Milestone 4

Revised List of Features

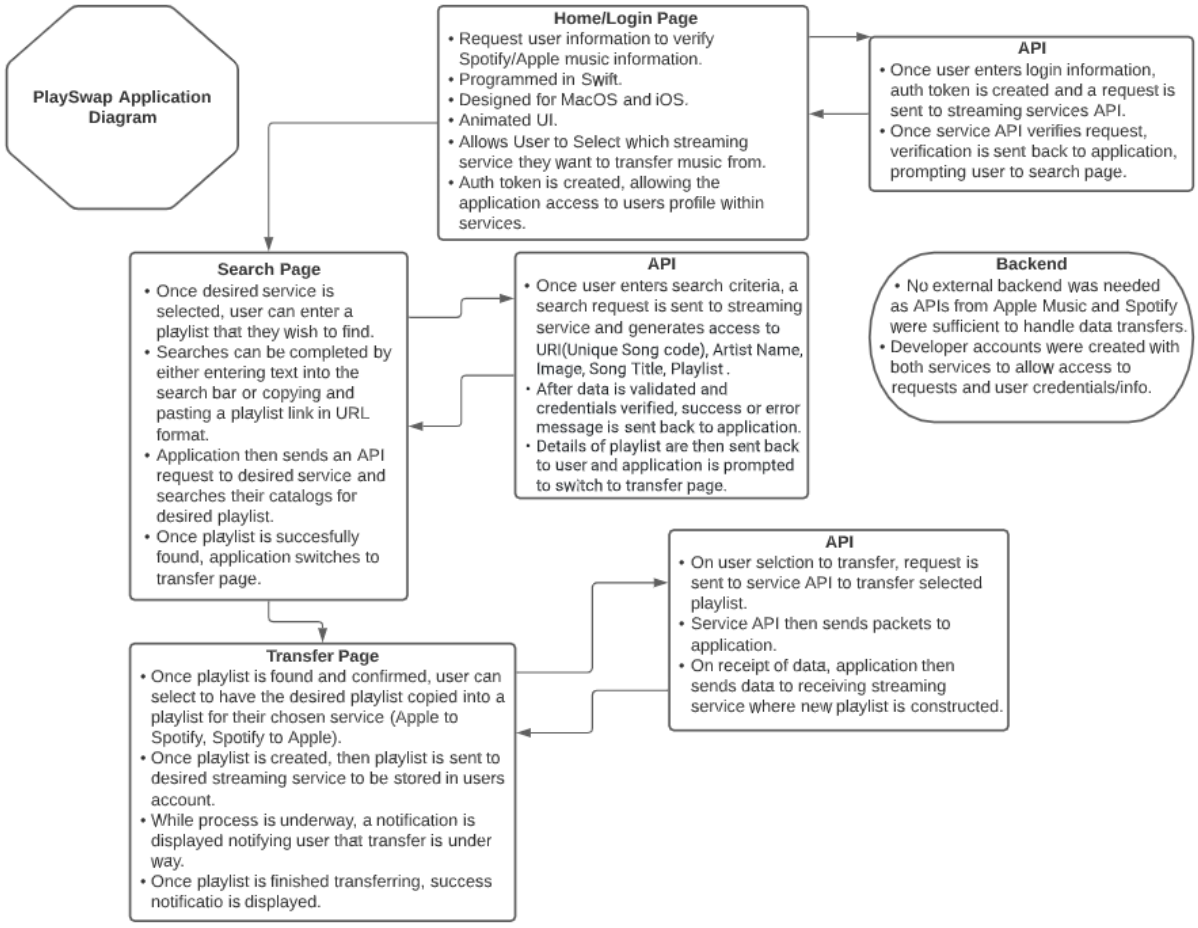
Complete Features

- User authentication: Both services
    - Users can login to both Spotify and Apple music. Upon opening the app, the user can click the icon of their music streaming service and login to their account.
- Transfer playlist (Apple Music to Spotify)
    - Users can lookup public apple music playlists and add them to their library.
- Playlist link support
    - Playlists can be shared by sending links to others. Users can take a link sent to them, paste it into PlaySwap and add that playlist to their library. This works for Apple Music links.
- Spotify playlist retrieve
    - Using the Spotify API a user's playlist can be retrieved from their library.
- Apple Music playlist retrieve
    - Using the Apple Music API a user's playlist can be retrieved from their library.
- Spotify create playlist
    - PlaySwap can create playlists in a Spotify user's library
- Search for songs
    - Backend searches for songs in the Apple Music and Spotify libraries to add to the playlist
- Search for playlists
    - Can search for public playlists

