16

**Report**

**For teaching set09103**

**Jacint Virag - 40270622**

Contents

[Introduction 2](#_Toc465260202)

[Home page 2](#_Toc465260203)

[Favourite’s page 2](#_Toc465260204)

[Genre page 2](#_Toc465260205)

[Show all page 3](#_Toc465260206)

[Search page 3](#_Toc465260207)

[Admin Panel 4](#_Toc465260208)

[Admin Add page 4](#_Toc465260209)

[Track page 5](#_Toc465260210)

[Design and architecture 5](#_Toc465260211)

[Folder structure 5](#_Toc465260212)

[URL structure 5](#_Toc465260213)

[Enhancement 6](#_Toc465260214)

[Database 6](#_Toc465260215)

[Front-end and functionalities 6](#_Toc465260216)

[Critical evaluation 6](#_Toc465260217)

[Personal evaluation 7](#_Toc465260218)

[Resources 7](#_Toc465260219)

[Frameworks 7](#_Toc465260220)

[Content 8](#_Toc465260221)

[References 8](#_Toc465260222)

# Introduction

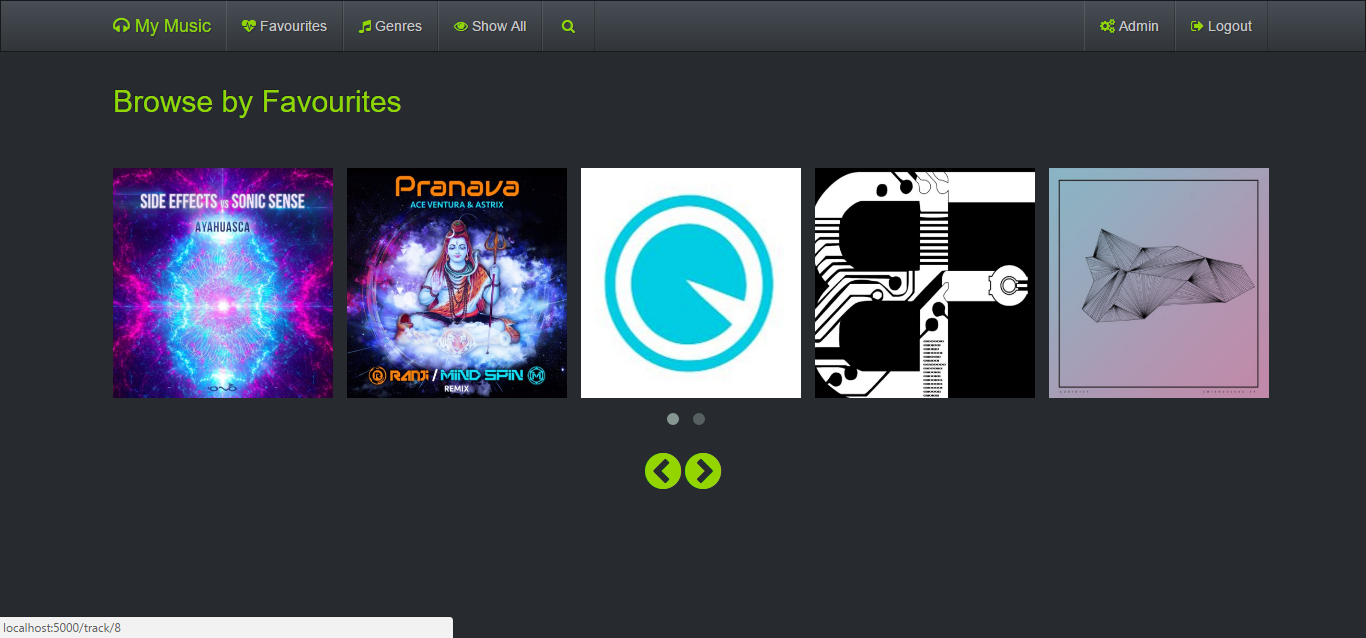
The web-app I developed is a music catalogue where the users can browse through and listen to mixes from different type of electronic music. The user can browse between genres, favourites or all mixes and also can search within the tracks by artist or track name. The user can log in as an admin and able to modify or delete music and its metadata or able to add new tracks into the database with file upload.

