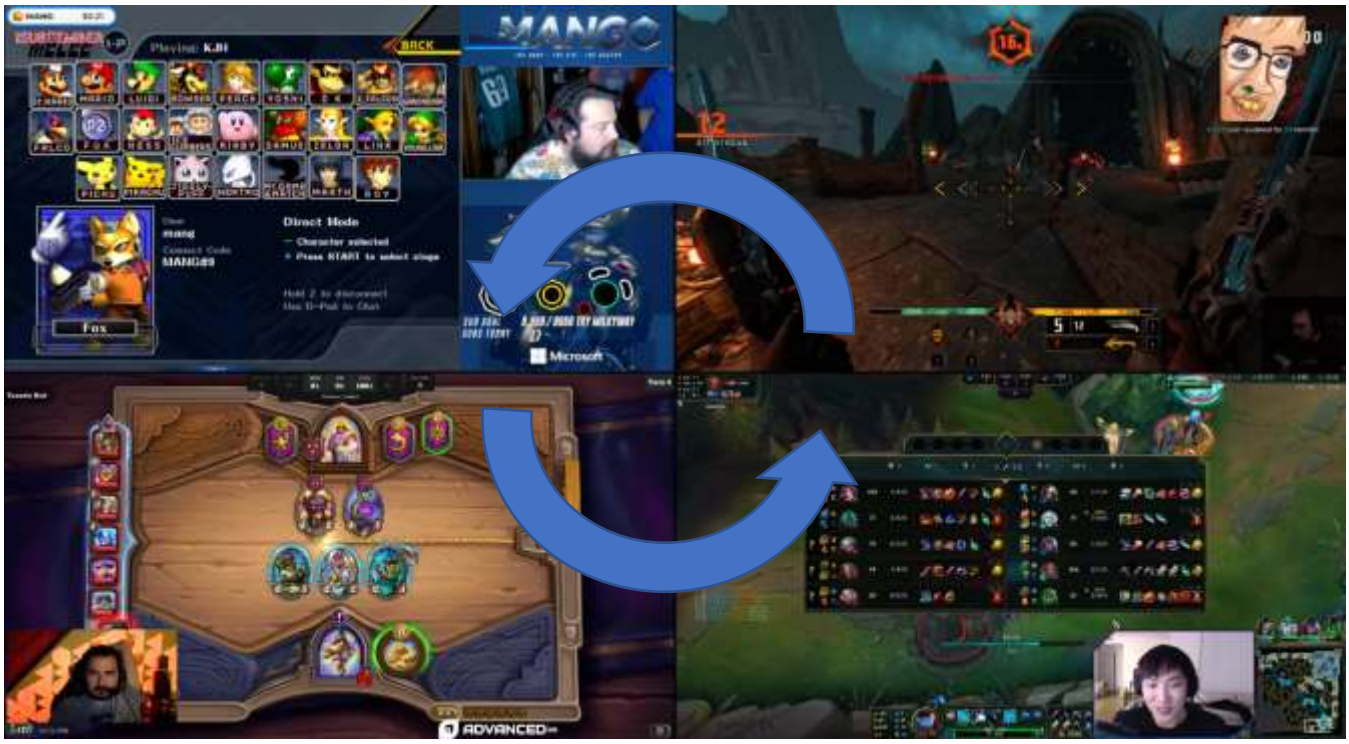
An instance of the client in viewing four livestreams.

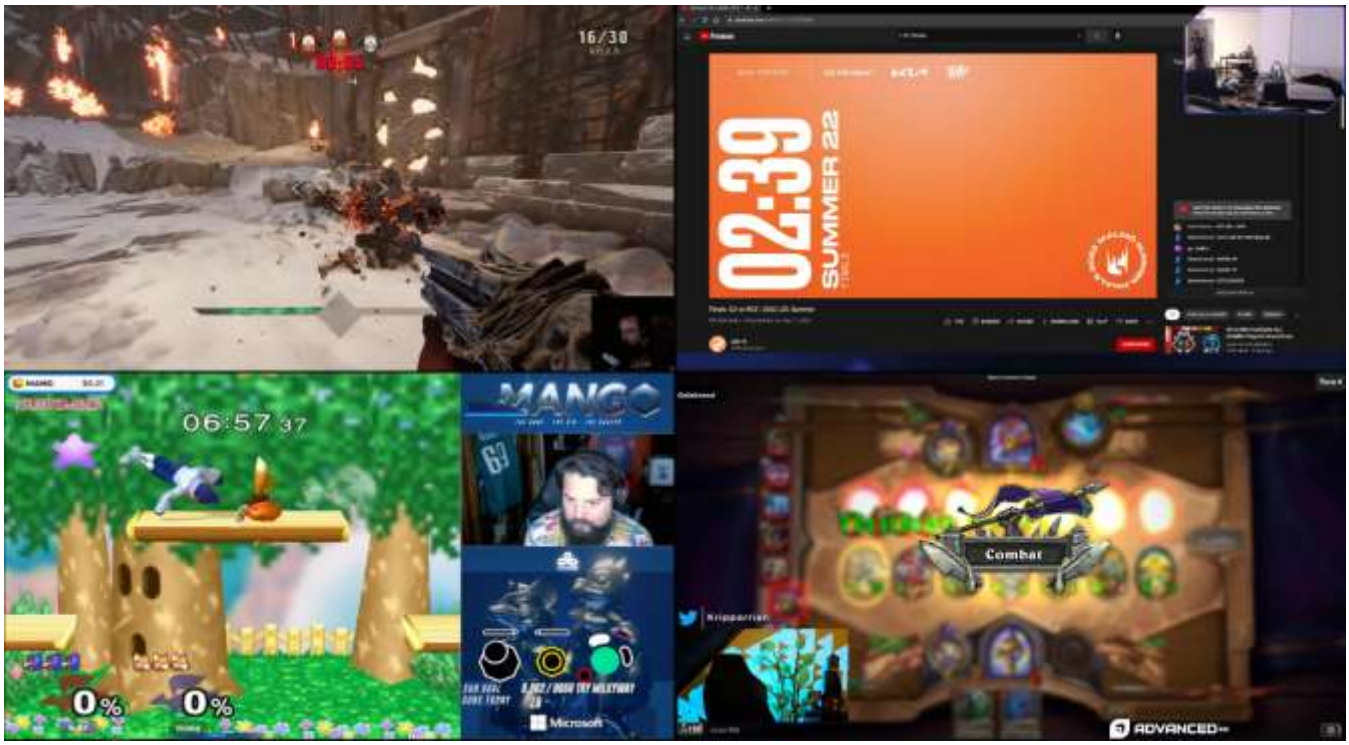
10. Clockwise Rotation of Livestreams (Quad Mode)



Livestreams can be rotated clockwise during the quad livestream mode.

11. Counterclockwise Rotation of Livestreams (Quad Mode)





Livestreams can be rotated counterclockwise during the quad livestream mode.

12. Inversion of Top Row of Livestreams (Quad Mode)

The top row of livestreams can be inverted during the quad livestream mode.

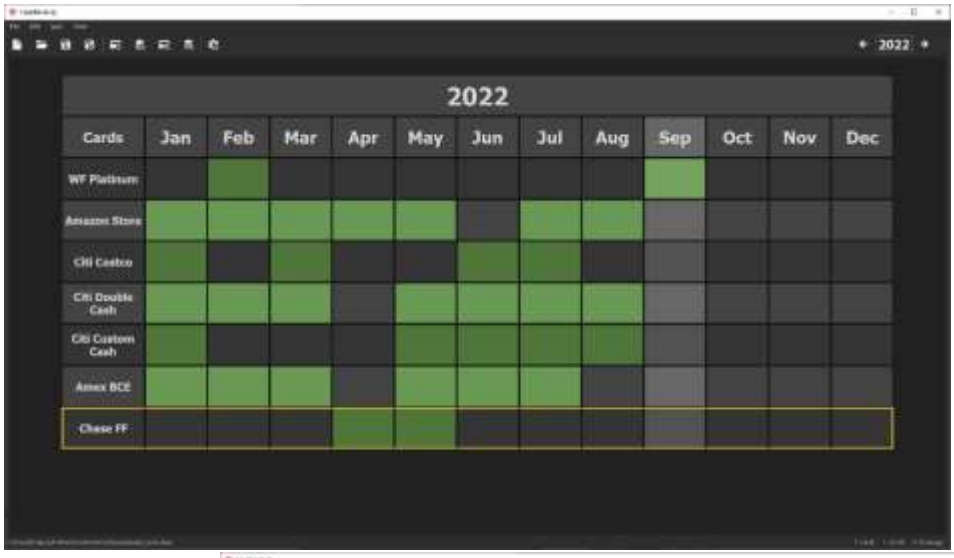
13. Inversion of Bottom Row of Livestreams (Quad Mode)

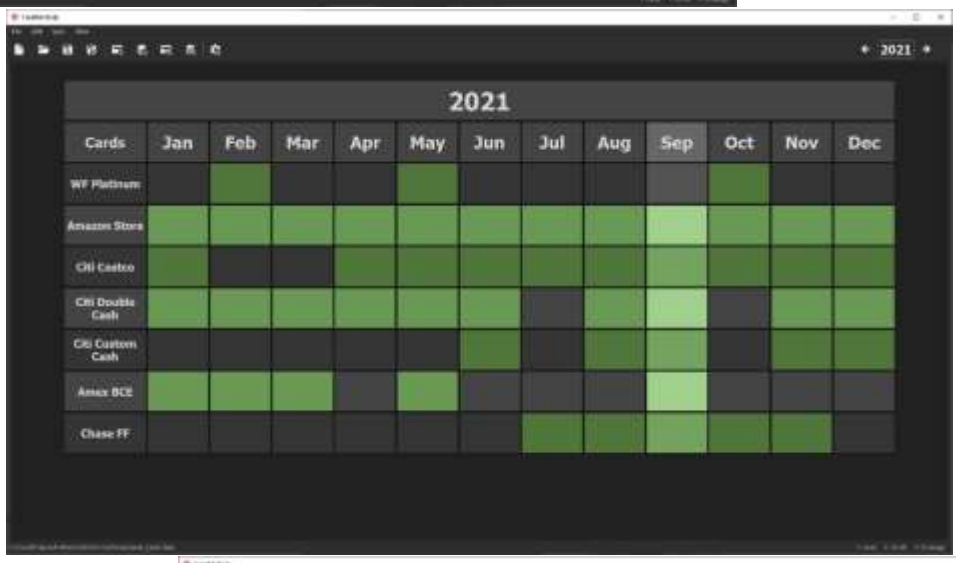
The bottom row of livestreams can be inverted during the quad livestream mode.

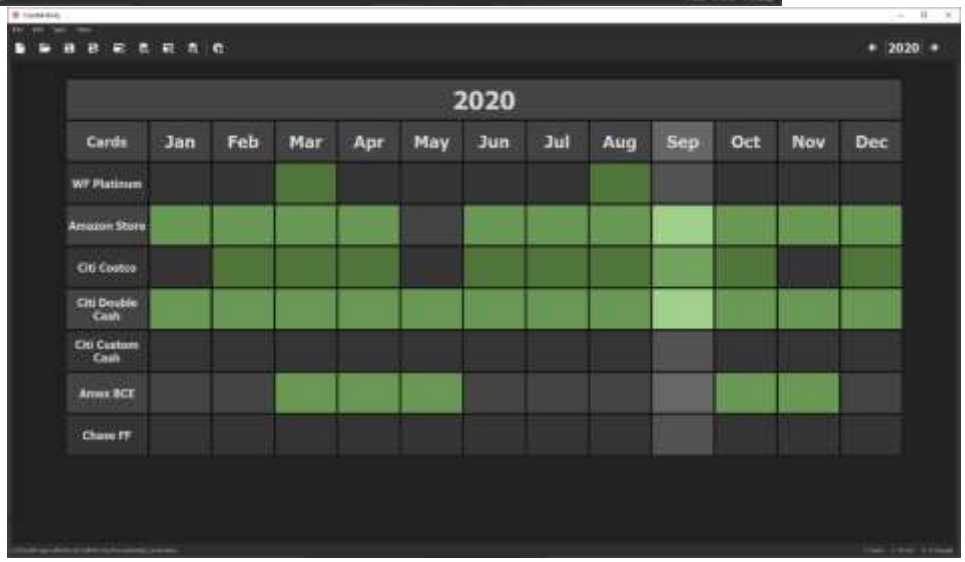
14. Multi-Monitor Support and Hotkeys

Multiple instances of a client can be opened simultaneously. Furthermore, hotkeys are implemented to quickly switch clients positions among my 3 monitors.