Incomplete features:

- Creating playlist in Apple Music
    - Creates a playlist in the Apple Music user's library from list of spotify songs
- Saving authentication keys for users on device
    - Allows users to not have to login each time they use app
- Adding only certain songs from a playlist to library
    - Allows users to take only parts of a playlist they want
- Pasting spotify links into the search page instead of manually looking up playlists
    - Gives the application more functionality.
- Reporting back to user which songs could not be transferred
    - Lets user know if there's been a problem
- Making sure songs are transferred in right order
    - User may want the playlist in the exact order it came in
- Show "loading login" when signing into apple music
    - User knows what is happening and that the app has not crashed

Architecture Diagram

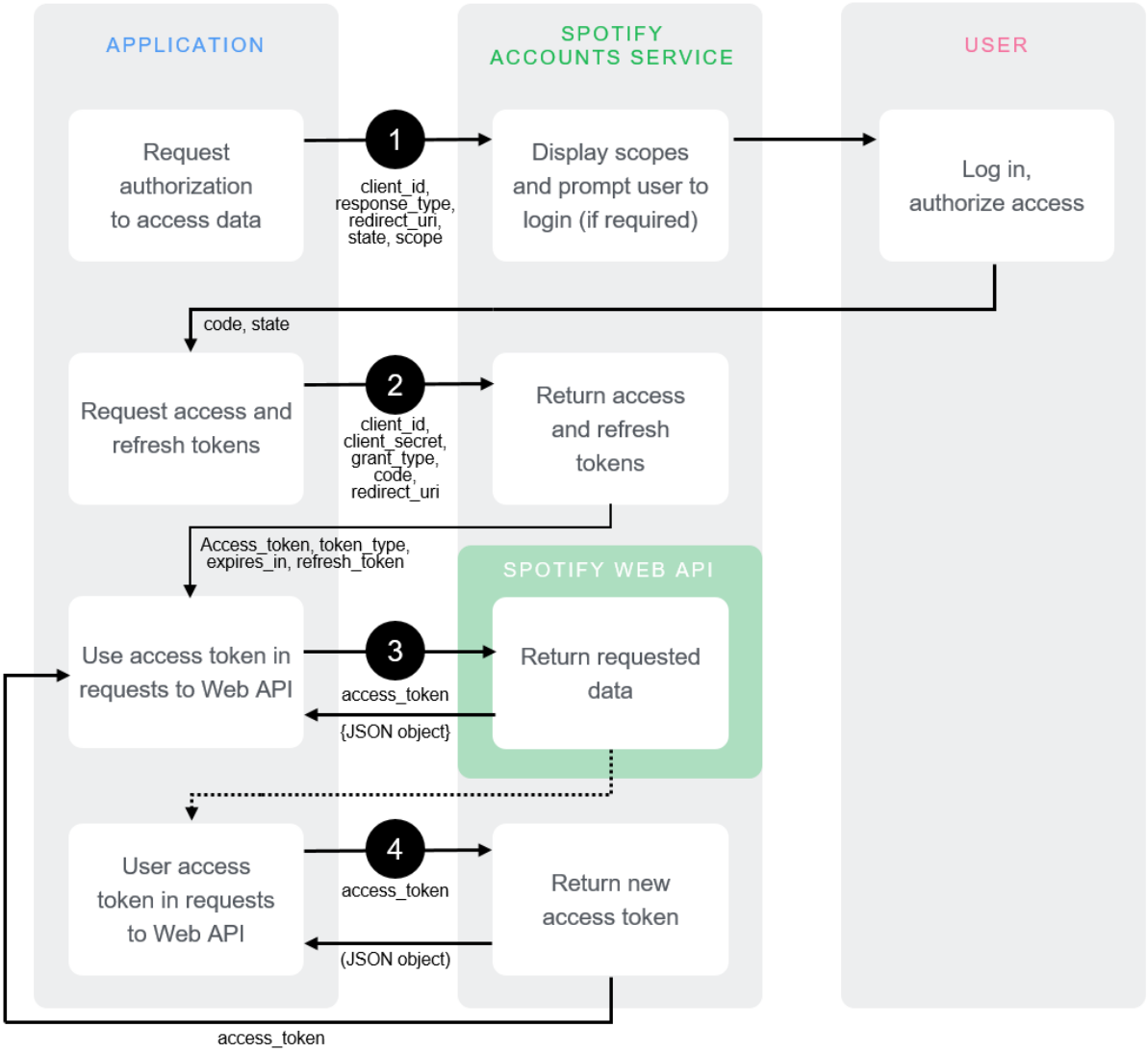


Front End Design Logan
https://drive.google.com/file/d/1Y9yw5-nRk1OuLCbpRab3tR_P5gH7Tteu/view?usp=sharing