## Home page



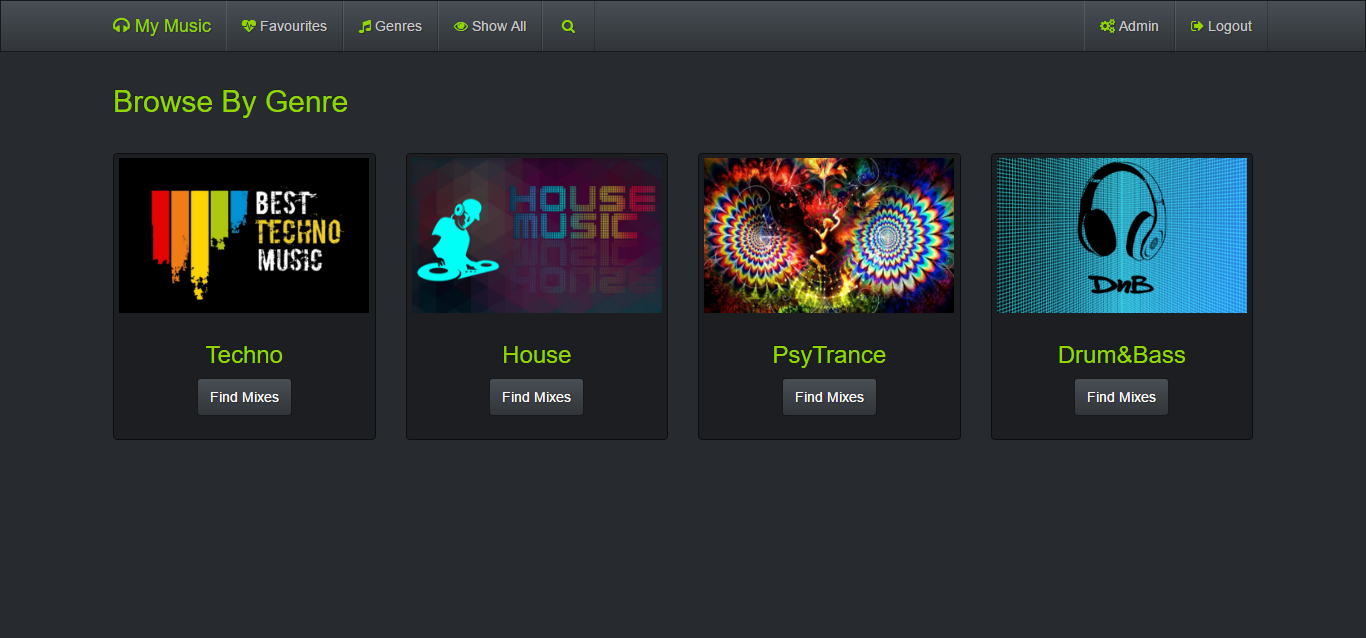
## Favourite’s page

I used OWL slider to display my favourite mixes. This feature brings some interactivity in. Every mix can be “favourite” and can be removed from favourites on the track page.



