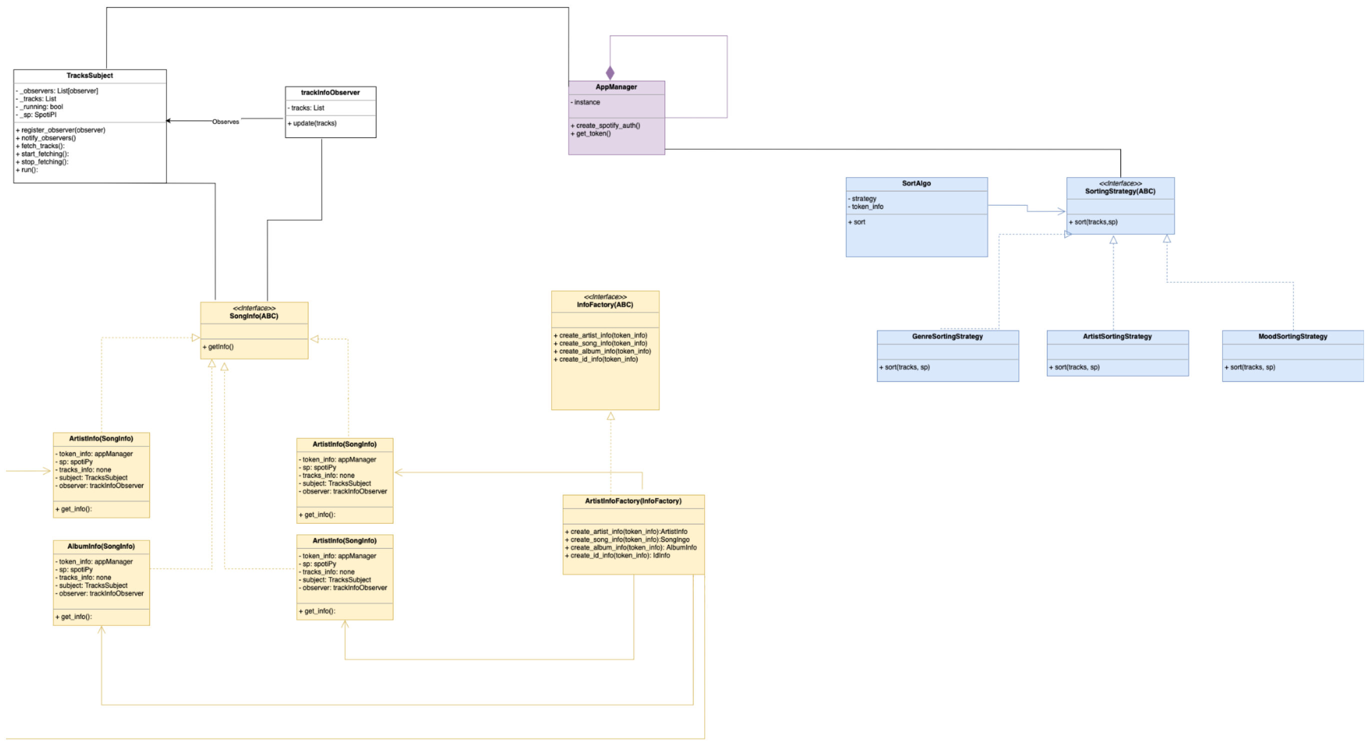
Project 7: Semester Project
Project Name: SpotiSort
Team Members: Kieran Stone, Lukas Voemel

