**Napier University**

16

**My Music Collection**

**Python Application Report**

**Katarzyna Gniady-Brzoza**

Contents

[1. Overview 2](#_Toc465260017)

[2. Software Design 3](#_Toc465260018)

[2.1 Images 3](#_Toc465260019)

[2.2 Application Development 3](#_Toc465260020)

[3. Application Evaluation 4](#_Toc465260021)

[3.1 How the application can be improved 4](#_Toc465260022)

[4. Application Screenshots 5](#_Toc465260023)

[5. References and Resources 11](#_Toc465260024)

# Overview

I have been asked to create an online application, which should be designed with Python and run within Levinux Environment. It should be available to access thought putty as well. The application should use python programing but can include other languages such as HTML, JavaScript or CSS as well.

I decided to choose the application which targets the group age 18 - 50, and which allows the user to interact with it in a very easy way. This application was designed for learning purpose so it is not perfect yet. It will be improved in the nearest future as soon as I feel more comfortable with Python (Flask) programming. It can be viewed and downloaded via my GitHub account via following address: <https://github.com/katgni/final>

“My Music Collection” is an application which allows users to see my music collection grouped by categories. There are a few pages that can be accessed thought browsing. The main page contains general information about the application. The Album, Artist and Genre pages are grouped by categories. There are also other pages such as login page and another page that allows user to upload records to database (mydatabase.db). Login page can be accessed via “My Account” link that is placed on the top navigation bar. There is also “Database” page which initially/previously was connected to SQLite but I have changed it to JSON due to some issues that appeared while testing connections and queries. I found JSON much more friendly and easier to use than SQLite.

The main purpose of the application is to show my programming skills and demonstrate my favourite music albums. I'd like the user to be able to do activities while using this application.

User should be able to perform task such as:

* Access the application via internet browser at <http://localhost:5000/>
* Read the overall description about the application
* Check general information about albums grouped in categories
* See appropriate pictures on each page
* Gather the most important information about the music I like
* Find information about each of albums from my music collection
* Login (I created 3 testing accounts: “kat”, “mat” and “simon”. Password for each of them is “napier”) The application should be able to handle the users input such as username and password.
* Add Records to database

Across the website there will be some design implications for users. For example:

* Easy navigation – users will typically interact with the system keyboard and mouse. Navigation and selection will be on the top. The navigation and will be consistent across all system however its content will change in “My Account” section and there also be a page that will be visible only for logged users. It is called “Music” and the user can add records by filling in the HTML form. The navigational system will be designed to be intuitive to users.
* Friendly fonts – fonts used on the site will be appropriate, readable and attractive.
* Clear and easy to understand content.

The interaction with the application has been kept as simple as possible because if it was more complicated or confusing, the users would lose the interests in using the application. All pages keep the consistent layout and colour scheme, both suitable for the audience.

To get all necessary information and resources which were required or helpful to complete the coursework I have used module workbook and notes, which are available on the module website. I have also used other resources that I found online. I found Stack Overflow website and documentation provided on Python Flask official website most helpful during designing the application.

Initially Levinux and “Putty” seemed to be quite complicated, annoying and confusing for me but with time I even started enjoying working with them.

# Software Design

## Images

All the images except that one which is placed in jumbotron were designed by myself. The picture that was downloaded from the internet is referenced in the bottom of the report in resources section [1].

## Application Development

I found this stage of the project most challenging because I didn't have any experience in working with Python ever before. In order to understand how it works I watched many tutorials on YouTube, one on Lynda.com and have spent plenty of time on researching the other sources. I also participated in tutorials available on the module website.

I started building the application from creating the new project and installing the environment in a place that I wanted the application to run on. Then I chose the templates of Bootstrap that were provided by Simon Wells on the course website. It thought it would help me to make the working application much quicker and easier. I followed all steps described in the workbook and successfully customised my application.