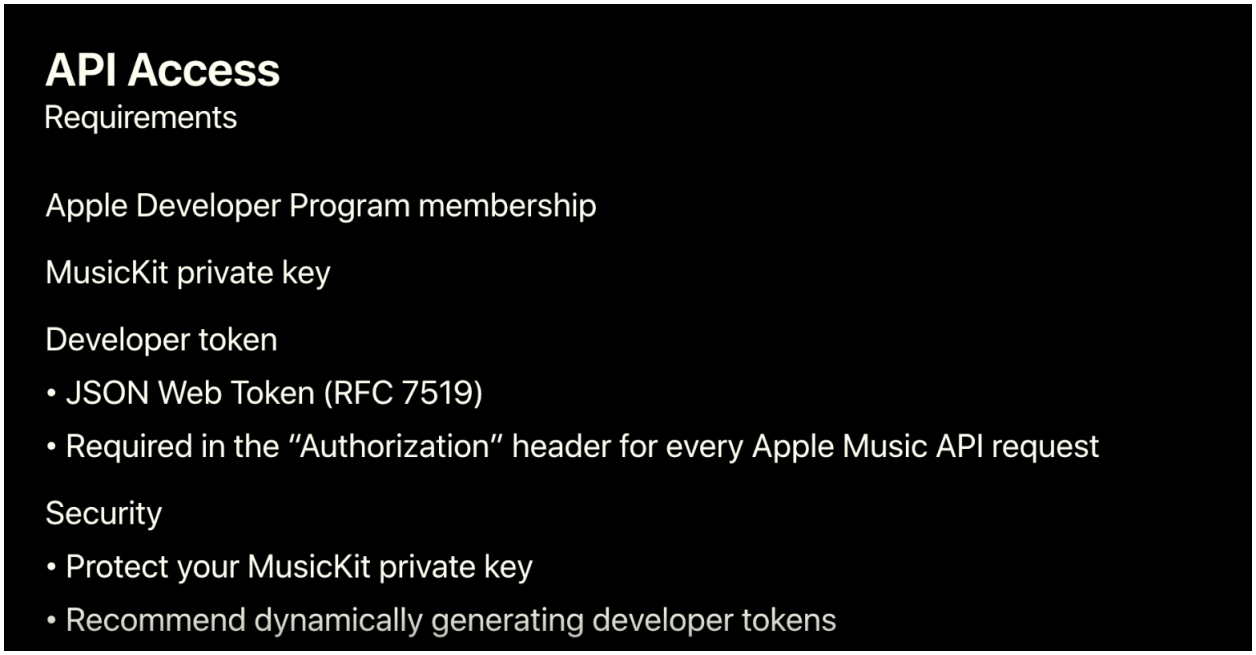
Web Service Design Jordan

Login Architecture

Spotify



Apple Music



- *Apple music has a unique authentication flow where its authenticated locally with the "music" app and no actual login is required but the authentication token is passed back through the app*

Search Playlist Architecture

Spotify

```
GET https://api.spotify.com/v1/search?q={query}&type=playlist
Headers: "Authorization: Bearer {auth token}"
Response Types
If error:
{
  "error": {
    "status": 401,
    "message": "Invalid access token"
  }
}

Else:

Returns array of Playlist<playlistItemsReference> as outlined in database
design
```

Apple Music

Database Design Clay

- PlaySwap utilizes many attributes from public API keys from Apple and Spotify, respectively. These attributes are *necessary* for our software to interpret each song, complete with information about artist, album cover, and song name. We additionally need to gain the information from each playlist created in either Spotify or Apple Music. These playlists are composed of, not only the aforementioned information with each song in the playlist, but additionally a playlist title, description, and cover photo.
- It is worth noting that our application is utilized via Swift, and uses such software to operate.
- Fields (Attributes):

