Project 7 Final Submission

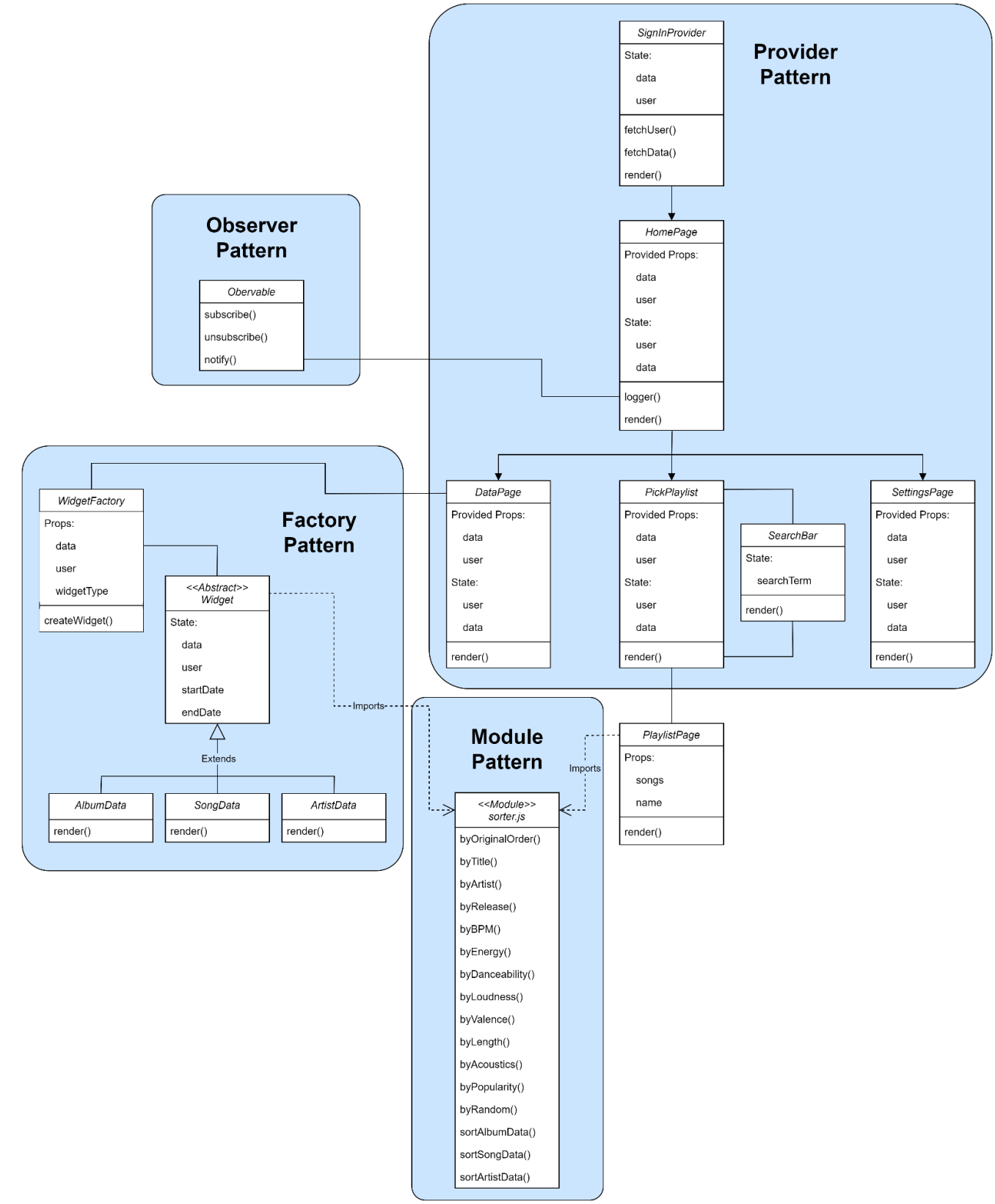
Project Name: Tune Tidy

URL: tunetidy.com

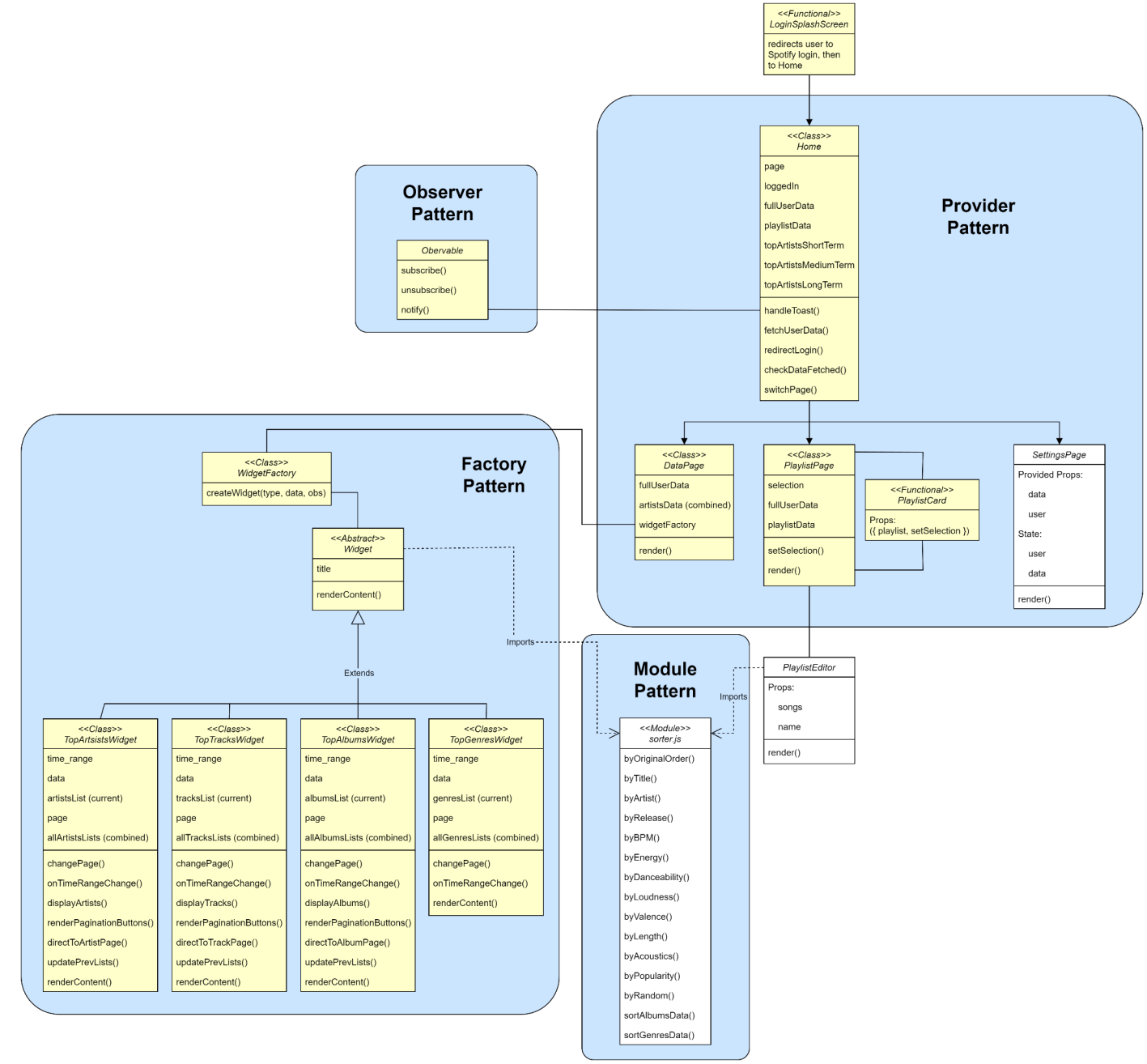
Team Members: Richard Roberson, Cebastian Adams, Rebecca Rasmussen

Final State of System Statement: All features were implemented except the ability to add/delete songs from a playlist due to running out of time before submission and this proving to be challenging to implement. This includes using the Spotify API and Authentication for login, sorting of playlists and overriding changes to the spotify library, displaying users' listening data over different time periods and having an account page. The only major change was from Project 5 where we no longer use a database and instead use local storage as we only need to save the user's authentication token. No changes were made from project 6.