Credit Activity Tracker







Project Description

Implements an application for tracking my personal credit card activity over multiple years. Furthermore, the project integrates with the API of You Need A Budget (YNAB), a service for budgeting one's finances. As a result, the application allows for easily syncing credit card activity from my personal YNAB account.

Motivation

Often, as one increasingly opens new credit cards with better reward structures, older and less rewarding cards lose their appeal. Naturally, in this case, most would prefer to shift focus to their newer cards. However, as the older cards are used less and less, or perhaps forgotten altogether, these cards may face account inactivity. If the period of inactivity extends long enough, then the financial institution may close the account.

I had first conceived of a tracking solution employing Excel to help prevent account closures. However, the initial solution was written using Visual Basic for Applications (VBA) which introduced difficulty when I aspired to make improvements. Furthermore, while Excel can be extremely intuitive due to its ubiquity, the application is quite inefficient and slow compared to a specialized solution.

Design Goals

- Remove the dependence on Excel and VBA by porting the initial solution to Python
- Retain and adapt useful features of Excel, namely its grid of cells
- Maximize the longevity of the solution by minimizing the number of Python dependencies
- Improve the solution's speed and footprint

Benefits

- Provides a lightweight application for tracking credit card usage
- Integrates with my personal YNAB account, allowing for an effortless sync with the financial service
- Ensures a long-term solution by minimizing Python dependencies, with the most likely point of failure being the `ynab-client` module