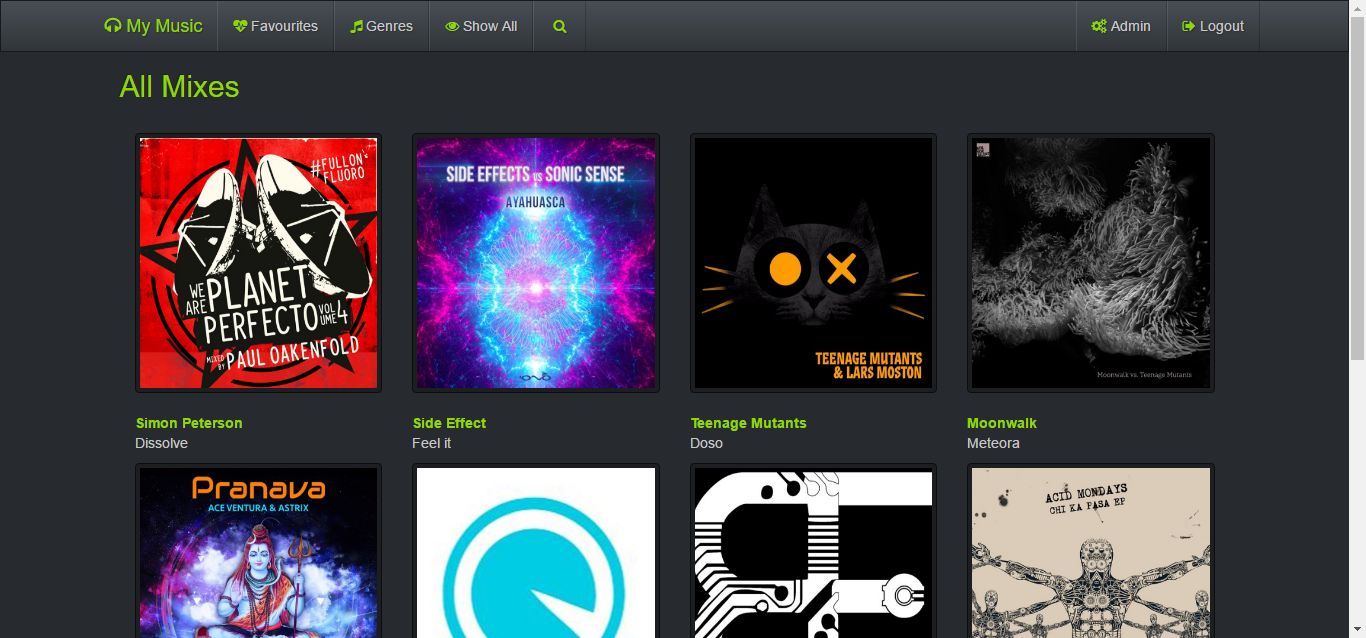
## Genre page

The genre page helps to select music by its type. At the moment I only used 4 types of genres but this could be extended and could be database driven.



