PROJECT 1

Pd 9 | Team Monotunes
Kevin Li, Adeebur Rahman, Michael Cheng, James Smith

WHAT DOES IT DO?

Our program is a lyrics library where you can search up a song by its name and artist or just search through an artist's albums to find a song. Once a song is selected, it brings you to a page displaying its lyrics and an audio file can be played through the page that reads you the lyrics. You may also have an account if you want to save your favorite songs somewhere.

OUR APIs

Musixmatch - a music database API to supply our search feature and lyrics
IBM Watson Text to Speech - Takes text you send it and returns an .wav file

TEXT TO SPEECH LYRICS LIBRARY - COMPONENTS

Homepage - '/'

- Presents a set of suggested songs(top 10 trending songs)
- an intro/how to use the website
- Search bar (by artist)
- Search bar (by song)
- Logout/Login button
- Profile Page button (if logged in)
- Create Account button

Create Account page - '/create'

- Asks for profile information for first time setup

Login page - '/login'

- Form for username and password
- Has a link to the create account page

Profile page - '/profile'

- Contains list of links to your favorite songs
- Option to remove songs from favorites

Song Page - 'song?title='

- Shows song title, artist, and lyrics
- Includes link to '/artist' route
- Takes query strings "title"
- Option to the song to your profile’s favorites

Artist Page - '/artist?artist='

Shows songs of the artist by album, which are links to the '/song' route
Takes query string "artist"

style.css

Additional styling to make it pretty

db.py

Handles the database, including login information and saving music preferences.

main.py

Flask app, handles front-end navigation and operations

api.py

Handles API calls and extracting information

profiles.db

Contains user information including username, password, saved music

FILES

template.html

Search bar, login/logout button, profile button, homepage button, and compulsory copyright information from our APIs.

Other HTML files

See ‘Components’ section for what they will have in them

style.css

main.py

- **add_session(user)** - creates a session cookie for the current user
- **logout()** - Deletes a session key of a user, if one is logged in
- **root()** - provides functionality for the / route
- **login()** - provides functionality for the /login route
- **create()** - provides functionality for the /create route
- **profile()** - provides functionality for the /profile route
- **song()** - provides functionality for the /track route
- **artist()** - provides functionality for searching for a song by artist