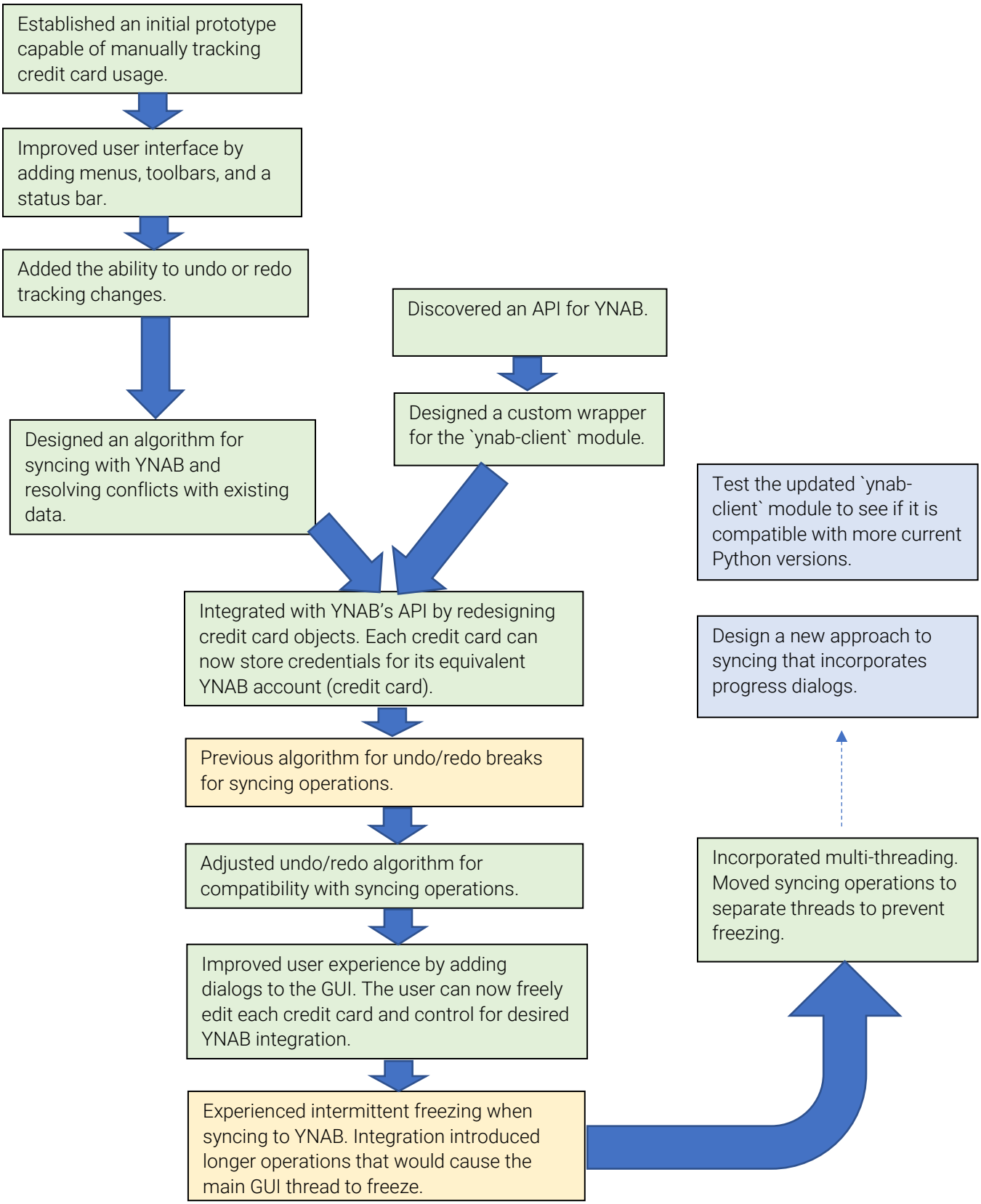
Languages

1. Python 3.6

Python Dependencies

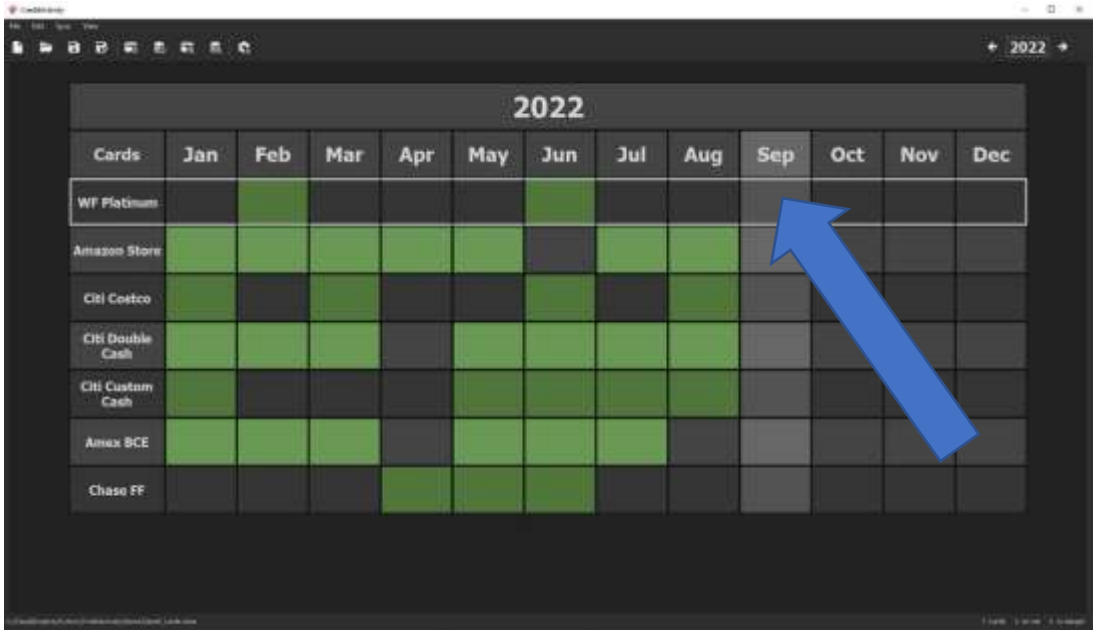
1. pyqt5
2. ynab-client

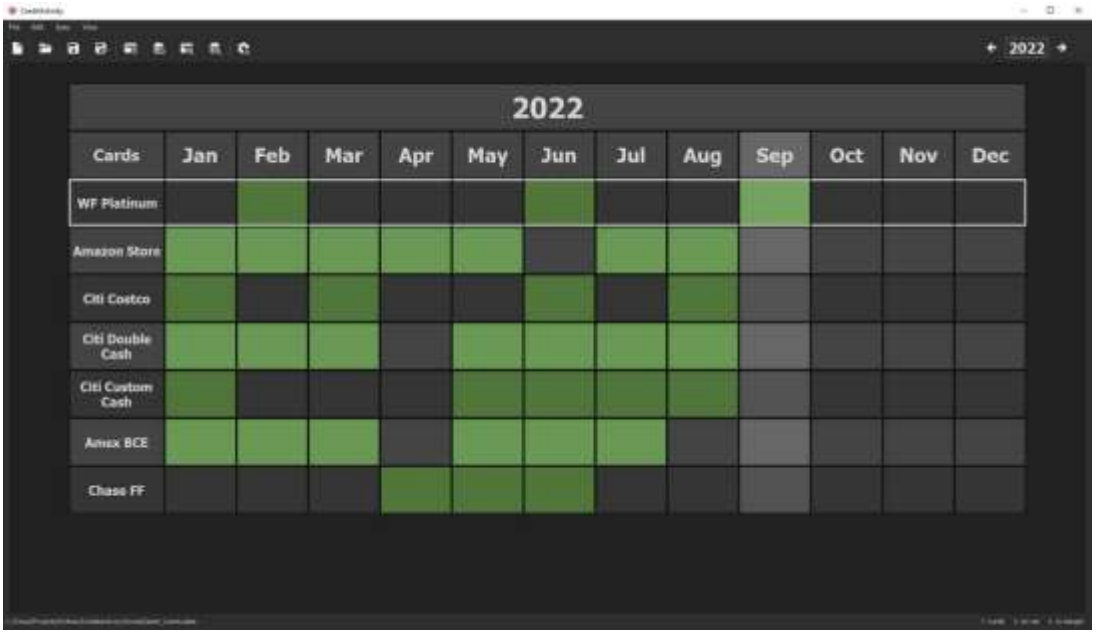
Notable Milestones & Roadblocks



Features

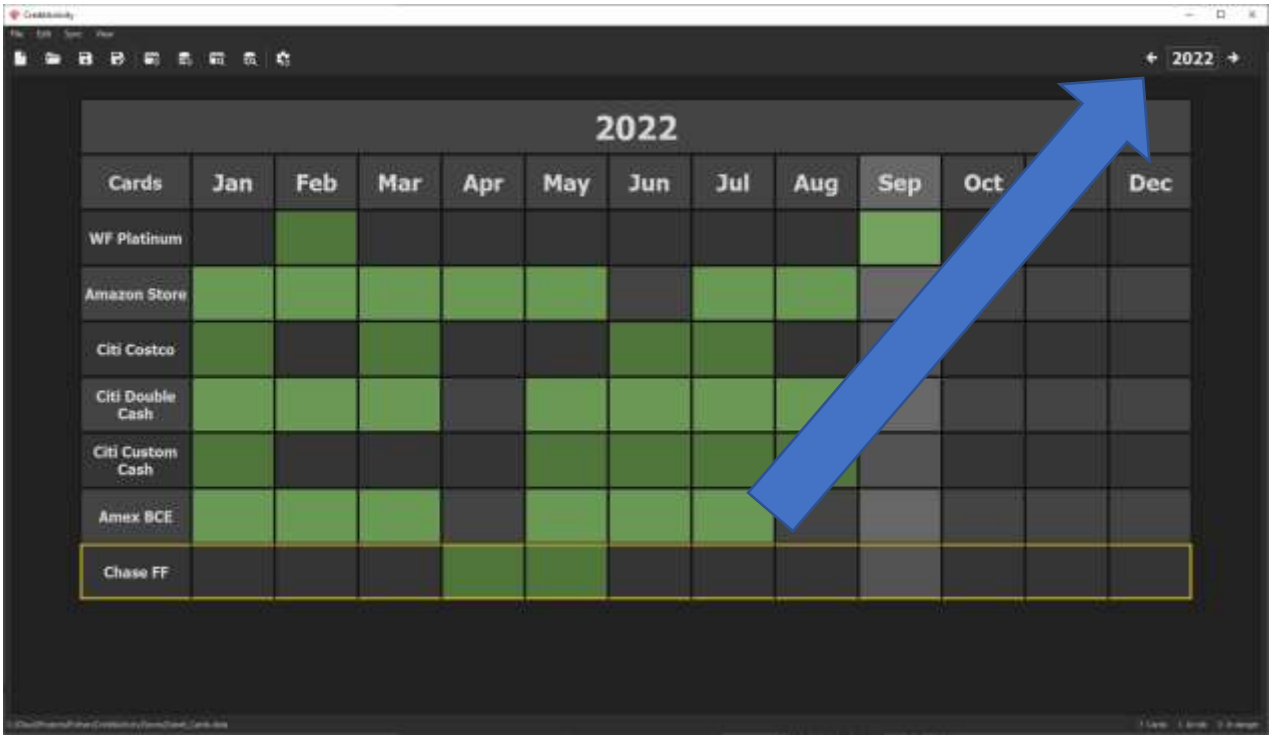
1. Tracking Credit Card Activity

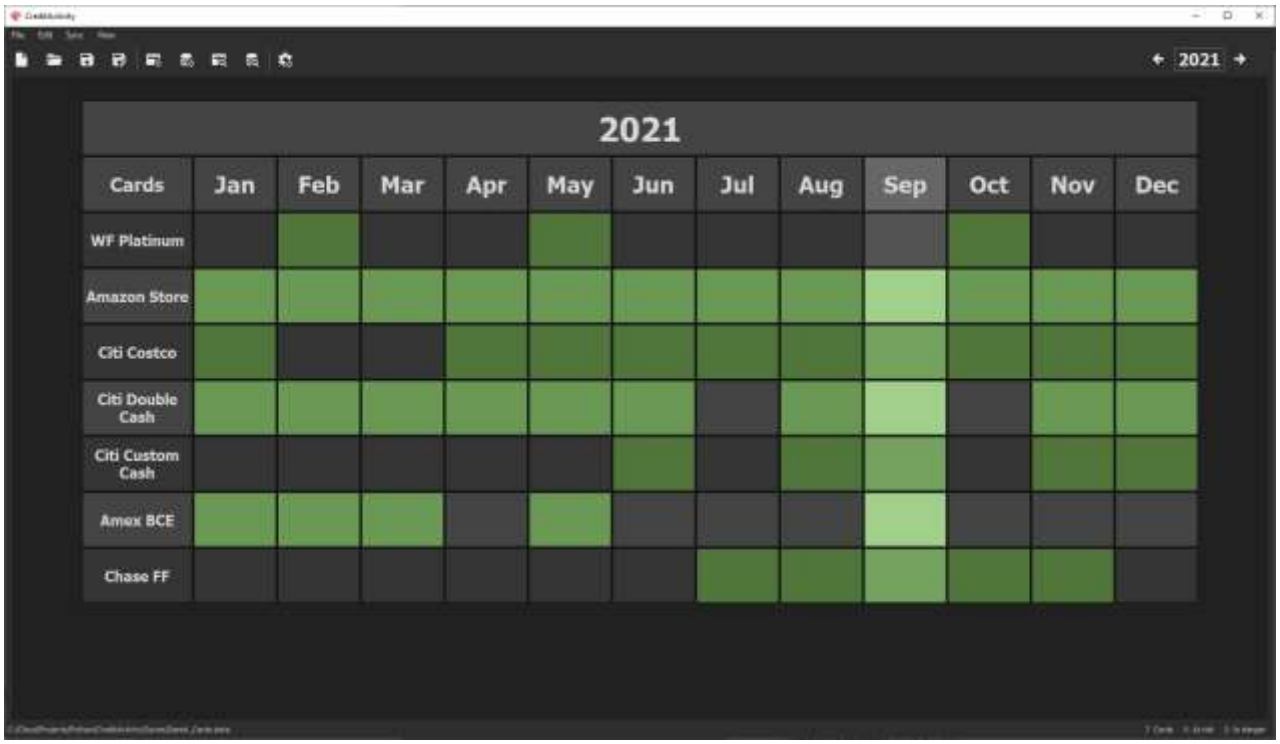




Clicking on a cell will activate it, indicating that the corresponding credit card was used that month.

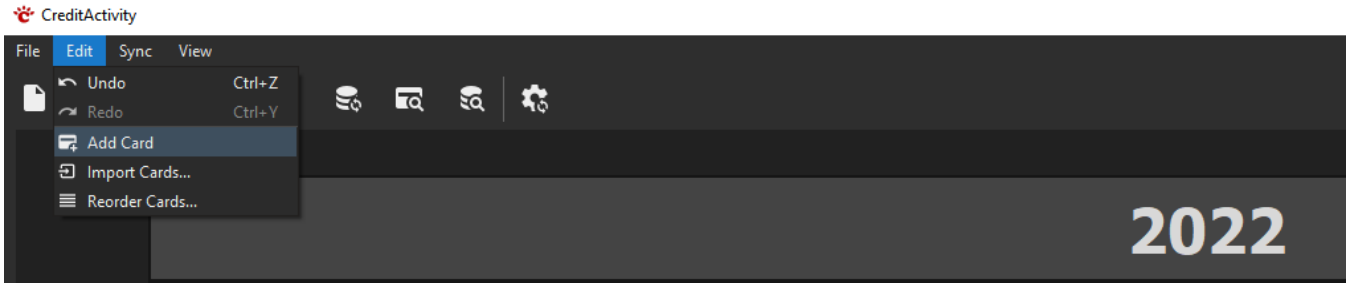
2. Multi-Year Support

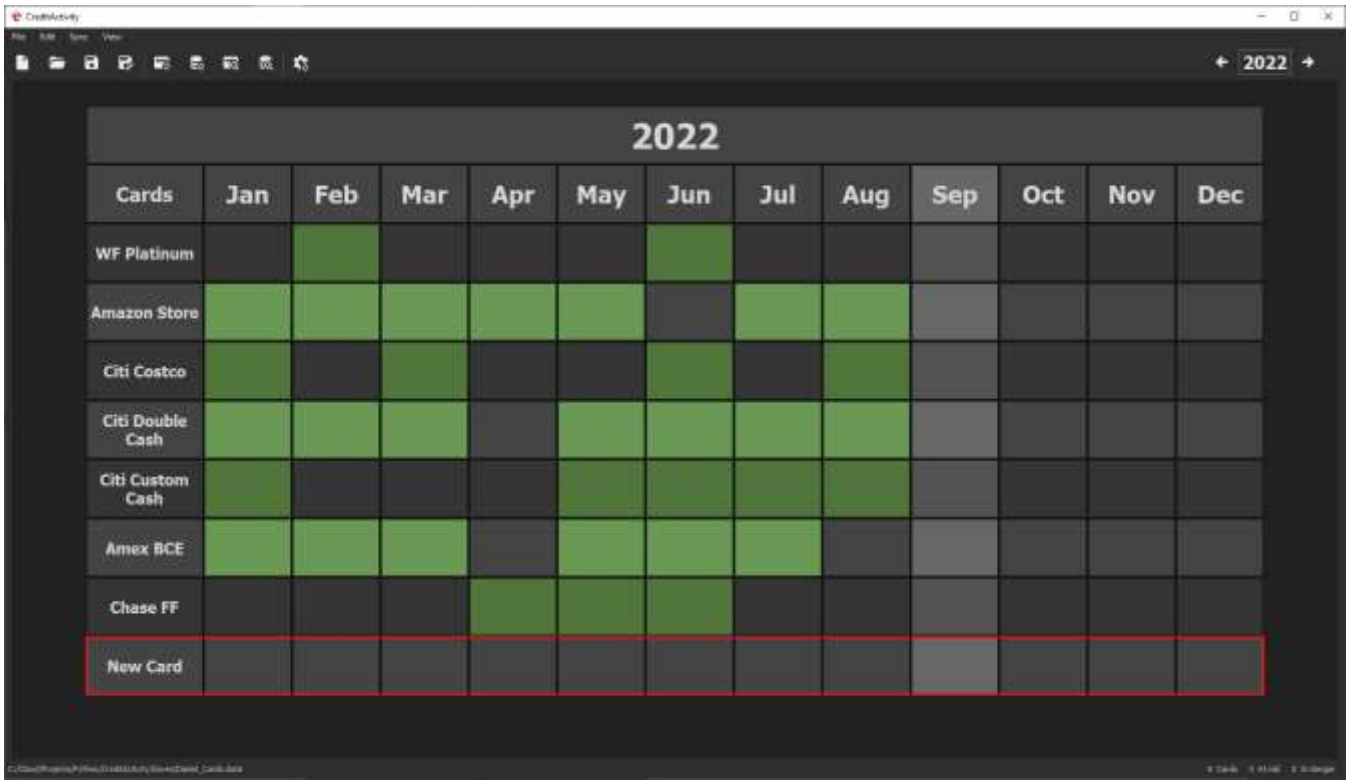




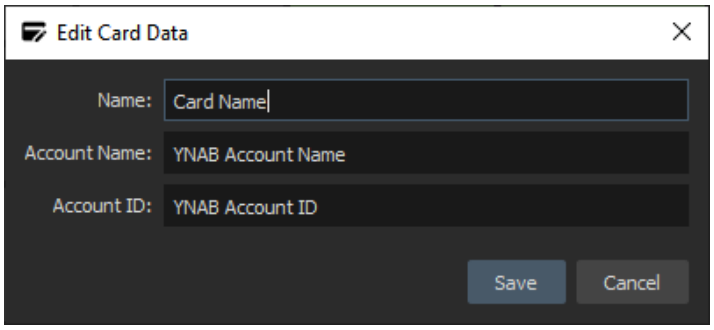
The current year for tracking credit card usage can be selected.