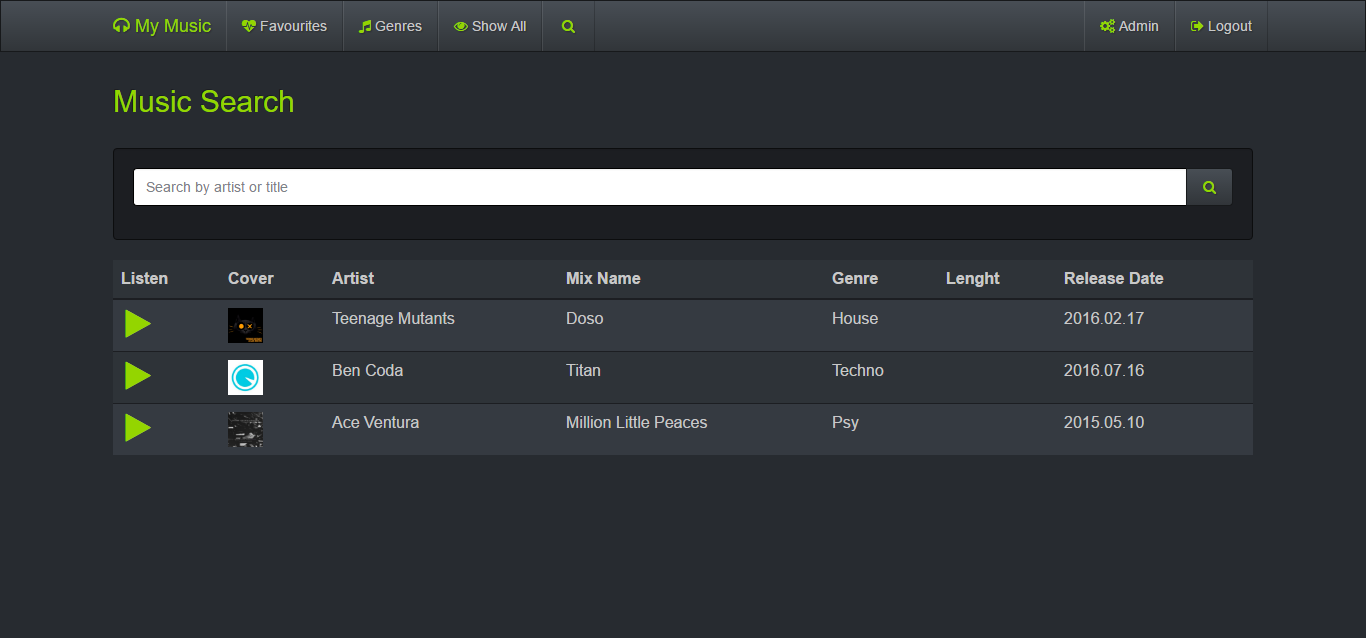
## Show all page

This page shows all the mixes from the database without any filtering.



## Search page

On this page the user can search for artist or tracks and can listen to the desired track.



## Admin Panel

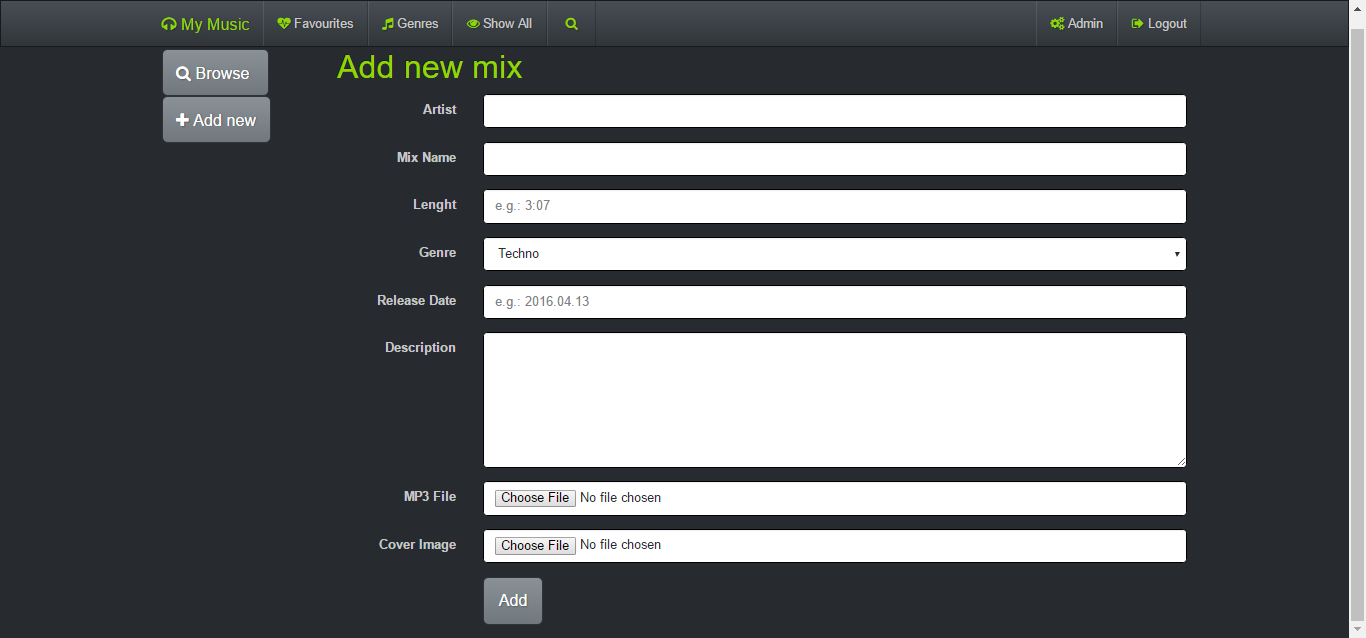
I created a table that shows a list of mixes. These can be edited or removed from the database. The cover picture and the mp3 files will also be removed from the static folder as well as the row from the database when the user deletes a track.