`var spotify_anonymous = SpotifyAPI( authorizationManager: ClientCredentialsFlowManager( clientId: "", clientSecret: ""`	Credentials for SpotifyAPI
`var playlistItem: Playlist<PlaylistItemsReference>!`	PlaylistItemReference
`appleMusicPlaylistItem: [JSON]!`	Apple Music Playlist Item
`@IBOutlet weak var searchResultsViewController: UITableView! @IBOutlet weak var playlistContentsTableView: UITableView!`	Search Results and Playlist Contents
`var backButton = UIButton()`	Back Button, attribute needed in essentially page, with few exception
`var spotifySearchResults : [Playlist<PlaylistItemsReference>] = [] var appleMusicSearchResults : [AppleMusicSong] = []`	Search results for Spotify Playlists
`var playlistTracks : [PlaylistItem] = [] var appleMusicTracks : [AppleMusicSong] = []`	Song information of each track in the playlist stored in variable
`var spotify = SpotifyAPI(authorizationManager: AuthorizationCodeFlowManager( clientId: "", clientSecret: ""`	Authorizations
`var spotify_anonymous = SpotifyAPI( authorizationManager: ClientCredentialsFlowManager( clientId: "", clientSecret: ""`	More Authorizations

- Files (Data Entities):

init(id: String, name: String, artistName: String, artworkURL: String, length: TimeInterval) {
self.id = id
self.name = name
self.artworkURL = artworkURL
self.artistName = artistName
self.length = length
}
	Playlist entities for Apple Music

var emailField = UITextField() var passwordField = UITextField() var webView = WKWebView() var loginButton = UIButton()	Login Information
var playlistTitleLabel = UILabel() var playlistImage = UIImageView() var playlistDescription = UILabel() var playlistAuthor = UILabel() var playlistAuthorImage = UIImageView() var playlistTransferButton = UIButton()	Playlist Information in Transfer Page
var playerLayer = AVPlayerLayer() var transferringFrom = "" var songQueue: [SpotifyURIConvertible] = [] private var cancellables: Set<AnyCancellable> = [] var transferType = "spotify" //spotify or itunes var currentStep = "choose_service"	Variable entities affecting UI control
webView.navigationDelegate = self view.addSubview(webView) webView.frame = CGRect(x: 0, y: 0, width: UIScreen.main.bounds.width, height: UIScreen.main.bounds.height) playVideo(from: "Mobile_Web_BG.m4v")	Making webview visible

Challenges

1. Swift can only be run on a Mac OS. Most of our group owns a Mac computer but two of our members do not. Without a Mac it is impossible to run our project. Preston has downloaded a VM to be able to work on the project. Jack uses the Macs at the library.
2. UI bugs and crashes. To have a complete project, bugs must be kept to a minimum. We have been purposefully trying to crash the app to discover any problems with our code.
3. Apple music playlist creation. This is a big part of the functionality of the project. This needs to be finished in order for the project to be complete. There are multiple weeks left for this to be completed and should be done before the end of next week.

Individual Contributions

Jordan - Worked with Clay and Logan to fix a bunch of ui bugs. Also worked on bugs that caused crashes. To prep for this week also created base functions/created architecture outline for the apple music transfer code.

Clay - First, added the initial UI elements and limited functionality for the search bar after signing into their Spotify accounts (searching apple music playlists). Then created the earliest version of the transfer button, to transfer public playlists from Apple Music to Spotify.
Latest commit -
https://github.com/jwoody02/PlaySwap/commit/e086a79c4db6d98b9f95c5604c8e71009aa42635

Logan - Created and added functionality for apple music specific links to be pasted into the search page and immediately bring the user to the transfer page per the link. Works on private playlists when the search API cannot fetch a private playlist from the user search input. Found UI bugs pertaining to apple music login and the 'back' button. Identified and fixed bugs accordingly.
Latest commit (link functionality) -
https://github.com/jwoody02/PlaySwap/commit/3a1868264b3a2cdcbad1f4fb804a9265dbf9b0ce
Latest commit (bug fixes - pushed by Jordan) -
https://github.com/jwoody02/PlaySwap/commit/1940fc40c38c9c94110a3bd3a0bfe9c03dcda6d0

Jack - Worked on playlist creation within an Apple Music user's library. Not functional as of now but will be added at the next team meeting. Creates a playlist in the user's library then iterates over a list of spotify songs and adds each one to the created playlist. Uses a search function to find the apple music equivalent song of a spotify track. Unable to test due to library computers not having the latest version of Xcode.
Latest Commit: Created own repository to not add potentially faulty code to functional program.
https://github.com/higginsjack/MacPlaySwapRepo/commit/89b9ac2c74a0a6ed75dccbefac727458710e82ce

Preston - Due to not owning a Mac and not having easy access to computers at school due to living in Denver, much of my contributions have been in the form of accessory tasks such as working on Milestones, researching API protocols and designs, and assisting in design meetings to help guide UI features and QA testing. However, I do now have a VM capable of running MacOS on my PC now and will be able to assist in more development in the coming weeks.