3. Creation of Credit Cards



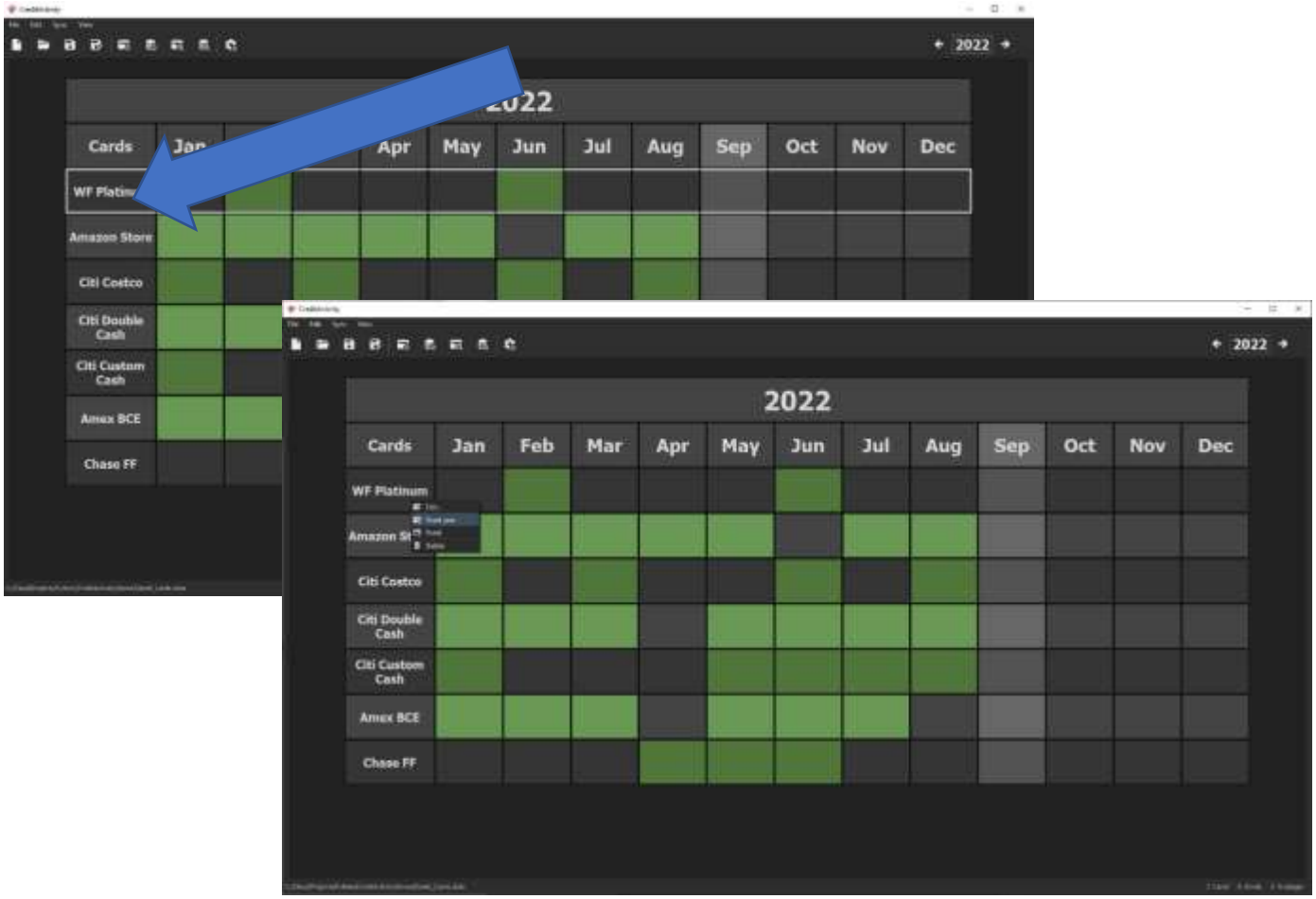


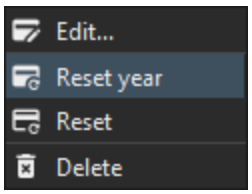
Cards can be created under the Edit menu.



The card's name, YNAB account name, and YNAB account ID can be edited afterwards.

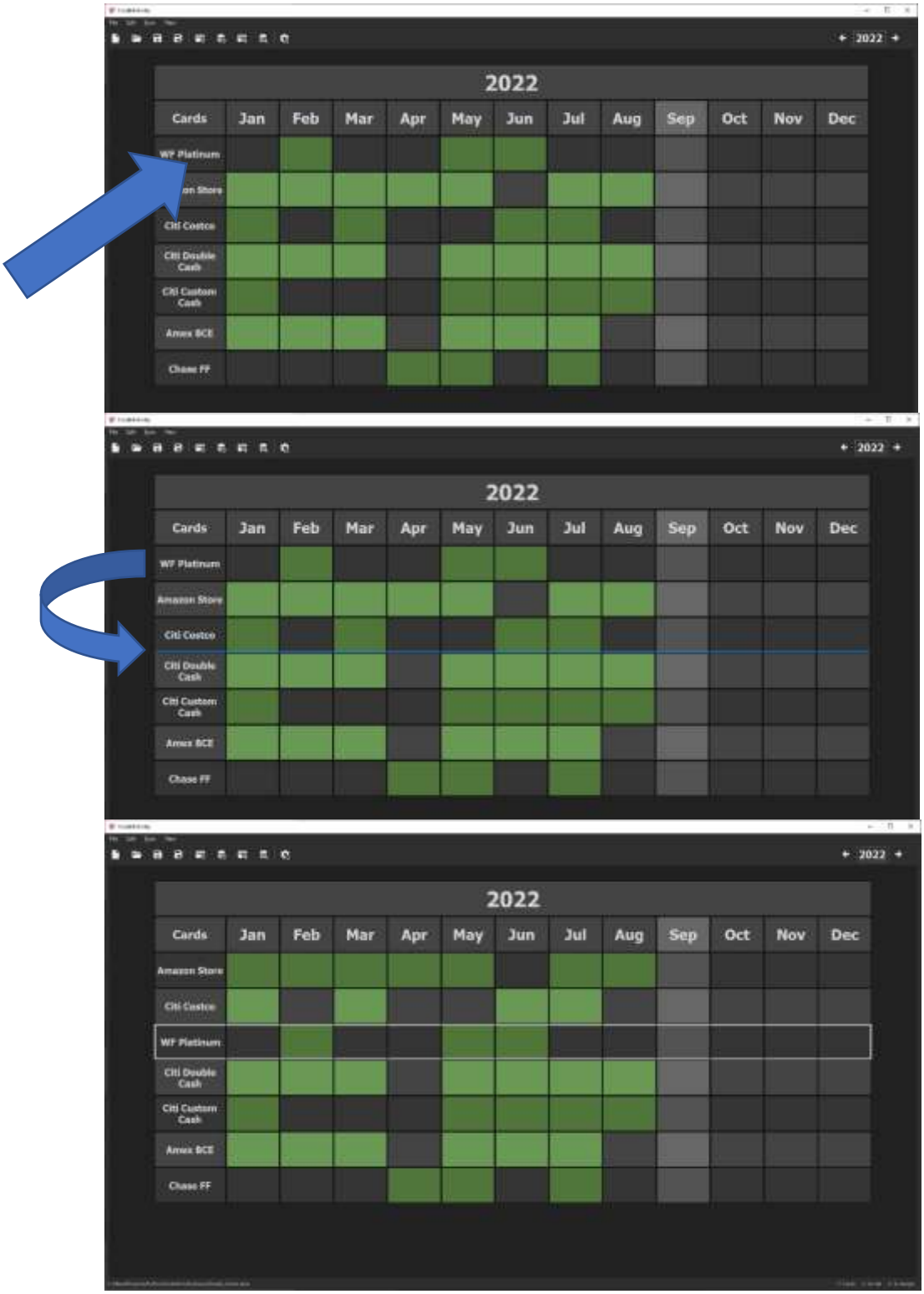
4. Credit Card Context Menu





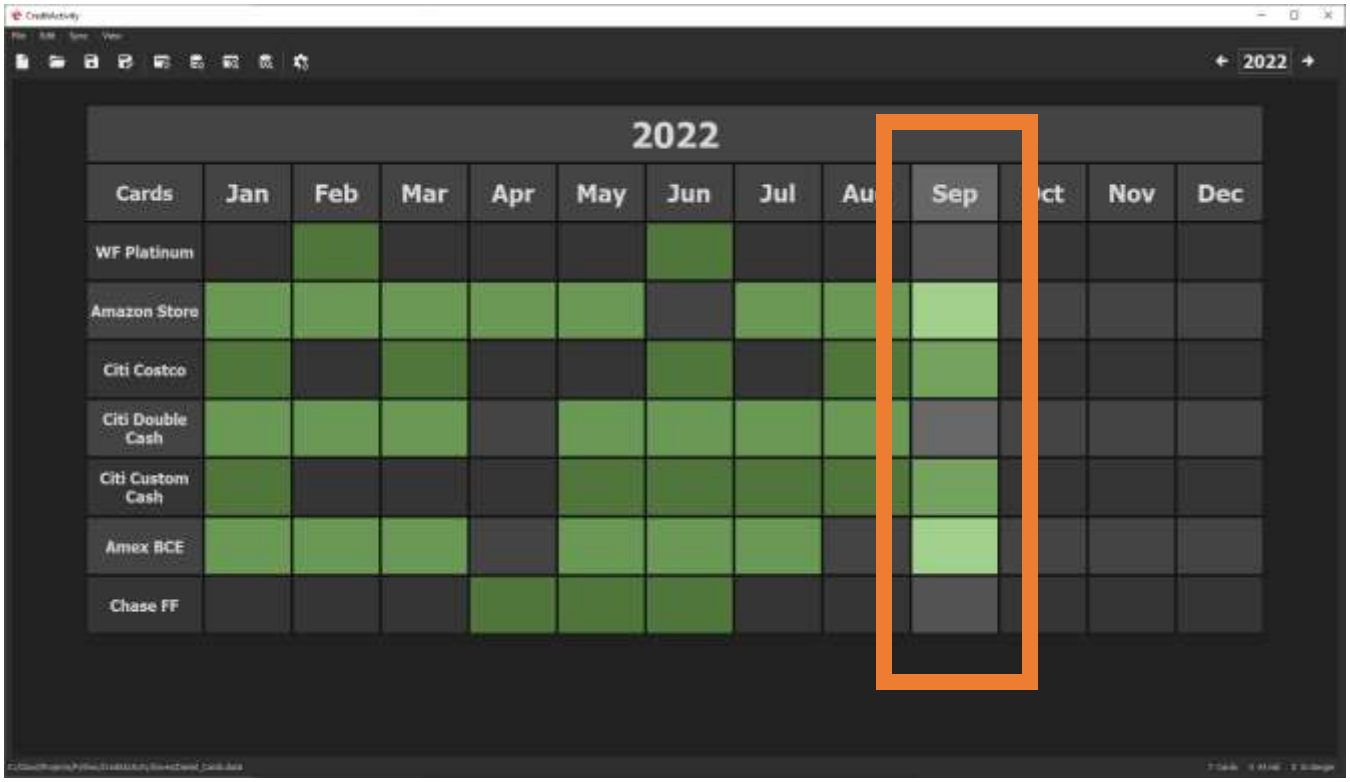
A context menu for credit cards can be accessed by right clicking on the name of a credit card. The context menu provides options for editing the name of the card, resetting its tracking history for the current year, clearing all tracking history, or deleting the card.

5. Reordering of Credit Cards



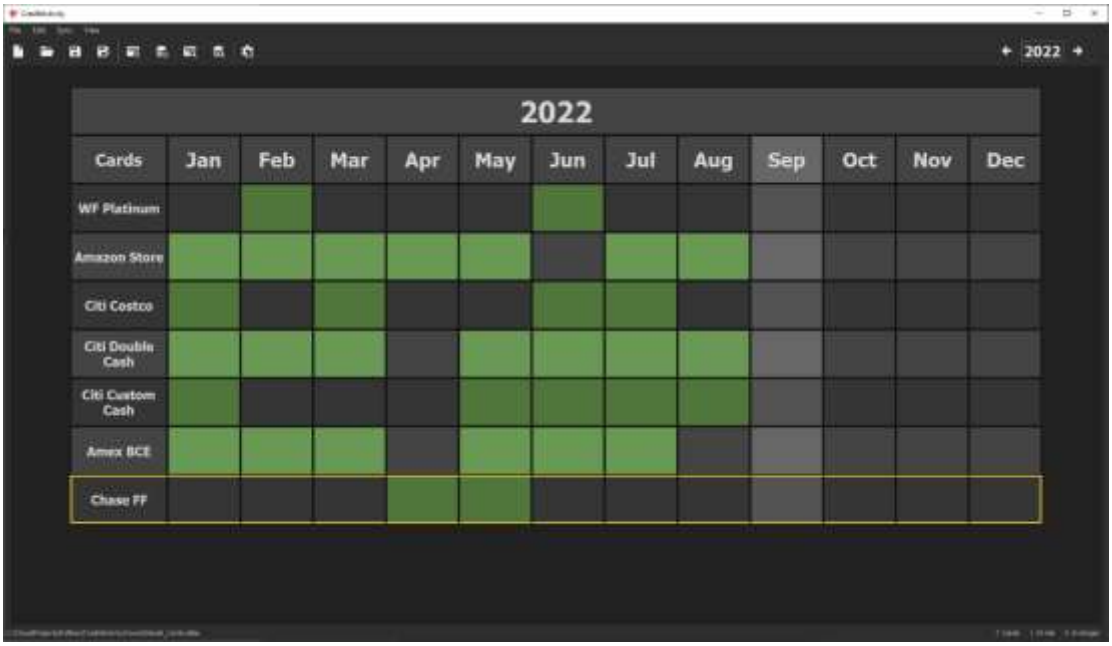
Credit cards can be reordered by clicking its name and then dragging it to the desired position.

