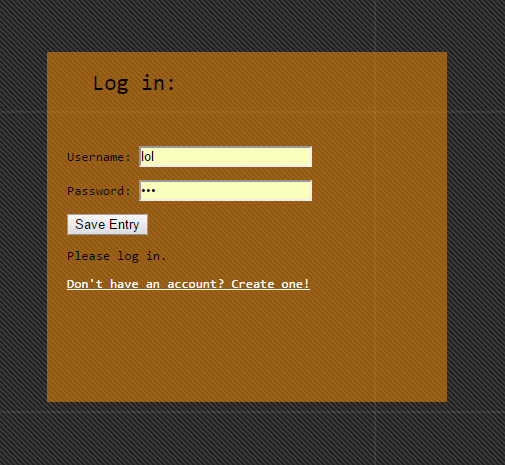
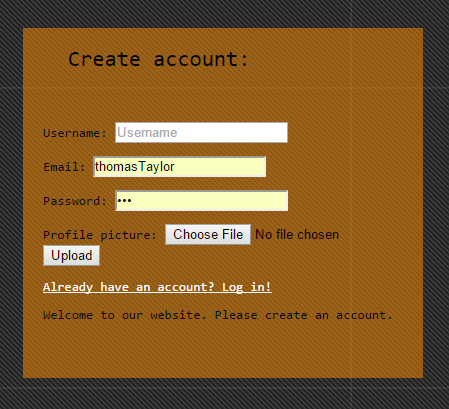
Website Technology coursework 2

The website I designed for this coursework is a media sharing platform. Users are asked to create an account before they are able to do anything. Once an account is created, every user gains full access to the website’s features. These features include adding friends, writing posts, commenting on posts, uploading music, listening to music and creating a playlist.

The purpose of this web app is to allow users to create small communities in which they can share music and keep each other updated on recent releases.

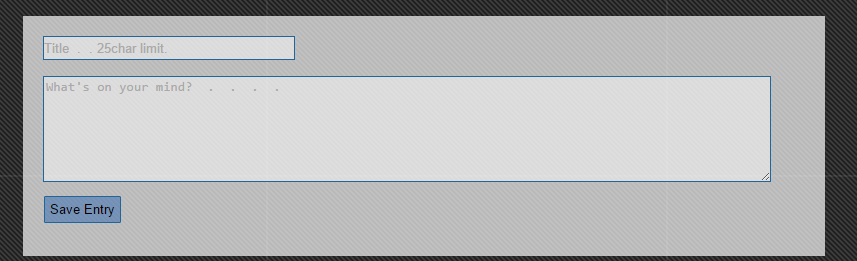
In this report we will go over the functionality of the website starting with first visit and finishing with the smaller less noticeable features. After which we will discuss possible enhancements and issues with the current app. I will then finish by explaining the major difficulties encountered while developing this program and each breakthrough.

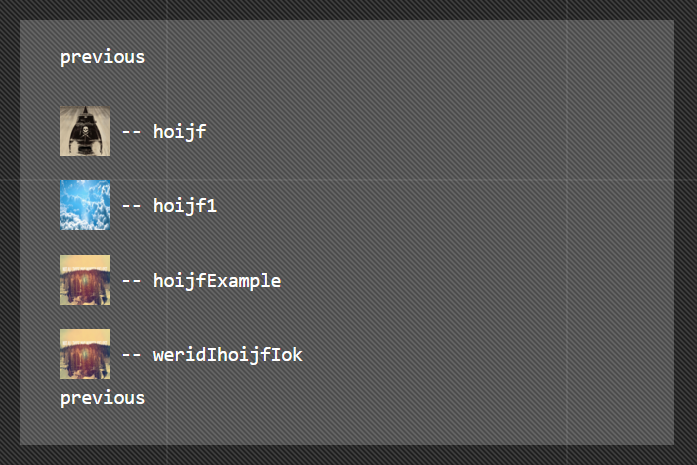
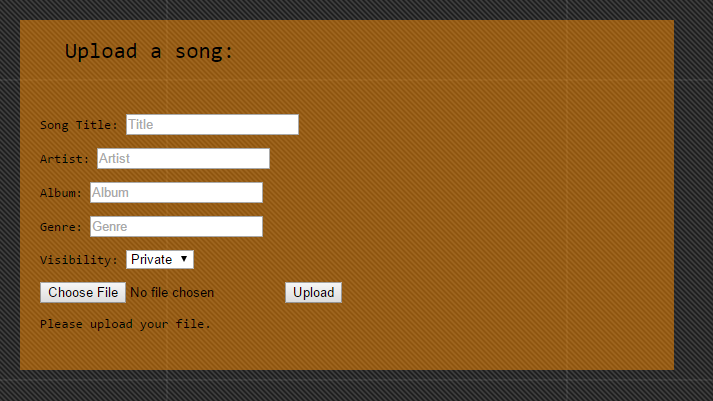
Upon first use the user is not yet signed in to any account and will get redirected to the log in page should he or she try to access any other parts of the website. Therefore the first page a user will interact with is the log in page. The website was designed this way because it is a more frequent interaction than creating a new account. However, new users are kept in mind and a link grants access to the new user page. Should a user try to log into an account which does not exist the website will simply reply with “wrong username or password”. This way hackers have more difficulty knowing if an account exists and therefore if it is hackable.