api.py

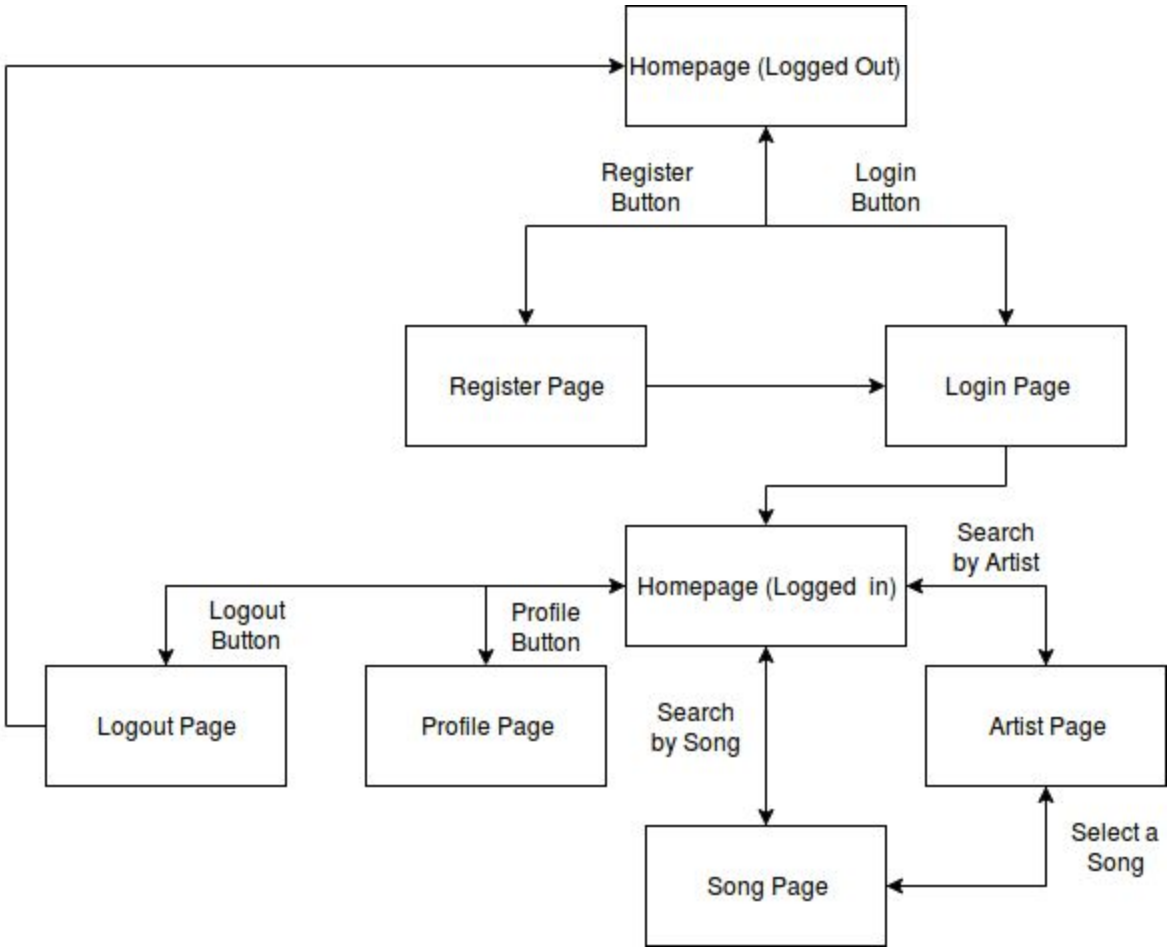
- **get_trackid(track, artist)** - returns the songid for the desired song
- **get_lyrics(trackid)** - returns the lyrics of a song given its trackid
- **get_artistid(artist)** - returns the artistid of an artist given their name
- **get_albums(artistid)** - returns a dictionary of the albums
- **get_album_tracks(albumid)** - returns a dictionary of tracks on the album
- **get_wav(text)** - downloads a wav file of speech generated from text and returns the file name.

db.py

- **login(username, password)** - Logs user in if username and password are correct

- **encrypt_password(password)** - Returns SHA-256 version of password
- **create_account(username, password)** - Creates a new entry in .profiles table
- **does_username_exist(username)** - Returns true if username exists, false otherwise
- **add_favorite(username, trackid)** - Appends trackid to current list of favorites using eval(idList) and repr(idList)
- **is_favorite(username, storyid)** - Returns true if user has this song already in their favorites

SITEMAP



DATABASE SCHEMA

Profile Table

The profile table contains 3 fields: username, password, and favorites. Username and password are used to log the user in. Username is a primary key so that you can't have a duplicate account. The last field, favorites, is a string representation of a list of track IDs (musixmatch API Song ID) that the user has selected as favorites. To edit the

favorites list, eval(trackList) can be used to get a working list, then the new ID can be appended, then repr(trackList) can be used to turn it back into a list to update that field.

Username (Primary Key)	Password *encrypted	Favorites (String of a list)
watson	ibm135	'[234678435, 238956384, 128593759, 285749384]'
sherlock	shrek	'[234678435, 573956382]'
moriarty	p455w0rd3	'[]'

PROBLEMS/LIMITATIONS WE FORESEE AND OUR SOLUTIONS

1. If a user attempts to add a favorite that is already in their list, a message will be flashed to prevent them from adding the same song twice.
2. IBM Watson Text to Speech API has a quota of 10000 characters per month which according to our approximation, gives us 8 tests of our program before it stops working. We plan to supply an alternate sound clip that says that and/or swap API keys(if approved by DW).
3. Musixmatch API only supplies 30% of the lyrics of a song. This is fine as a test case, but certainly limiting.
4. If the user searches for an artist/song that the database can't find the program will flash a message on the search bar.

TASK ASSIGNMENTS

Kevin: Project Manager and CSS
Adeeb: Backend and watson API
Michael: Frontend HTML files (helps other members when finished)
James: flask app and musixmatch API