6. Highlighting the Current Month

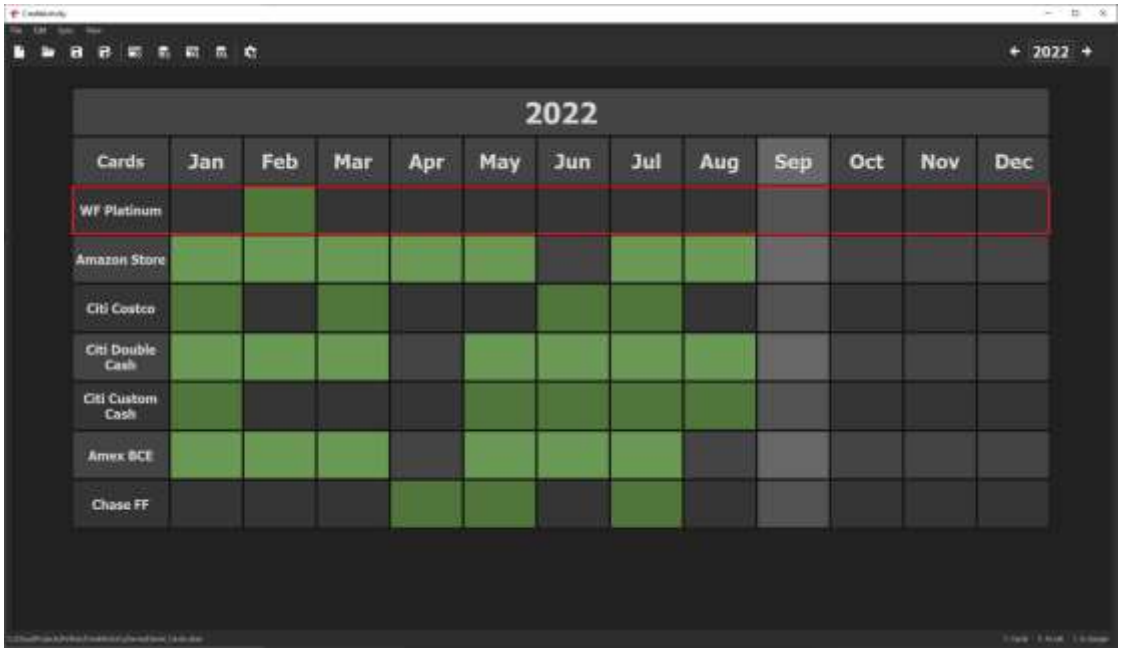


Cells under the current month are highlighted to contrast with other cells, thereby allowing the user to quickly discern the current month.

7. Highlighting of Inactive Credit Cards

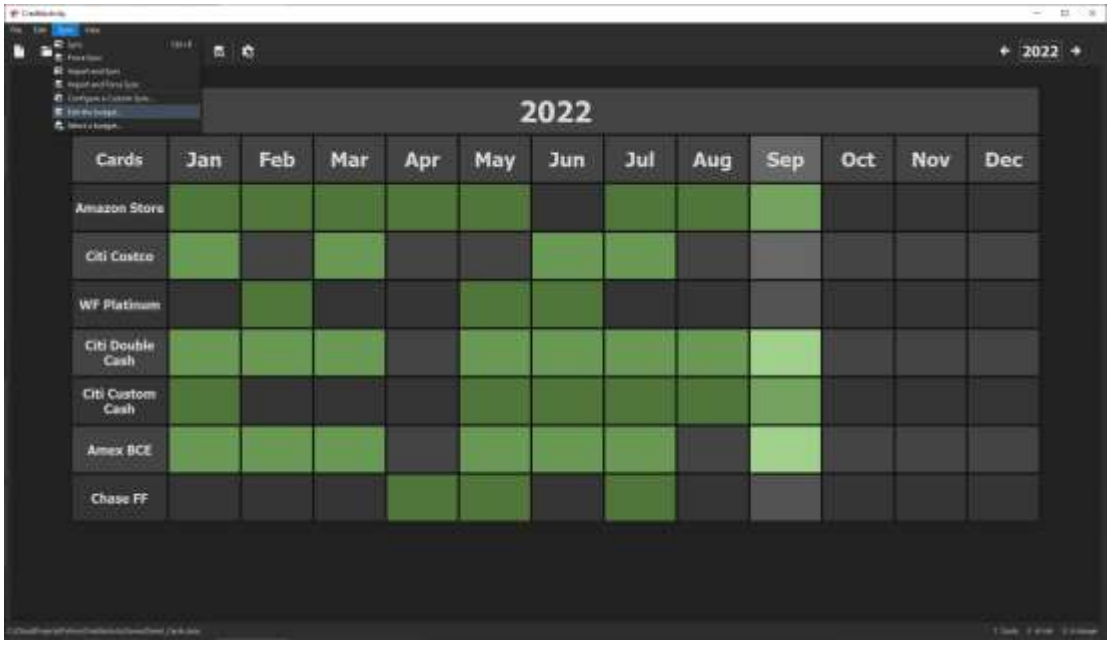


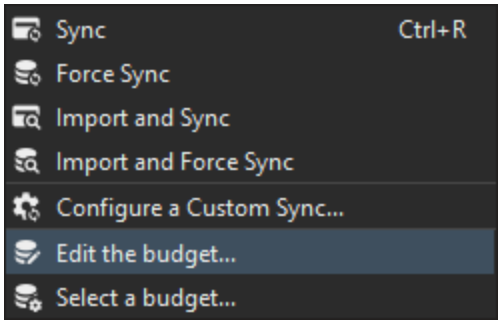
Credit cards that have not been used within the last **4 months** are highlighted yellow to warn of inactivity.

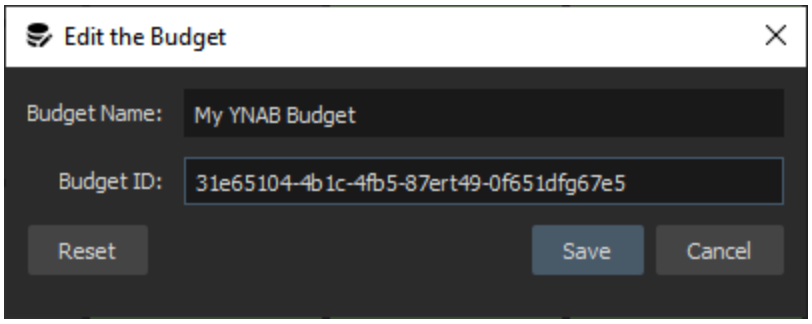


Credit cards that have not been used within the last **7 months** are highlighted red to warn of **account closure**.

8. Integration with a YNAB Budget

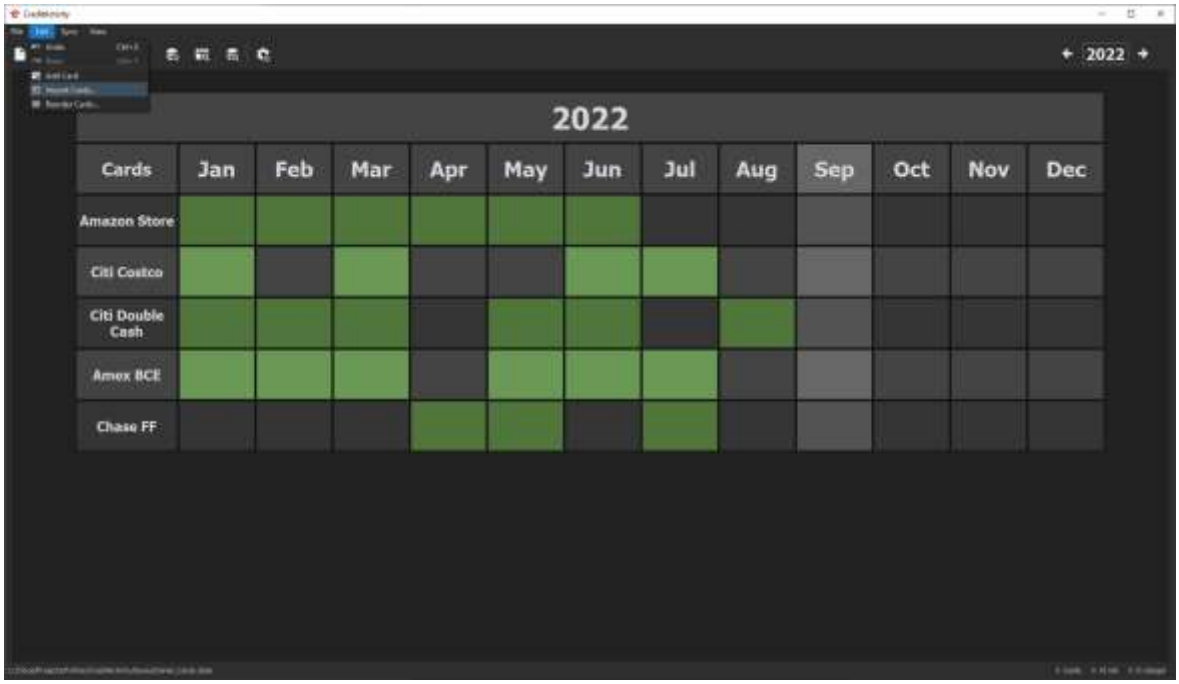


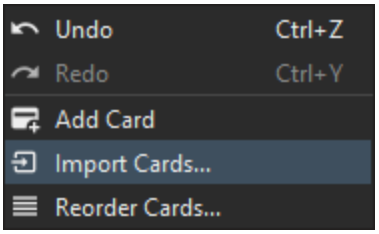


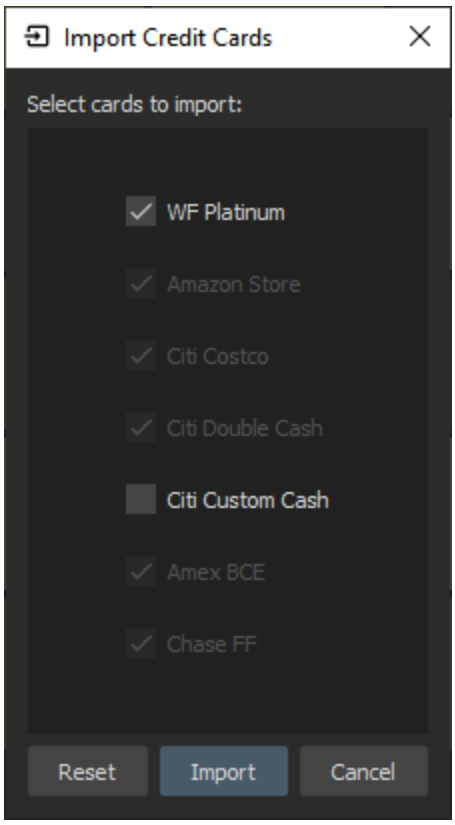


A desired YNAB budget can be selected under the Sync Menu, thereby providing integration with my personal YNAB account via the service's API.