2. Final State of the System Statement

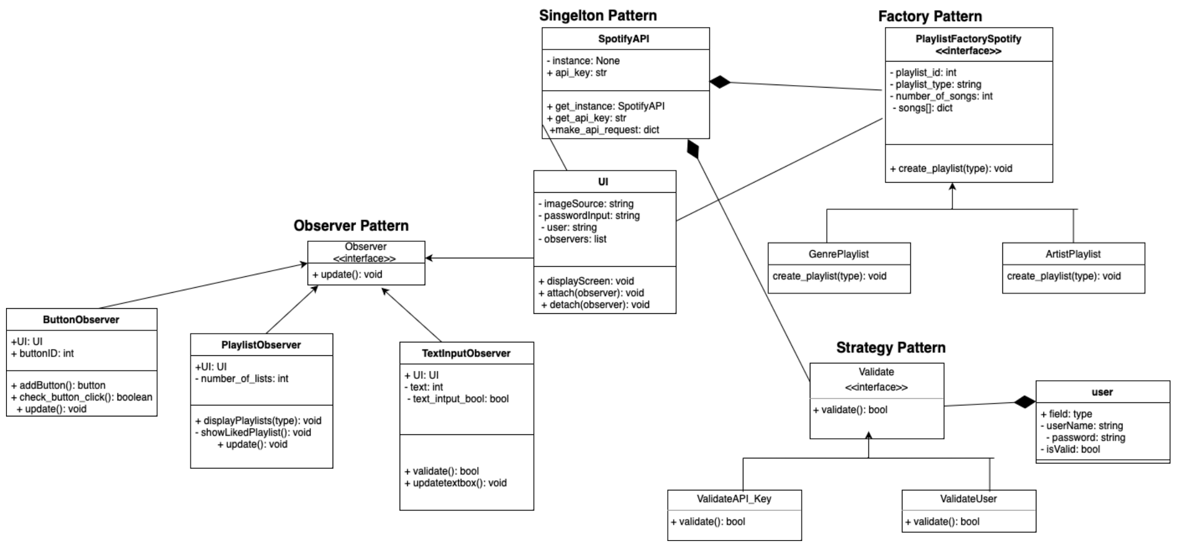
The features we implemented are removing songs from your liked playlist, sorting the songs in your liked playlist into different playlists based on either genre, artist, or mood, and letting the user choose which of those playlists to actually create in Spotify. The only feature we did not implement is removing the songs from your liked songs playlist after using one of our sorting methods to create a new playlist with those songs. The reason we did not implement that is because we realized that it didn't really make sense to do that because you might want to sort by genres and mood for example and create playlists with songs in both categories. That is why we added the feature to remove songs individually.

3. Final Class Diagram and Comparison Statement (Zoom in to have more readable text)



White Color: Observer Pattern
Yellow Color: Factory Pattern
Purple Color: Singleton Pattern
Blue Color: Strategy Pattern

Class Diagram From Project 5



Changes from Project 5 to Final Project:

As we stated in the video, one of the more challenging things for this project was coming up with an accurate UML diagram from the beginning since we did not do sufficient research on the technologies that we were using. So given that out UML and class structure has changed a lot since the beginning. To start off, we were thinking that we would use the strategy pattern to authenticate users ourselves, however, since spotify takes care of the authentication and the only thing that we are responsible for is generating the OAuth token, that idea had been changed to us using the strategy pattern to handle the sorting algorithms of the liked songs. The singleton pattern stayed basically the same, where we were using it to store the keys and the methods to always have an updated key that would allow us to interact with individual user data. Since we were using html to code the front end and a very lightweight python application flask, to handle the data between both, we ended up using an observer pattern to keep track of updates in the liked songs list, so that every 30 seconds a new list is fetched from the api and passed on to the observers from the subject. This allows us to have a real time list when the app is running, so if we like a song while we are using the app the list will be able to fetch that and process that data. Lastly we ended up using the factory pattern to fetch individual pieces of data from the returned Json instead of creating the playlists, since the playlists are also created by spotify.

4. Third-Party code vs. Original code Statement

All of the code in our project is original code. We did not use any third-party code other than methods from Flask, Spotipy, and the Spotify API.
Flask: https://flask.palletsprojects.com/en/3.0.x/
Spotipy: https://spotipy.readthedocs.io/en/2.22.1/
Spotify API: https://developer.spotify.com/documentation/web-api

5. Statement on the OOAD process for your overall Semester Project

Goal:

We had a pretty clear goal in mind going into this project where we knew that we wanted to be able to sort the songs in the users liked songs playlist and interact with the actual spotify app all while having a UI so that the user can interact with a more visual interface vs a command line interface. This was a very positive thing since it allowed us to decide what tools we could use to help us achieve these goals, and when the tools and ideas were changing we would have a clear understanding of why they were changing since we knew where we were going. It also allowed us to create a good plan upfront since we had good foundations of the goal.

Design:

Designing the architecture for this app would be a bit more challenging. Choosing the technologies was somewhat simple since we knew that our project was on a smaller side, so we would not have to worry about scaling, meaning we could pick lightweight technologies. We ended up sticking with all of our initial tech decisions, so that was a positive aspect. However once we got to designing the patterns for the classes and the actual code architecture, we thought we had it all figured out, however once we got to coding, we started to realize that some of out designs just did not make any sense and would not work. So we ended up doing a lot of design changes while working on the project, making us somewhat feel that we had wasted some time at the beginning of the project since we thought pretty hard about this problem. So that was a bit disappointing and we need to see how we can improve completion of designs vs throwing the original design away and coming up with designs as we go.

Plan:

Given that we had a well thought out goal, we knew the steps we would have to take to continue the process. We knew that we would need to get everything authenticated and start fetching user data, and start building an interactive app before we could even worry about any sorting or displaying. So once we had all that figured out, we could start seeing what features we wanted to start adding and how we would add them. So overall having a good plan allowed us to set good time goals and get a good working product in on time.