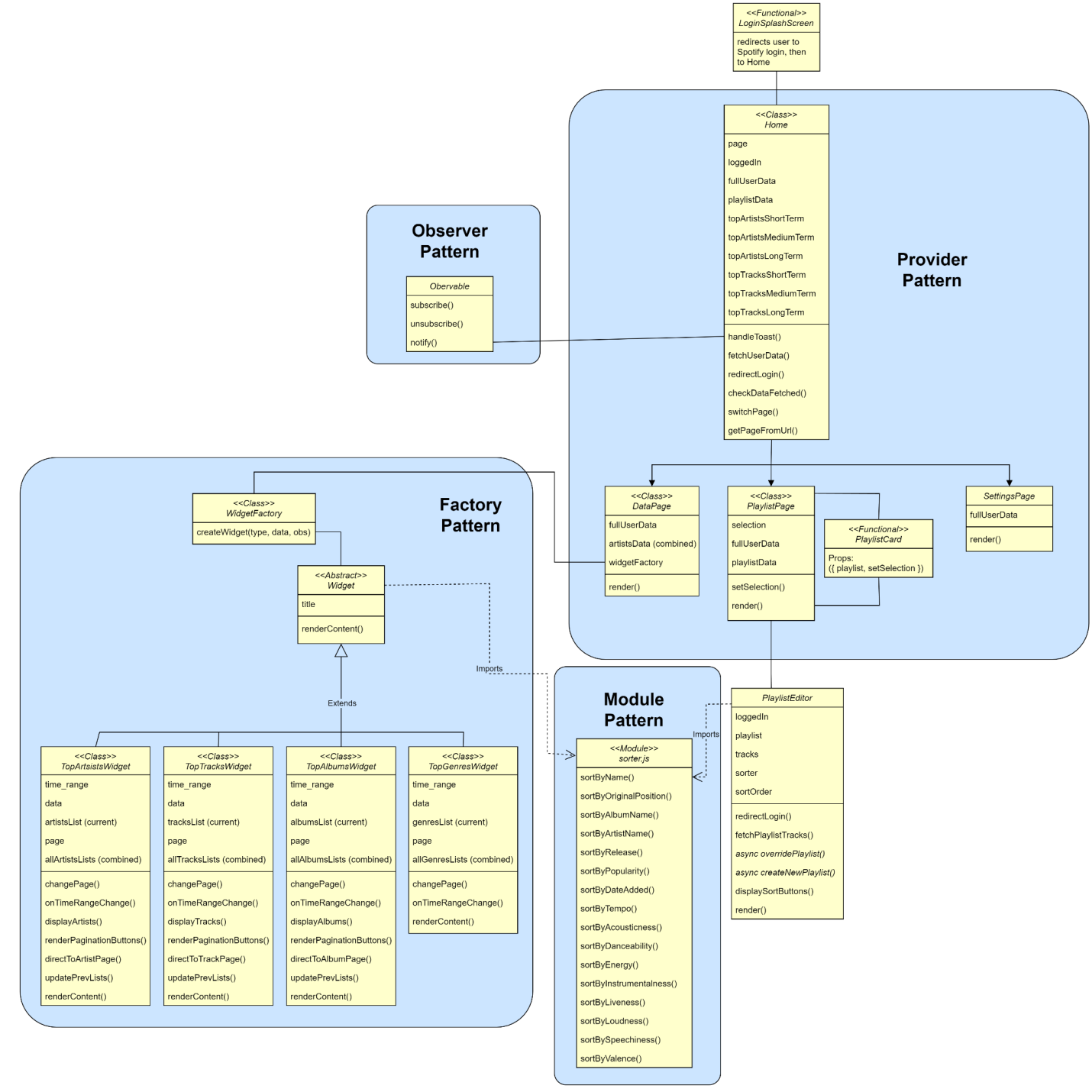
Project 5 Class UML:



Project 6 Class UML:



Final Project 7 Class Diagram:



Final Project 7 Class UML Compared:
During the development of app since the submission of Projects 5 and 6, there have been no major changes to the overall structure of the UML class diagram. The most significant adjustment involved removing the LoginSplashScreen from the Provider pattern between projects 5 and 6. This decision was made because we shifted the API calls to the Home component, making the LoginSplashScreen no longer responsible for providing data. As a result, it was removed from the Provider pattern.

Another alteration was the elimination of the SearchBar that we had initially planned to include. Upon further consideration, we realized that a search bar was not necessary for our app. Apart from these modifications, the rest of the changes to the UML diagram have been a natural progression, with new class methods and attributes being introduced as needed to accommodate features that we had not initially anticipated requiring those methods or attributes.

Third-Party code vs. Original code Statement: Here are URLs where snippets of code were taken from outside resources.

- In the top genres widget, the code in the following URL was used to sort a dictionary in javascript:https://www.educative.io/answers/how-can-we-sort-a-dictionary-by-value-in-javascript
- In the top genres widget, used the following code to make a pie chart in recharts: https://www.tutorialspoint.com/reactjs-how-to-create-a-pie-chart-using-recharts
- Different pattern implementations in React that we got our patterns from, https://www.patterns.dev/posts
- Chakra UI Framework https://chakra-ui.com/
- Used various Chakra UI component templates (similar to a css framework, but for React components) https://chakra-ui.com/docs/components
- Navbar template that we edited to fit our needs https://chakra-templates.dev/navigation/navbar
- Template for playlist cards, with the nice shadow design https://chakra-templates.dev/components/cards
- Spotify Web API JS (npm package) https://jmperezperez.com/spotify-web-api-js/, https://www.npmjs.com/package/spotify-web-api-js
- React docs: https://legacy.reactjs.org/

Statement on the OOAD process for your overall Semester Project:

Our group experienced a fair amount of different issues and positives while we were developing this project. One of them was initially caused by the provider pattern. Spotify limits the amount of requests a developer can make in a certain amount of time. Normally this wouldn't be a problem but because we were using the provider pattern and collecting all of the data at once, it would not complete the requests. After optimizing the code and making it so that the amount of requests were still within the limit provided by spotify, it worked. Another negative we faced was that it was particularly hard to implement the factory pattern with react. When using react it felt as if it struggled

sometimes with auto updating the information for the widgets. Once we understood the specifics of react, we were able to proceed with development. A positive that we all felt coming from this experience was that we were able to actually apply various OOAD principles in an application setting. Using abstraction and more class based code was something all of us had less experience with prior to this class. Therefore changing how we view the development process and what we need to implement in a team setting was very useful.