Jasmine Glaze, 2953164

Logan Michels, 2927340

Sean Cunningham, 2935773

Josh Berkley, 2945294

Suhaib Ansari, 2957919

Team 5: Final Project Design

Project Name:

SwapStream

Project Synopsis:

Web application that seamlessly integrates various music streaming platforms.

Project Description:

The project that our team has decided on deals with integrating various music streaming platforms for easy user access. Our project has the goal of giving the user the ability to access all songs on one platform and to seamlessly integrate the music they are listening to into one space. This solves the problem of having to utilize multiple different streaming sites to get all the music they are wanting and allows the user to be able to listen to their music in a faster and more streamlined manner. This will be accomplished through combining various existing platforms (such as Spotify, Apple Music, SoundCloud, etc) by using their API's so that all the songs will be in one place. Our current execution plan is to utilize a combination of typescript, css and html to create a web application for easy user access. Later on this project could lead to the development of a mobile application with the same functions as the web application, but with more ease as it could be utilized more readily on a mobile device.

Project Milestones:

- Gantt Chart:

Task:	Oct	Nov	Dec	Feb	Mar	Apr
Determine and Gather APIs	Oct-15th					
Set-Up Basic Web Application		Nov-15th				
Integrate API's			Dec-15th			
Debug/Add Functionalities				Feb-15th		
Stylize User Interface					Mar-15th	
Finish Documentation						Apr-15th

- Fall 2021
  - Oct 15th: Determine and gather the API's for each streaming service that we intend on integrating (such as Spotify, Apple Music, etc)
  - Nov 15th: Set up basic web application (no extra design, just bare bones) to get a baseline for the rest of our project
  - Dec 15th: Integrate API's and music functionality (start/stop button and a way to select various songs)
- Spring 2022
  - Feb 15th: Debug program and add in extra functionalities for users ease (such as a replay button, a way to add songs to a playlist, etc.)
  - March 15th: Web application user interface complete with styling (make the web application look professional and add graphics as necessary)
  - April 15th: Finish all documentation and any additional materials
- Defining roles:
  - Jasmine Glaze - documentation/status reports/additional assignments

    - Logan Michels - documentation/assist with coding and assignments as needed
    - Sean Cunningham - back end coding and database management
    - Josh Berkley - connecting front end and back end (working closely with both)
    - Suhaib Ansari - front end coding

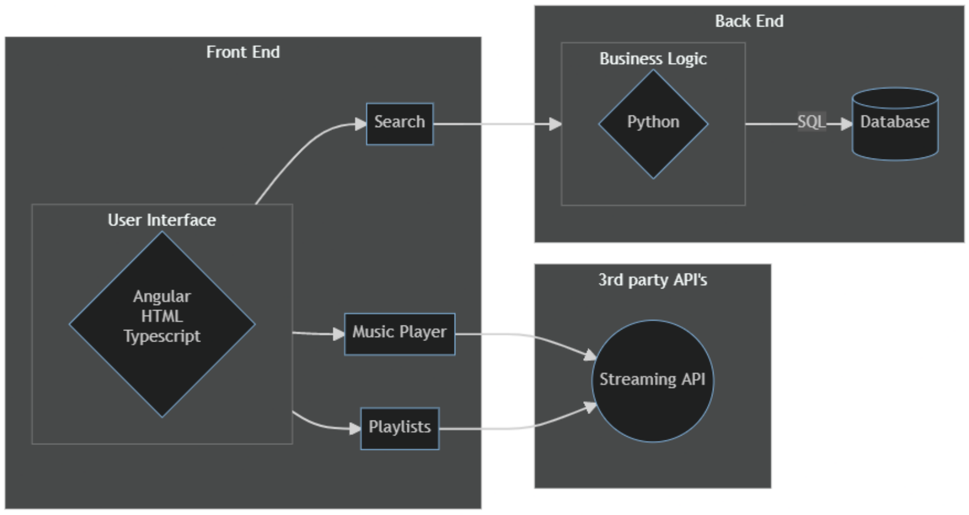
Project Budget:

The cost of this project would be negligible because the APIs along with the platforms to utilize these languages are free. There will also be no vendors or special training needed. We will host our service on Amazon Web Services, which is not a free service, but it does begin free so we will not need to make a purchase unless we exceed a certain amount of computations. This is highly unlikely during our testing and would only occur after gathering a real user base.