The username is “admin” , password is “admin”



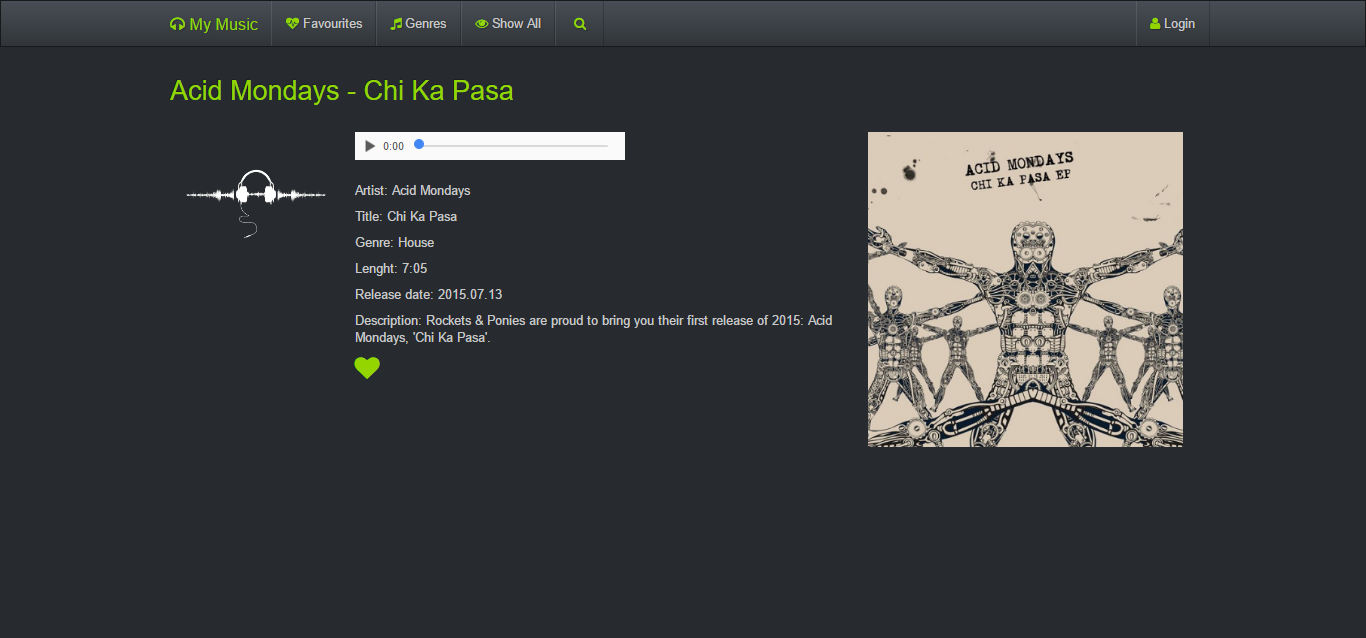
## Admin Add page

This is where the admin can upload new tracks to the database.



## Track page

The user can listen to the music by using a simple HTML5 audio player. By clicking on the heart the track be added or removed to favourites.



# Design and architecture

## Folder structure

I used SQLite to store information. The database file is “db/music.db”

The database, logging, secret key, admin credentials and basic setup are configured in: “etc/config.cfg”

All CSS files such as bootstrap, owl carousel, font awesome and custom theme are in “static/css”

All JS files are in “static/js”

All images are in “static/img”

All mp3 files are stored locally in “static/mp3”

The log file can be found in “var/logging.log”

All HTML templates are in the “templates/” folder

The app can be started by running the index.py file

The database schema can be created by running the db.py file

## URL structure

/route

/favourite

/catalogue

/genre/<type>

/show\_all

/search

/admin

/admin\_edit

/admin\_add

/login

/track

**Routes without html association:**

/logout

/delete

/upload

/update

/remove\_fav

/add\_fav

# Enhancement

## Database

* The database could be extended to more tables, such as artist or genres; these tables could be joined together.
* The artist table could provide information on my track page about the artist and display a picture. I could also create a separate route where the user could browse by artist.
* The genre table could drive the genre page and could be extended to more type of music.