The main problem that appeared during the development process was the “broken pipe” error, which forced me to run the application again. I also had some problems with uploading my application to GitHub. Once it was uploaded, I wasn’t able to make any changes through command line on my PC. I left the first uploaded version in “coursework” repository on my GitHub account but also created a new repository called “final” (available at: <https://github.com/katgni/final> )

# Application Evaluation

## How the application can be improved

If I had to do similar application in the future, I would add more functionality and would make the website more interactive.

Things looked very complicated in the beginning of the project. The problem was that I had never worked with similar projects before and didn’t have any experience in this area. I was a little bit confused but when the project is finished I can say, I have learnt a lot while working on it.

I’m quite happy with the final product but there are some factors I would like to improve in the future. It could be improved by adding more items to the database and to the HTML content.

I would also improve the functionality for users by giving them more options to choose. If I had more time, I would spend much time on creating and implementing Python scripts that allow users to edit/delete their details form the database.

I would add the registration form. All their details would be stored in the database and displayed on users screens each time when they’re logged into their accounts. Another option for users would be the ability to write reviews. Reviews are essential parts of most of the professional websites and I would include this option to my website too.

From my personal point of view the whole project could be better managed and if I could do it again:

• I would create a new plan and follow the plan step by step.

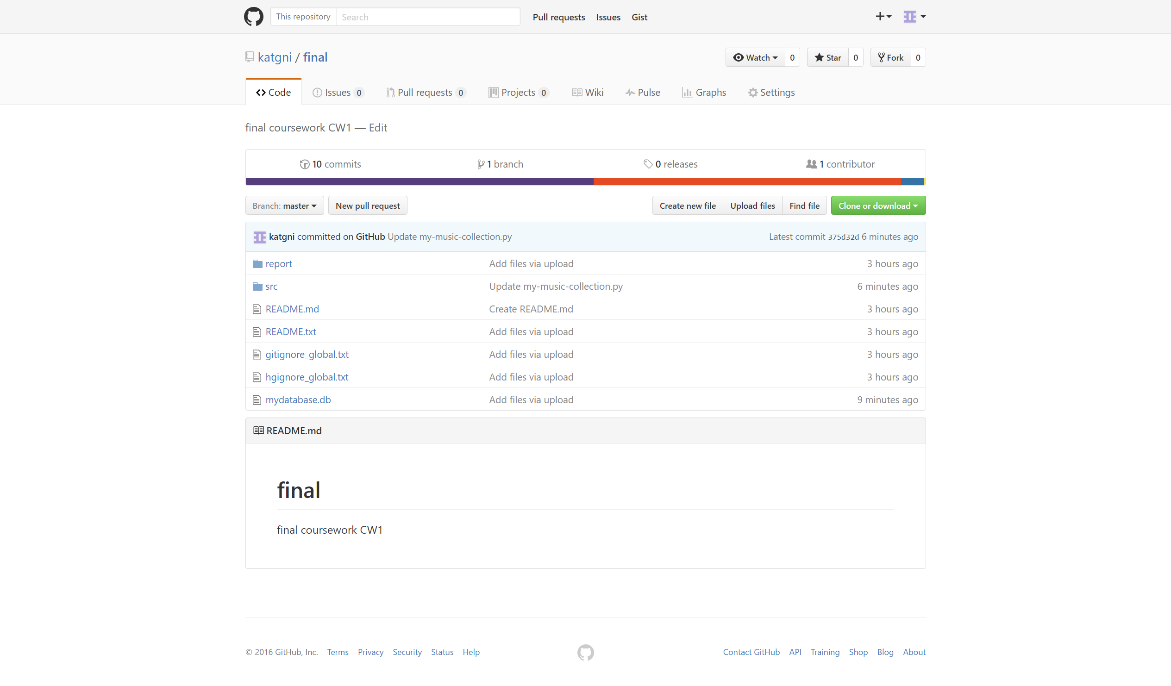
• I would manage better the time, which I spent on the project.