Final Project Design:

- Back End Software:

  Because our project will include lots of user interaction with our web application, we are creating an extensive front and back end in our software to create something that is highly functional but also easy for the user to interact with. When user data is needed or user data is produced, the front end will offload the processing work to the back end by sending requests to our back end API. This will be handled through the REST standard for requests. The back end will handle data processing and communication with the database. This separation of concerns is a key point of our design. This will leave the front end free to focus on fast rendering speeds and streaming the music. Our back end service will be written in Python and there will be three divisions of labor: the API which handles requests and sends responses, the main business logic which will handle all data processing, and the DAO (Data Access Object) which will send and receive data from the database. The database will most likely be PostgreSQL. We will have locally hosted testing databases, but the official one will be hosted on AWS servers.



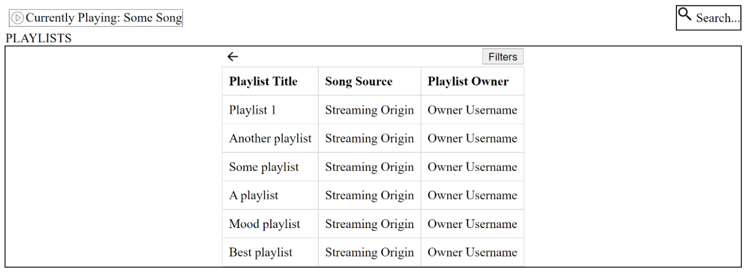
This architecture is a 3-tier model with an added 3rd party API configuration that enables access to a user's streaming service of choice. To circumnavigate some limitations with some of the streaming service API's involved, the database is used to store playlists associated with a user's account in said streaming service. For example, if a Spotify user wants to create playlists that belong to an Apple Music account, the search feature will look through the database and retrieve the playlists in a JSON package that can be used to generate a playlist in the user's library for their usage.

- Front End Software:

  Although the back end is very important for this project in terms of functionality, we also took a lot of consideration when creating a plan for the front end. This is due to the fact that users will be constantly coming back to the web application to play songs, choose new music etc. and this needs to be something that the user can use easily. Before a user can actually use the central playlist sharing feature of this application, new users are directed to what type of account they use. They then log in through their streaming service. Beyond that, the design of the front end is simple with three primary components: in-browser player, search bar, and music display. The display contains the list of playlists a user has with the service which includes any they upload and any they get from our database. The name of the playlist, the owner's username associated with the streaming service, and the original streaming service (Apple Music, Spotify, Soundcloud, etc.) will be displayed on each playlist. A user can click on a playlist which will then change the list to the songs inside of the playlist while displaying the name of the playlist at the top of the display module. During search, the playlist module will show results of playlists based on the search. These can be filtered by streaming services or sorted. The other module is for playing music in your playlists, but can also be used to load a preview playlist that can be listened to before adding it to their library. If they like it, they can add it with search. Users can search for playlists based on genre, artists, users, and other data that the streaming service API allows to view and categorize songs and playlist.

Display View:

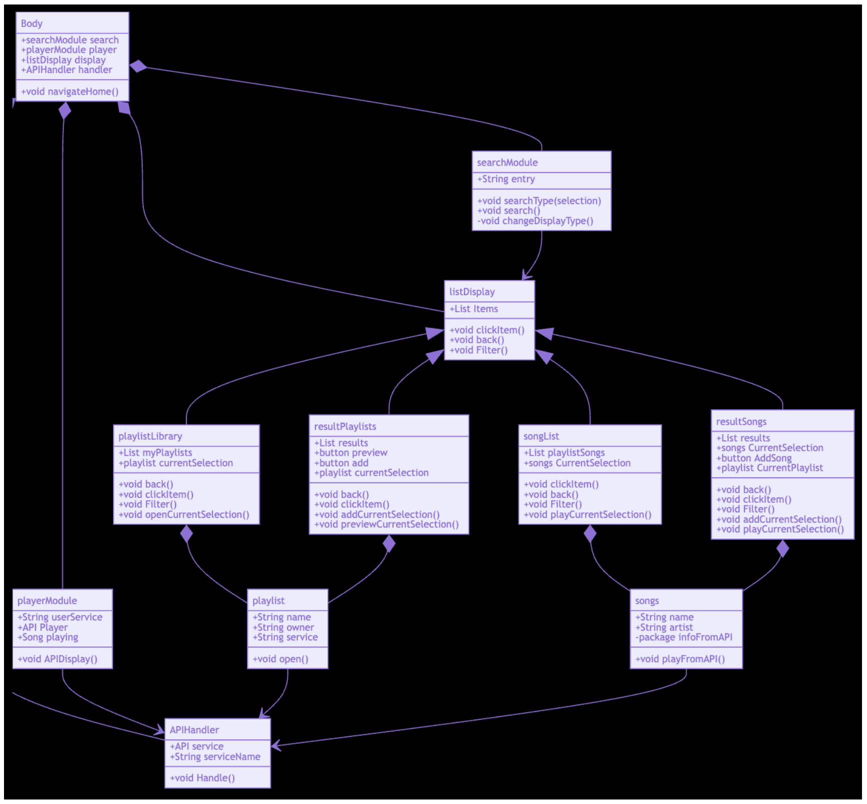
This screen diagram shows the view of the user and the options they have for navigating between playlists. There are 4 types of lists that can be displayed: library playlists, songs in a selected library playlist, search result playlists, songs in search result playlists/songs in search results (which function the same). When playlists are in the library