## Front-end and functionalities

* Although I am using bootstrap as a responsive framework I could further optimise the mobile experience.
* Add recommended track on the track page.
* Change the basic HTML5 player to a more advanced JavaScript player like Dewplayer or Flowplayer.
* Add modal to view enlarger cover pictures.
* Add hover effect on thumbnails, like a play icon with transparent background.
* Add form validation for the Admin\_add page, datepicker to avoid different date formats, File extension checker for images (png or jpg only) and mp3 for audio.
* Add error messages or success messages on file upload
* Check image and mp3 size on upload; also add id number to the files for better organisation.
* Add cover image and audio file change to Admin\_edit page so the user can change more than just a metadata
* Add pagination to Show\_all page or Admin page
* Add more dynamic calls like Ajax to avoid page refresh. For example on the track page if the user adds a mix to the favourites the music stops playing.
* Deploy the app

## 

# Critical evaluation

I believe I managed to cover most aspects what we learnt from the workbook. I am happy with the final website as it has all the features that was required be the assessment but obviously I know that this is just a good base to start from. As I mentioned above there are many things that could improve the user experience. For me personally the secure file upload and a form validation are the most important things that I would like to extend and have a better understanding as these things always come up during development.

My code is not very organised and could be a bit more logical, for example the database could be called in one function instead of calling it and closing it on every page. I know there is a feature like that in the workbook but by some reason it didn’t with my queries.

# Personal evaluation

The reason I choose this module to learn something new. This was definitely fulfilled as I haven’t used Python and I haven’t even heard of Flask or Jinja2 before.

The biggest challenge at the begging was the Linux environment and using Vim as a code editor. This kind of slowed me down at first but now I am confortable with the command line and the Vim as an editor.

The workbook provided an easy to follow guideline to learn Python and since I coded before in PHP just had to get used to the syntax and the templating but the logic was kind of the same behind. Although I still miss calling PHP wherever I want in the page.

SQLite was very similar to MySQL but it wouldn’t be my choice for my next project, I found that it is limited in many ways and I also would like to try a NoSQL database system. The day before the hand in I noticed a bug which was related to SQLite. After few hours of research I found the answer on StackOverflow. This website also helped me to crack other problems regarding python, bootstrap and css.

I am happy that we use GitHub for our coursework instead of a simple upload to Moodle. GitHub is very popular nowadays within companies so this will at least give me the basic knowledge of how to use it. Although I had to admit I should commit more often.

Another confession I need to make is that I completely ignored CSS in this project; I mainly used inline styling which today is almost obsolete. I just didn’t find it important for the projects sake. Looking through my code it would have been so much simpler to create a few classes and use them thoroughly.

# Resources

## Frameworks

**Bootstrap**

Type: Front end framework

URL: <http://getbootstrap.com/>

License: MIT

**OWL Carousel**

Type: Slider

URL: <http://owlgraphic.com/owlcarousel/>

License: MIT

**Font Awesome**

Type: Font/Special charachters

URL: <http://fontawesome.io/>

License: MIT

**DB Browser for SQLite**

Type: Software

URL: <http://sqlitebrowser.org/>

License: GNU

**DB Browser for SQLite**

**Bootwatch**

Type: Stylesheet

URL: <https://bootswatch.com/>

License: MTI

## Content

All music and cover images were downloaded from Sound Cloud. All of them are under the MIT or Creative common license.

## References

Bootstrap documentation: [http://getbootstrap.com/](http://getbootstrap.com/%20)

Python Flask handbook: <https://www3.ntu.edu.sg/home/ehchua/programming/webprogramming/Python3_Flask.html>

Jinja2 documentation: [http://flask.pocoo.org/](http://flask.pocoo.org/%20)

SQLite3[: http://flask.readthedocs.org/en/0.9/patterns/sqlite3/](:%20http:/flask.readthedocs.org/en/0.9/patterns/sqlite3/%20)

Various examples: <http://stackoverflow.com/>