9. Importing Cards from YNAB

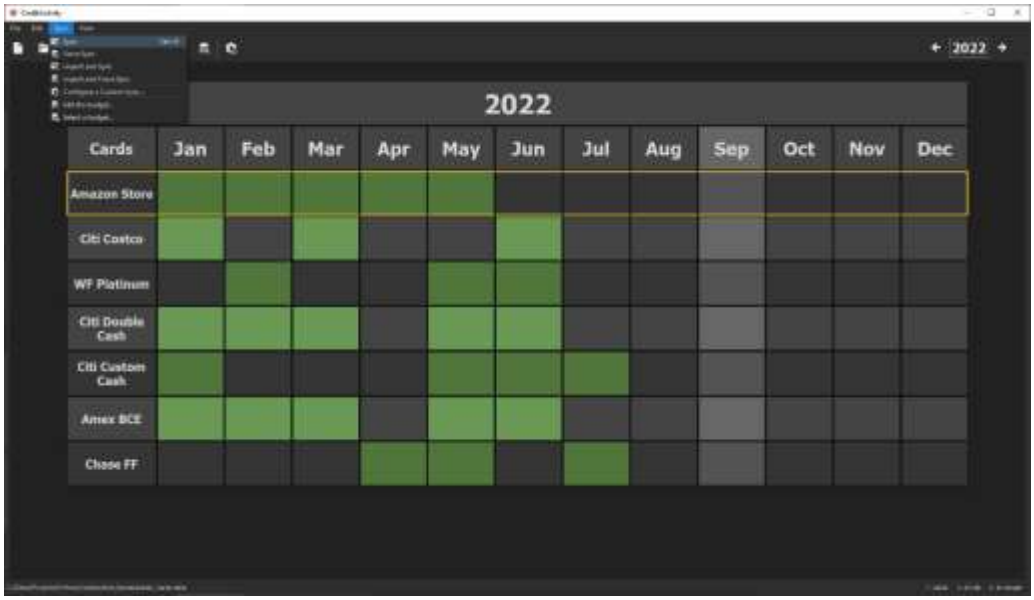


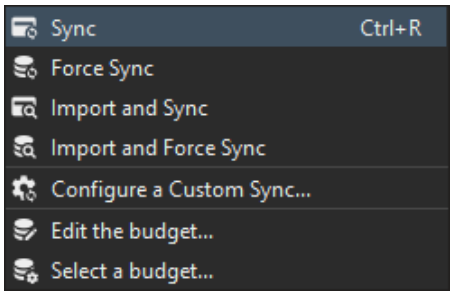


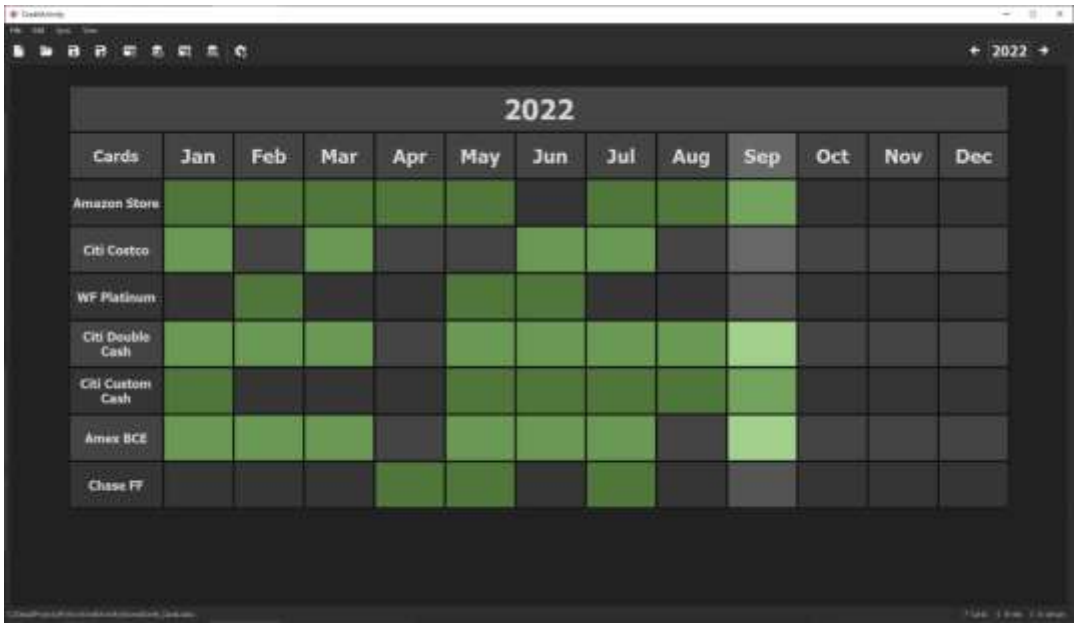


After a valid YNAB budget has been selected, credit cards from YNAB can be imported into the application under the Edit Menu.

10. Syncing Tracking History with YNAB

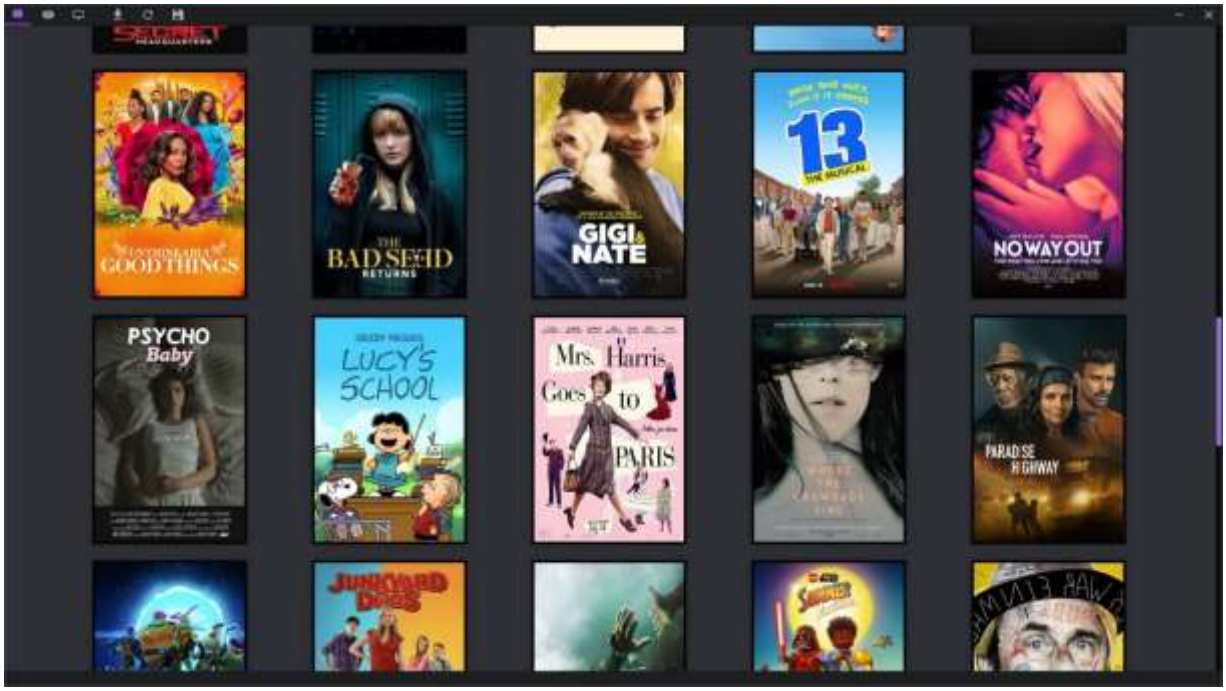


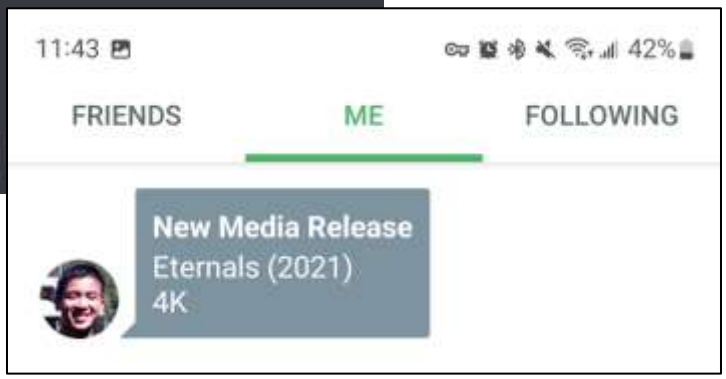




After a valid YNAB budget has been selected, tracking history can be synced with YNAB for quick and effortless tracking.

Blu-ray Release Tracker





Project Description

Implements an application for tracking the Blu-ray releases of desired media.

The project consists of multiple components:

1. GUI Application
2. Raspberry Pi 3 / 4
3. Smartphone

The GUI application facilitates selecting desired media for tracking. Next, the Raspberry Pi is a low-power and low-cost solution for storing all data employed by the application within a database; moreover, the Raspberry Pi can perpetually check for Blu-ray releases at set intervals of time. Finally, once a release has been found, then the Raspberry Pi will send a push notification to my smartphone.

Motivation

I enjoy watching movies; however, not all movies deserve the same level of interest: for some movies, I would prefer watching at home rather than in the cinema. Ironically, I find difficulty in staying informed of the Blu-ray release of these movies.

In a similar manner, for TV shows, often I would prefer the greater bandwidth (quality) offered via Blu-ray as compared to streaming services.

Therefore, I aspired to design a solution to inform me of upcoming media and minimize as much effort required from me to track their Blu-ray releases.

Design Goals

- Provide a continually updating catalog of upcoming movies, reducing the need for me to stay updated with current releases
- Streamline the user experience by reducing or hiding loading times as much as possible
- Utilize the Raspberry Pi as a server for storing media data and tracking releases 24/7

Benefits

- Eliminates the need to manually check for Blu-ray releases
- Automatically notifies me of desired releases