Playlist Title	Song Source	Playlist Owner
Playlist 1	Streaming Origin	Owner Username
Another playlist	Streaming Origin	Owner Username
Some playlist	Streaming Origin	Owner Username
A playlist	Streaming Origin	Owner Username
Mood playlist	Streaming Origin	Owner Username
Best playlist	Streaming Origin	Owner Username

mode, the playlists owned by the user are displayed and can be opened. When opened, the list will transition to the songs in that list. A user can click a song to play it in the above embedded player. When playlists are in the search mode, there is an additional option to add the playlist to a library and an option to preview the playlist which will play all of the songs in order. The playlists can also be clicked to reveal the songs within. When adding to the library, the songs will also be previewed with the option to remove any tracks before adding as well as warnings if a song isn't available in the user's streaming service. Even if the user doesn't remove it at this time, the song will automatically be deleted from the copied playlist. Below is the class diagram for the front end.

Frontend Class Diagram:



The classes involved here are broken down based on user interactions with the components of the user interface. Three of the items that link to the Body class represent the modules directly present in the front end namely "searchModule," "playerModule," and "listDisplay." The classes linked to them below control the states of the web page through Angular and Typescript. The fourth class, APIHandler, is invoked when another module needs to resolve a request to the streaming service API. The other classes are encapsulations of the data provided to the users. The types of playlists all inherit from the "listDisplay" class, an approach that uses polymorphism to invoke the functionality for the given list type through class design.

- Design Constraints:

  Some design constraints we may run into include the typical technical examples such as programming language, operating system or platforms supported and use of a specific library or framework. In regard to programming languages, because we are still college students, we do not have experience with every language out there, so we might come to find out that there may be a better suited language for our project that we do not have in our repertoire. This could be a constraint because our abilities and overall knowledge of programming are limited. This sentiment also aligns with the operating system and library or framework that we are going with. We only know the resources that we know and we could come to find that we overlooked an option due to our lack of experience. In regard to business constraints, our team could run into issues in terms of scheduling, budget, team composition and licensing. Many of these are especially true when it comes to this sort of project since we are all in college with different schedules, goals and personalities. With our schedule and team composition, we are going along with the deadlines that are set for us and trying to accomplish our tasks together while trying to balance our other commitments and find times that work for everyone. In terms of budgeting and licensing, these issues could arise, but for now they seem to be non-issues.

Ethical Issues:

This project has few ethical issues due to the purpose of this project being based in creating a better and more user friendly streaming application for users. We will always do our best to be honest and trustworthy and keep the principles of the code of ethics in mind throughout the creation of our project. One ethical issue we might run into, however, is not taking the time to "respect the work required to produce new ideas, inventions, creative works and computing artifacts" (principle 1.5). We need to make sure we take the time to credit those we are basing our project off of and give them credit for their original ideas. As long as we do this and continue following the code of ethics, we should create a good and ethical final product.

Intellectual Property Issues:

This project has few intellectual property issues due to the nature of all the music we are wanting to use being publicly downloadable/accessible. One issue we might run into, however, is in terms of the intellectual property of the streaming applications themselves. We will need to be sure that we are careful with how similar ours is to the ones we are basing ours off of as to not steal their intellectual property. This will be mitigated by us not having the source code of these streaming applications and our lessons we have gleaned through our labs in this class over intellectual property.

Changing Log:

- Section: Defining Roles
    - Change: Previously we had said that we were not creating roles at the time of the project description, because we wanted to see how our roles naturally developed and not force anything so early on. Now we have created roles based on what

    each team member is gravitating towards and this is reflected in this modified section.

- Section: Project Description:
    - Change: Added more detail to the project description to ensure that it meets the minimum word count.