CSE 115A – Introduction to Software Engineering

Release Summary

Product: UNUS

Team Name: BIGAS UNUS

07/25/2023

Key user stories and acceptance criteria:

User story 1.1: As a user I need a website that I can interact with.
1. Become somewhat familiar with Flask (3 hours)
2. Create the basic shell of a website using basic HTML frontend and Flask backend (5 hours)
Acceptance Criteria: Shell of a website with some buttons, sign in and logout functionality. Login and sign up tested and worked as intended.

User story 1.2: As a user I would like a profile that stores my information to log back into.
1. Make sign in/sign up buttons that redirect the user to either a login or profile creation page (1 hour)
2. Set up a database using SQLAlchemy that will store the user profile information (username password etc) and connect it to the backend. (3 hours)
Acceptance Criteria: Database stores profile information and the information is retrievable. Test that information is stored and retrieved without errors.

User story 2.1: Api connection: As an avid music listener I want to connect all my music to one place by logging into my different applications all on the same website.
1. Set up user authentication to the spotify/soundcloud API's (5 hours)
2. Create a page after signup where the user chooses which of their music app accounts they want to connect to their UNUS profile (3 hours)
3. Save their information to the database and encrypt (2 hours)
Acceptance Criteria: Spotify authorization routing works properly. Soundcloud widget can be connected to. Test the codes returned by spotify when making requests to make sure full access has been granted.

User story 2.2: Show libraries from connected API : As a user I want to be able to see the libraries from the applications I've connected.
1. Pull the user library information from the API's (3 hours)

2. Store the library information in the database (2 hour)
3. Display the libraries in the frontend (3 hours)
Acceptance Criteria: All desired songs are displayed in a relatively organized/appealing manner. Similarly to US 2.1, test the spotify connection and make sure the soundcloud widget is working properly.

User story 3.1: As a user I want to be able to play the music in my library directly from the application.

1. Implement the Spotify playback SDK (5 hours)
2. Implement the Soundcloud playback widget (3 hours)

Acceptance criteria: Music is playable from both spotify and soundcloud. Test that full playback functionality is working (i.e. pause, play, restart)

User story 3.2: As a user I want to be able to choose which applications I want to link to my account:

1. Add a page that allows the user to connect their spotify and/or soundcloud accounts to their UNUS account

Acceptance Criteria: Spotify authorization and Soundcloud user ID upload all work properly and don't break the code already implemented. All routing works properly. Users can't access this page before logging in.

Known Problems:

- Soundcloud API Authorization is missing (substituted with manual input user ID)
- User ID not being properly stored within SQLAlchemy database
- If "cancel" clicked on Spotify Authorization, still retains previous authorization
- Html widgets and spotify player embeds take long to load, slow loading time
- Naming conventions in code somewhat inconsistent
- Login notifications somewhat clunky with Html objects
- Play both softwares at the same time

Product Backlog:

- Dynamic refresh of Spotify Token needed
- Spotify Authorization integrated in player widget
- Search function for spotify songs
- Integrate more music popular platforms