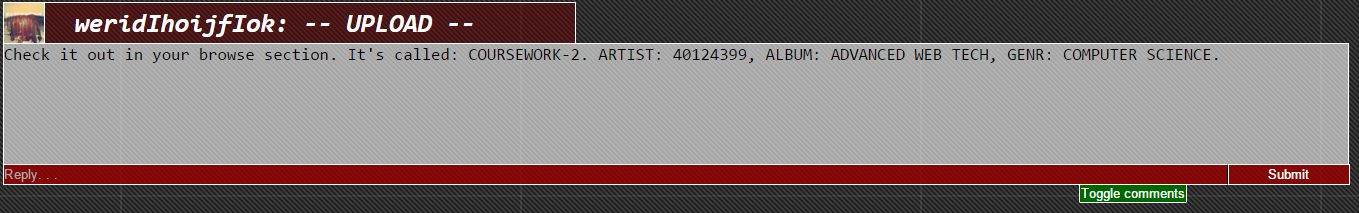
Once the user gets access to the sign up page he will be granted by a form. This form asks for the desired username, user’s current email a password and a picture which will be uploaded to the server. Here if the user tries to create an account with an existing username the app will reply with “Username already in use please select a different one”

This is a way in which hackers do get access to the information but it is valuable information for the user. Once the form is filled correctly the user will automatically be signed in and redirected to the home page. Upon first visit this page has very little to offer. You may write posts and see them. You will also be able to see posts from people who have friended you and comment on any post you see. First let’s add some friends so they can also see our posts. To do this, we use the search bar in the header. Type the username of a friend and a list of usernames will appear, for example: hoijf.