Languages

1. Python 3.8

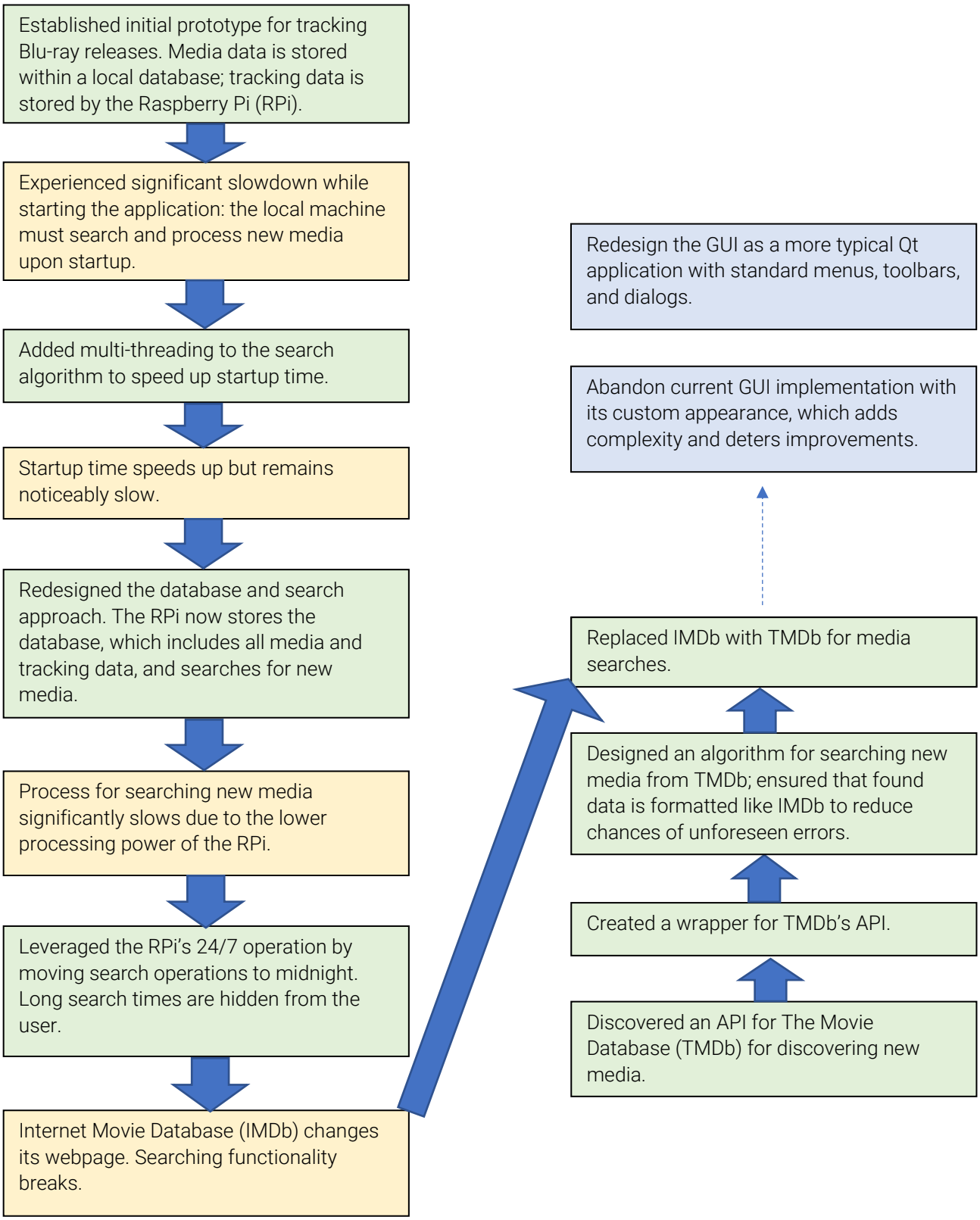
Python Dependencies

1. requests
2. bs4
3. pyqt5
4. omdb
5. cinemagoer
6. python-dateutil
7. pushbullet.py
8. mysql-connector-python

External Dependencies

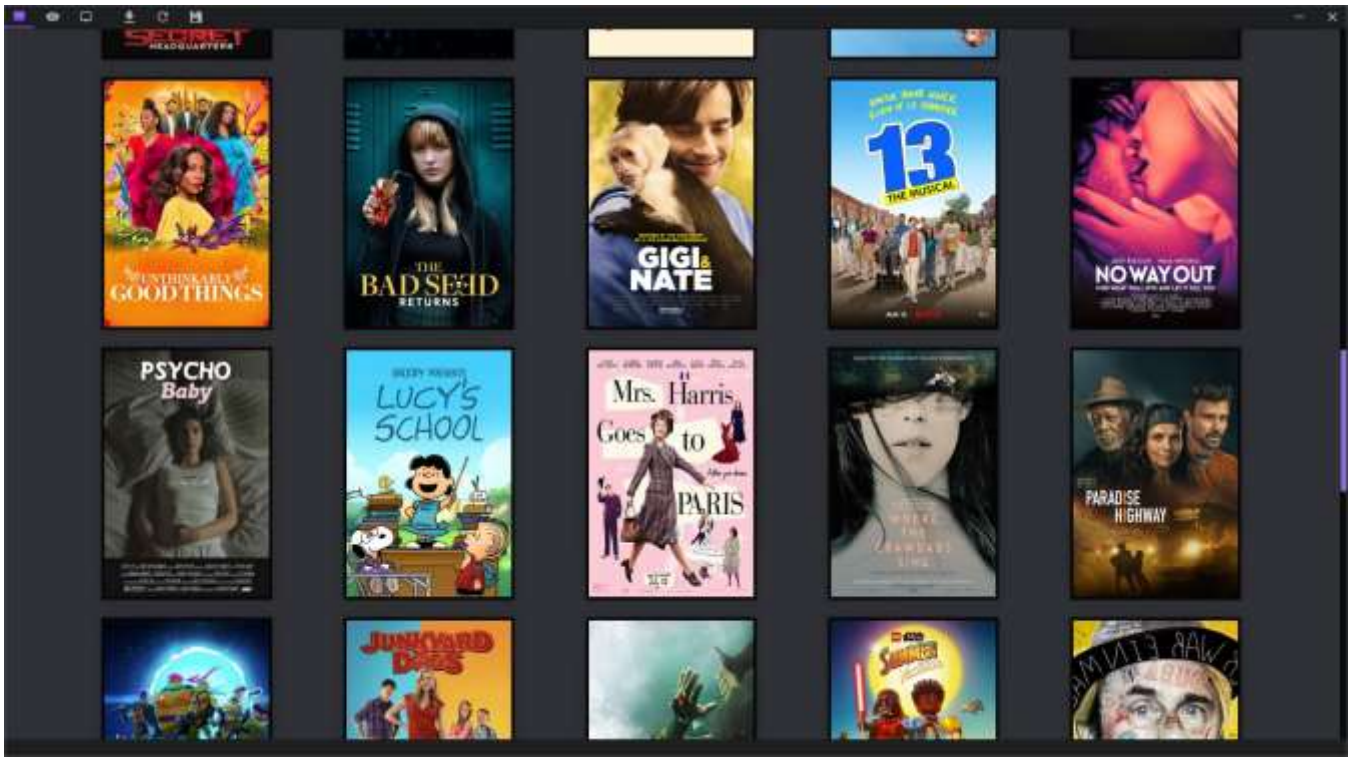
1. Smartphone
2. Raspberry Pi 3 or 4
3. MariaDB
4. HeidiSQL

Notable Milestones & Roadblocks



Features

1. **Revolving Catalog of Newly Released Movies**



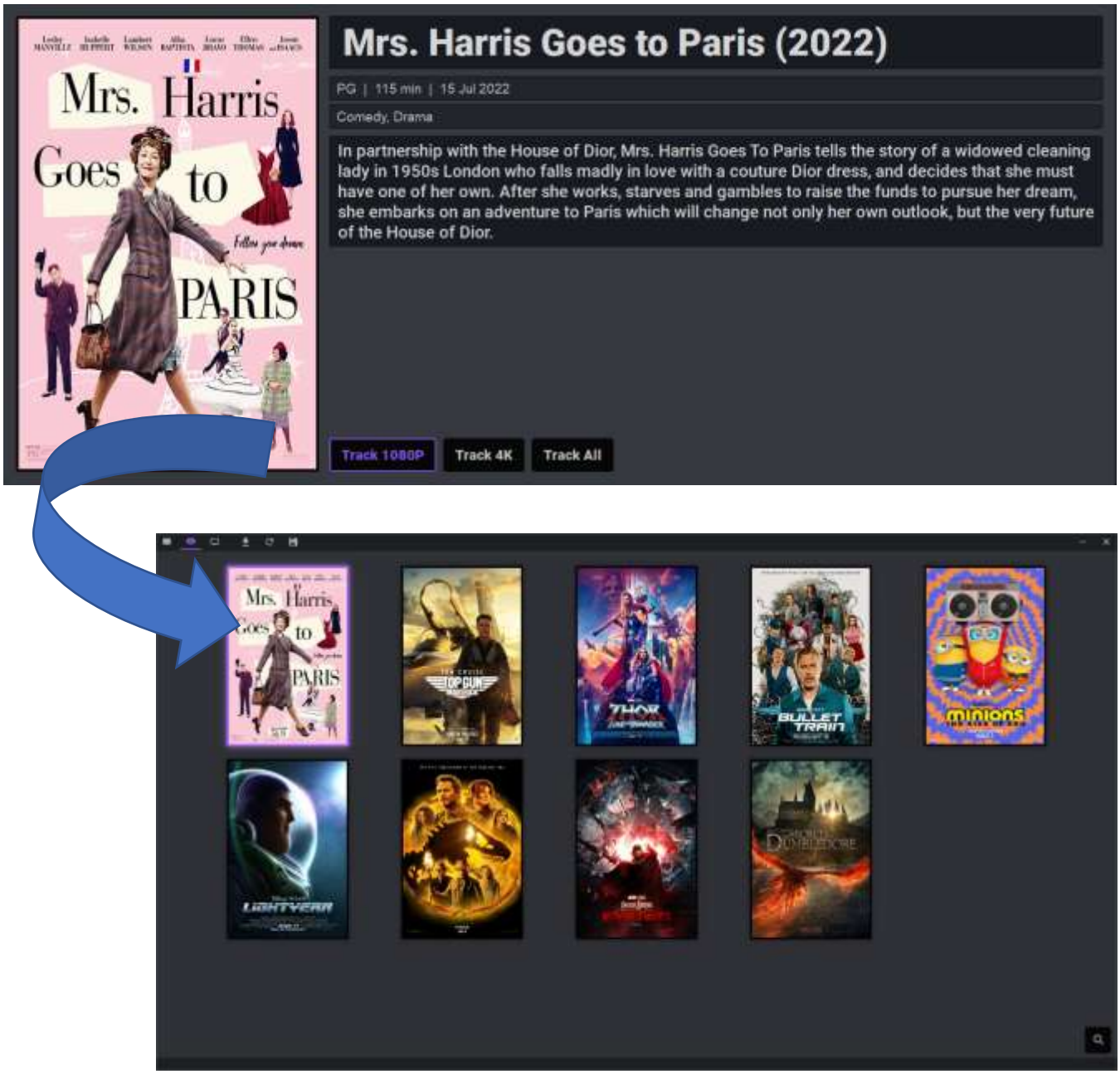
The Raspberry Pi performs weekly checks for newly released or upcoming media and imports the data into its database. A revolving catalog is maintained by expunging undesired media outside of a three-month window from the current date.

The GUI application (pictured above) displays the revolving catalog to the user on a desktop PC.

2. Inspecting Media

Left clicking a media will expose an inspection menu, providing options for tracking or untracking (if eligible) the media's Blu-ray release(s).