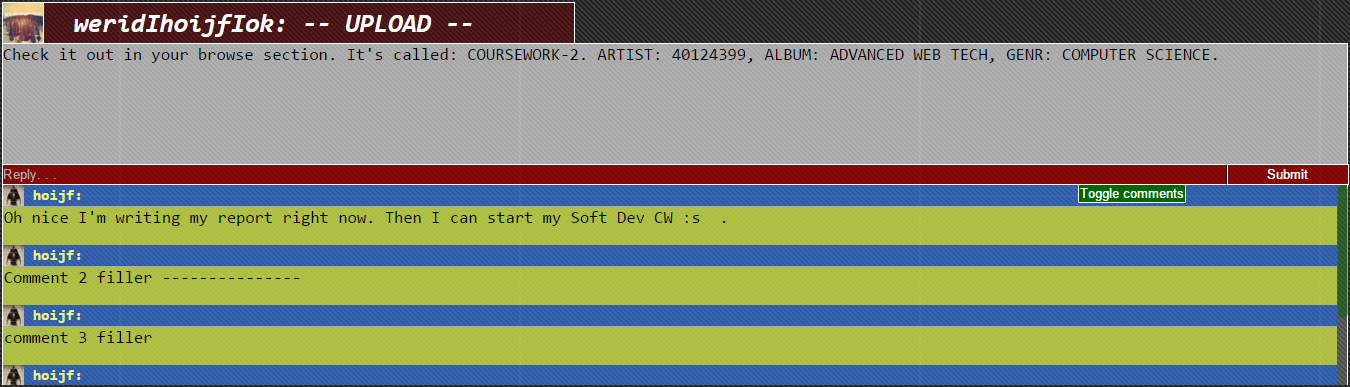


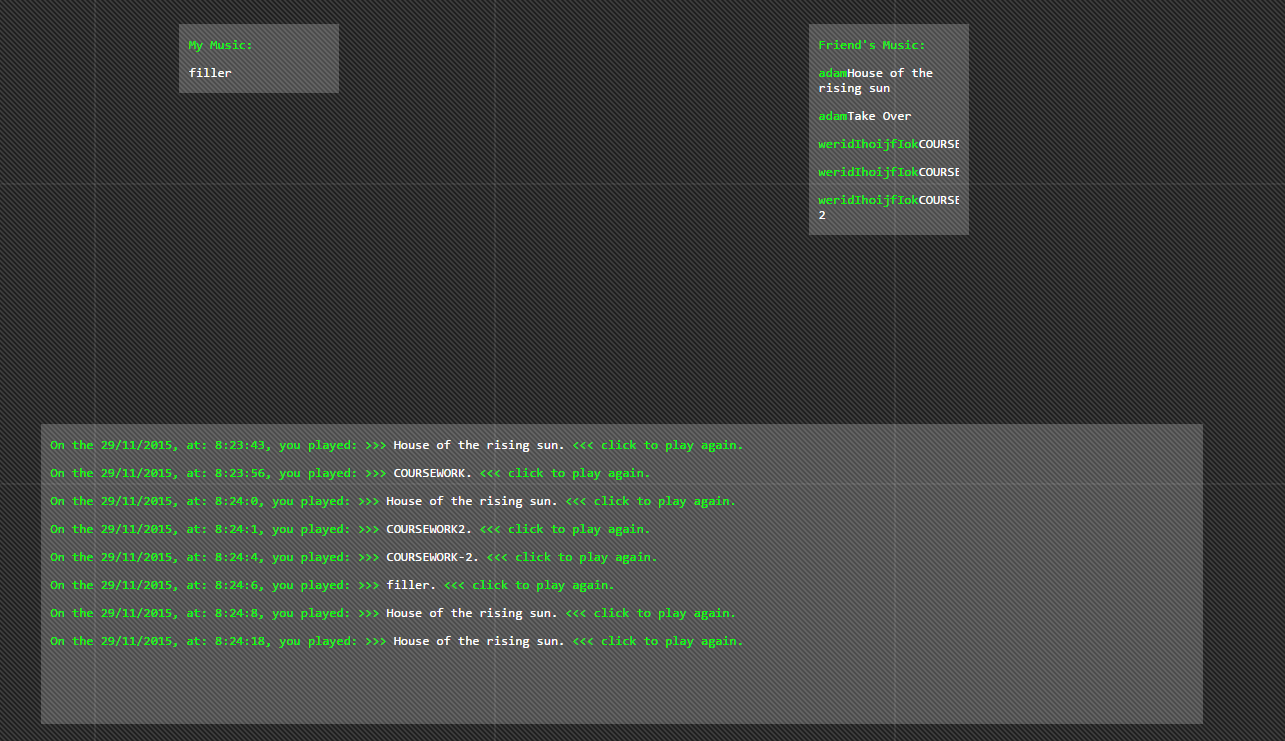


This is the resulting window. Here you can see the users in a nicely organized list, to avoid server overload there is a page system so you can fetch the next X results on the next page. Now to add a friend the process is as simple as clicking the user you want and he will be added to your friends list. This means he can now see your posts. For you to see his posts, he will need to add you to his friends list. Now lets upload a song to see how that works. Click the uploads tab in the header. Here the user is asked to fill out a form which includes information for title, artist, album, genre, visibility and to choose a song to upload. Once this is achieved the user can press the upload button and the file will be validated by the server. Currently file size is limited to 2mb. If the file name and file size pass validation the file will be uploaded to the server and a post will automatically be written to tell all friends what song has been uploaded.

For this page there are no redirects because I think users will often upload more than one song at a time. Once the user is done he can check his posts and the comments received. This is what one of the automatically generated posts will look like.

In the title there is the user picture, username and then a tag saying UPLOAD. The content of the post is simply the meta data and a suggestion to go listen to the song in the brows section. However this is a good moment to show the commenting system.

comments are shown one after the other from oldest to newest so anyone coming into the conversation later on can start from the top. There is a limit to the vertical size of the comments so the page doesn’t get filled up to quickly. If there are more comments than can fit in the given space then a scroll becomes available to look at the rest of the comments. This is a very efficient way of keep the page tidy while allowing a lot of content. Should the user want to hide the comments there is a toggle button which will hide or show the entire comments tab for this post depending on their current state.

 Once a user sees this post he may be interested in listening to this new song. To do so, all he has to do is visit the browse section of the web app. Here there are 2 lists, first “my music” then “friend’s music”. My music will be displayed in the correct section as will my friend’s music. The username who uploaded the song is also displayed in the list next to the track. The larger box at the bottom of the screen serves as a history or playlist system for the user. It keeps track of all the music he has listened to including the time. This way he can easily find songs he likes. To play a song the user must click the song name and this will play it. When this is done, an audio player will appear in the footer. If the operator wishes to skip ahead, turn the sound down, pause this is how he will achieve it. Should he want to play a different song all he has to do is click the name of this new song.

Enhancements, bug fixes and improvements:

For a better user experience I would like to change the search system to work for posts, songs and users. This would work by using a falldown menu which would pass a variable over to the next page this would tell the server which search is being executed and then allow it to render the appropriate service. Another addition to the search system would be to not show users which are already in my friends list.