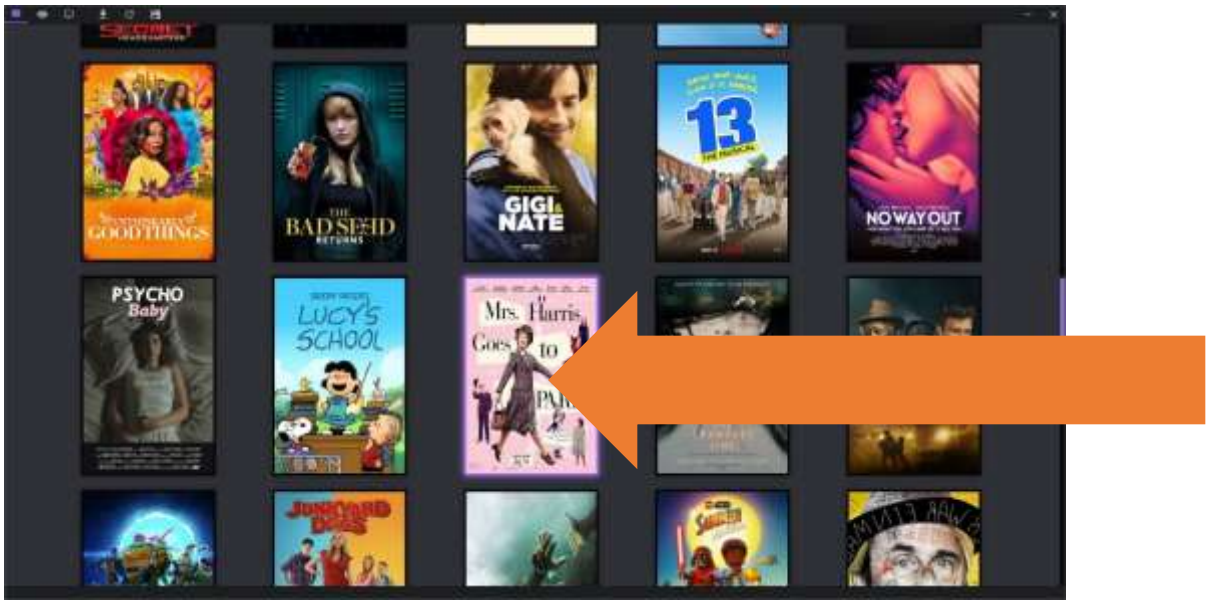
3. Checking the Blu-ray Release Status of Currently Tracked Media

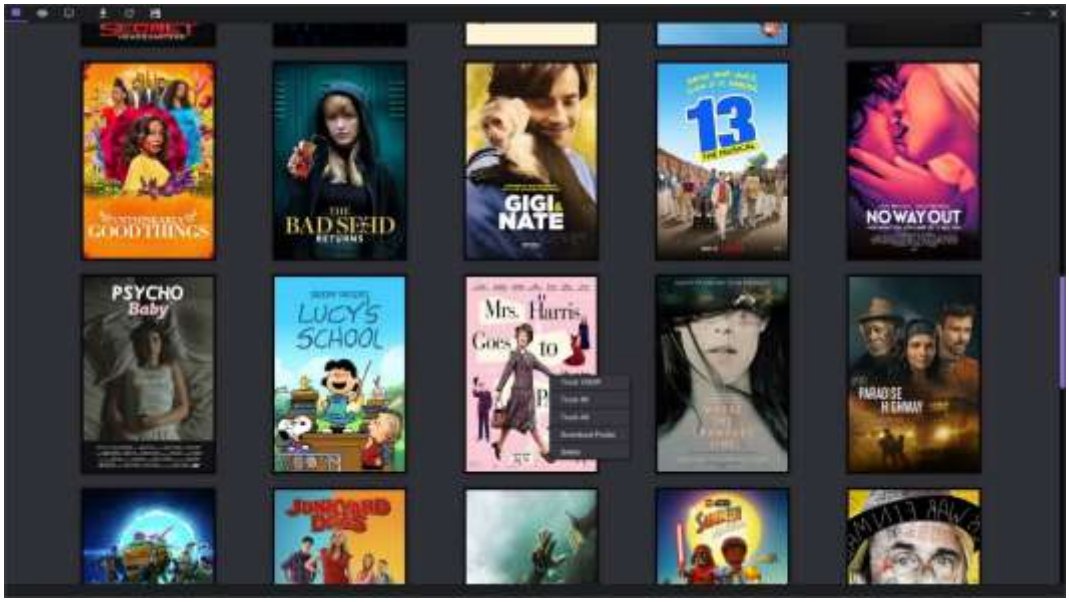


Tracking the Blu-ray release of a media will transfer the media into a separate catalog of either desired movies or television shows.

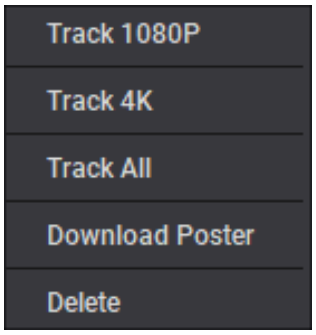
The Blu-ray release status of a media can be checked by inspecting a desired (currently tracked) media.

4. Media Context Menu

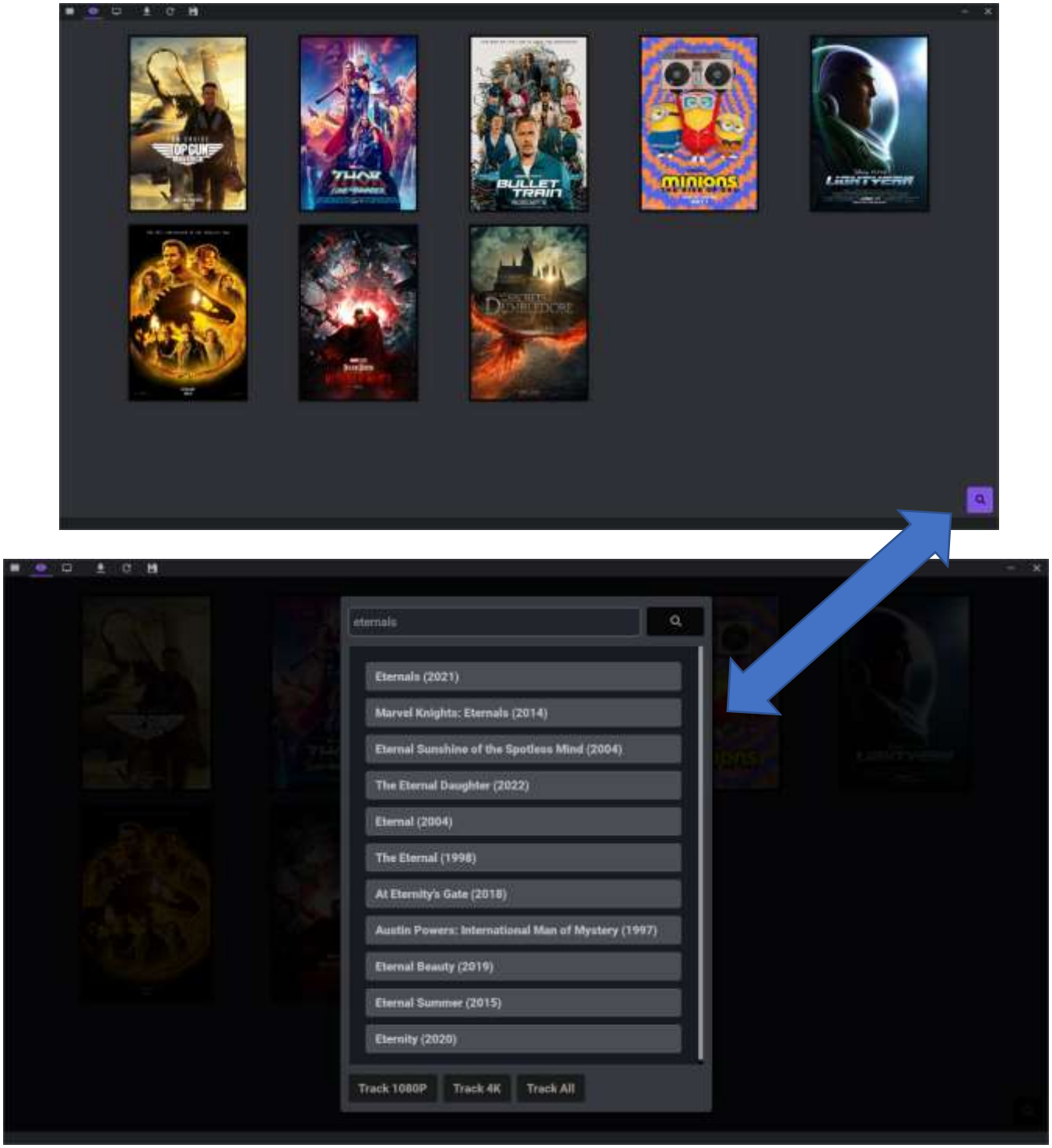




A context menu for media can be accessed by right clicking a media. The menu provides options for tracking or untracking a Blu-ray release, re-downloading (refreshing) the media's poster, or deleting the media from the database.

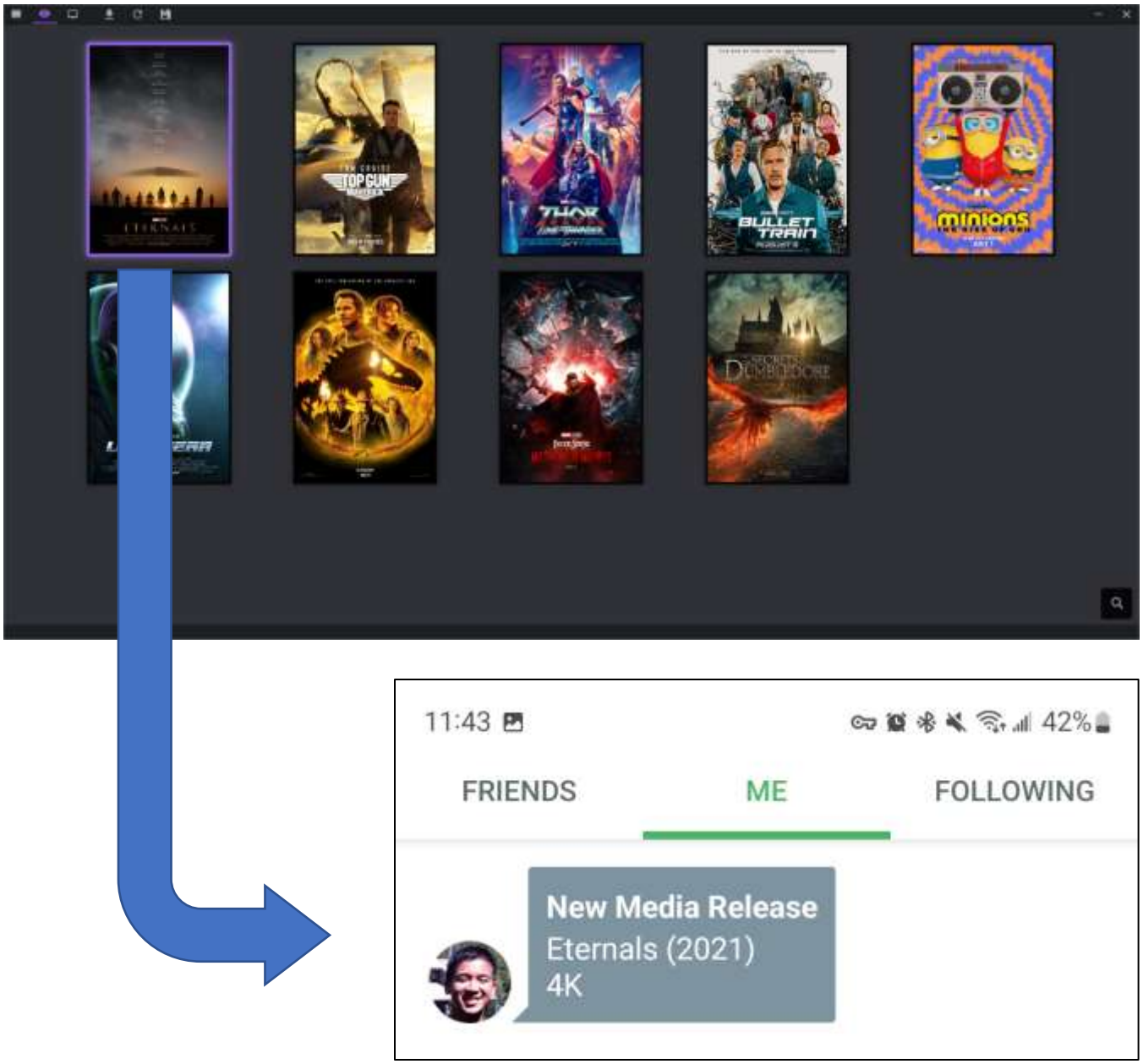


5. Manual Search for a Desired Movie or Television Show



Clicking the search button within the catalog for desired movies or television shows will expose a dialog for custom searches. For example, searching for Eternals will yield the results shown above.

6. Notification of a Blu-ray Release



The Raspberry Pi checks for desired Blu-ray releases at set intervals. Once a Blu-ray has been found, then the Raspberry Pi will send a push notification to my smartphone.