It could be interesting to add a like system to songs and posts. Although this wbsite isn’t trying to be a facebook for music, it could be useful to have a song ranking and user ranking simply for users to be able to find good music more easily. Although in this scenario the friending system should be changed to a follower system in which to view a user’s posts you need to follow him not be in his friends list. This ties into another addition which would, if tied with this one, make the website far more interesting. When writing a post or uploading a song, a tag could tell whether the information is public, private or personal. If personal then only the user can see it. This could be for listening to your own music wherever you go for example. Private would only allow friends to see the information and public would be for followers and friends. To create something like this, I would need an additional column in my posts and songs segment to store the viewing rights of the data. This can then be compared the social connection between the 2 individuals and displayed or not displayed depending on the resulting connection. For the up vote system, each post and song would also need an additional column storing its total likes and dislikes. These would then be added or removed from the corresponding users total. This value could be reset on a monthly basis so the information is always relevant.

The music player needs improvements, if users could sort through data on the fly for example a search on the browse page in which you can search for album, title, up loader, likes, genre, new’ness’. Also the music player will overload the server if to many requests are sent in succession. A fix for this would be essential for improving user experience. To achieve this, each user could be allocated a limited amount of bandwidth to the server or the player could automatically block to many requests.

For the posts page, an interesting change would be to replace the current page system with an ‘infinite’ feed which the user can just scroll down for as long as he wants. The problem with this is loading only x amount of values at a time to not overload the server.

Some characters when inputting string will cause an error, characters such as “é”. These shouldn’t cause errors and a fix for this would be extremely useful. Although the problem can almost certainly be programmatically fixed, there is the possibility the python installed for levinux which is missing certain features simply cannot handle the characters although I find this highly implausible.

Initially the plan for this website would have also allowed users to share content anonymously using a peer to peer connection which would have been established using webRTC. This would have made it possible for users to host internet radios as well. This was abandoned due to time restraints.

Link to webRTC basics: <http://www.html5rocks.com/en/tutorials/webrtc/basics/>

Finally an important improvement for security purposes would be to check input string for any sql injection and remove the user from the database should he attempt this.

Critical Evaluation:

Although the website does generally run smoothly and all information is easily accessible. The sqlite running behind it isn’t very well organized. The friends list works by creating a new table for each user which holds the IDs of his friends. This means that there for each user there is a table containing his user info, and a table containing his friends. Both of these tables need to be checked every time someone is looking at the posts or song page.

Much like the sqlite, the python isn’t particularly well organized. I must say using vim doesn’t facilitate the use of functions but nonetheless, a lot of my functions will have some repeated code.

The music player implementation is less than perfect and really needs some revision. The crashing from clicking multiple songs in rapid succession is without a doubt the biggest issue with this website. Although it does work for intended use, any abuse will cause a server crash.

Bugs aside, the website offers good functionality, has correct us of redirects, sessions, dynamic urls, javascript, css, html/jinja2, uploads, loging in/out and creating account. Offers an involved experience in which the user has enough interaction to actually have some practical use from the product. Everything runs very quickly, and there are few notable bugs. Content is no longer being pulled from other hosts. All data is which is displayed is stored on the server, this avoids external hosts from changing pictures and songs to inappropriate content or any type of unwanted data. It also means their bandwidth isn’t hurt.

After using the website myself to listen to music while writing this report, I can say that it would be worth using as a free data storage and music player while on the go(mobile phone).

Resources:

<http://www.tutorialspoint.com/sqlite/sqlite_constraints.htm>

<http://www.tutorialspoint.com/sqlite/sqlite_using_autoincrement.htm>

Stack overflow:

<http://stackoverflow.com/questions/18817699/python-check-if-exists-in-sqlite3>

<http://stackoverflow.com/questions/13531149/check-for-a-cookie-with-python-flask>

<http://stackoverflow.com/questions/15586810/javascript-play-and-pause-a-song-with-one-click-on-the-same-image>

<http://stackoverflow.com/questions/195951/change-an-elements-class-with-javascript>

<http://stackoverflow.com/questions/27671882/how-to-play-a-selected-song-from-the-list-and-pause-the-earlier-song-that-was-pl>

<http://stackoverflow.com/questions/4071872/html5-video-force-abort-of-buffering/13302599#13302599>

w3school:

<http://www.w3schools.com/sql/sql_syntax.asp>

flask.pocoo

<http://flask.pocoo.org/docs/0.10/patterns/fileuploads/>

others:

<https://www.cs.tut.fi/~jkorpela/forms/file.html>

<https://developer.mozilla.org/en-US/Apps/Build/Audio_and_video_delivery/buffering_seeking_time_ranges>