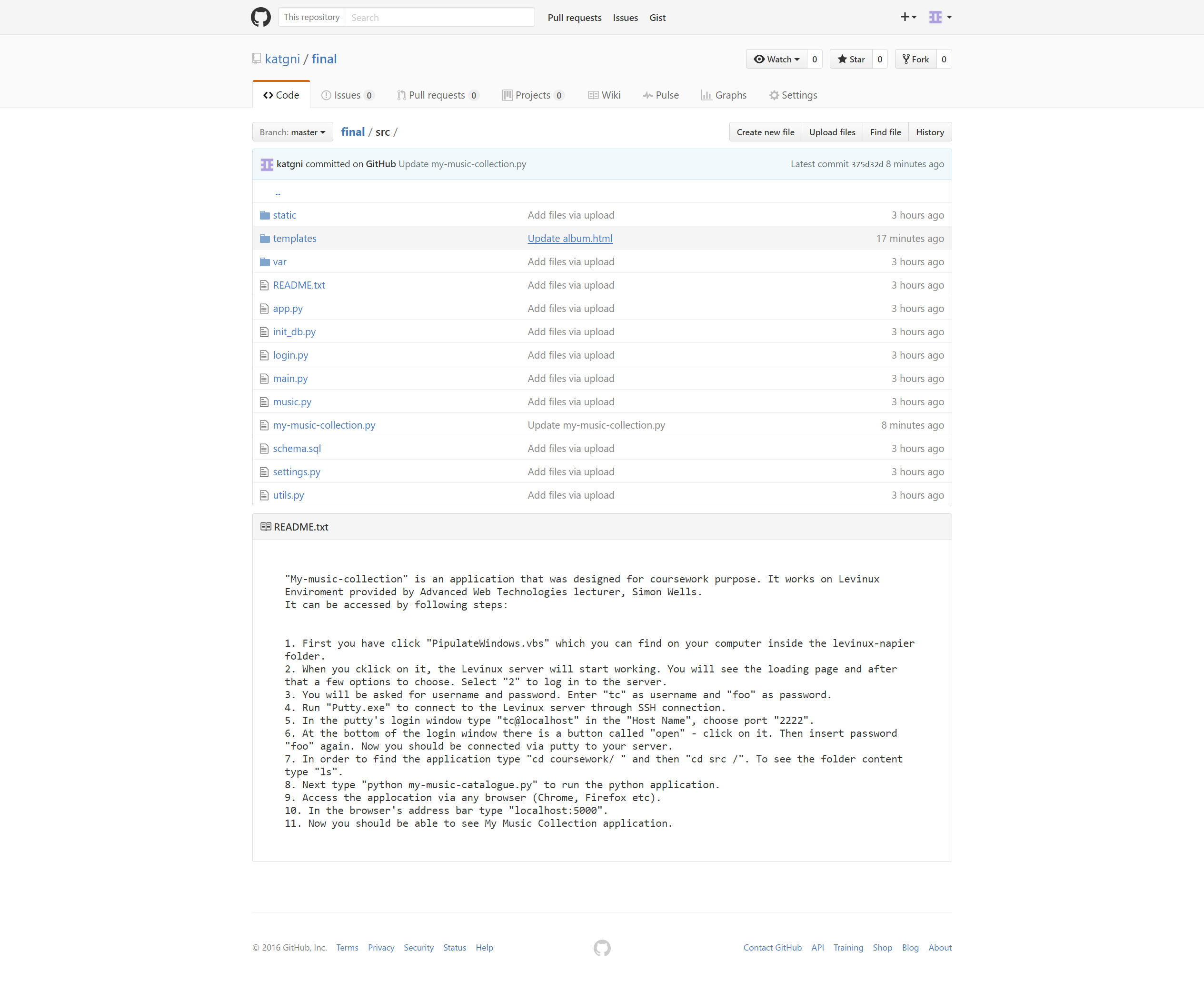
There were some advantages of the project. I had access to all required resources for project delivery such as Levinux, Python and Putty, VIM, and Microsoft Word.

The weak point of the project my project was the paperwork. It took a lot of time until I successfully managed to complete my report. Lack of time didn’t allow me to spend enough of time on research. While designing websites in the future, I will spend much more time on researching in order collect more information about this kind of websites and the functionality that needs to be implemented.

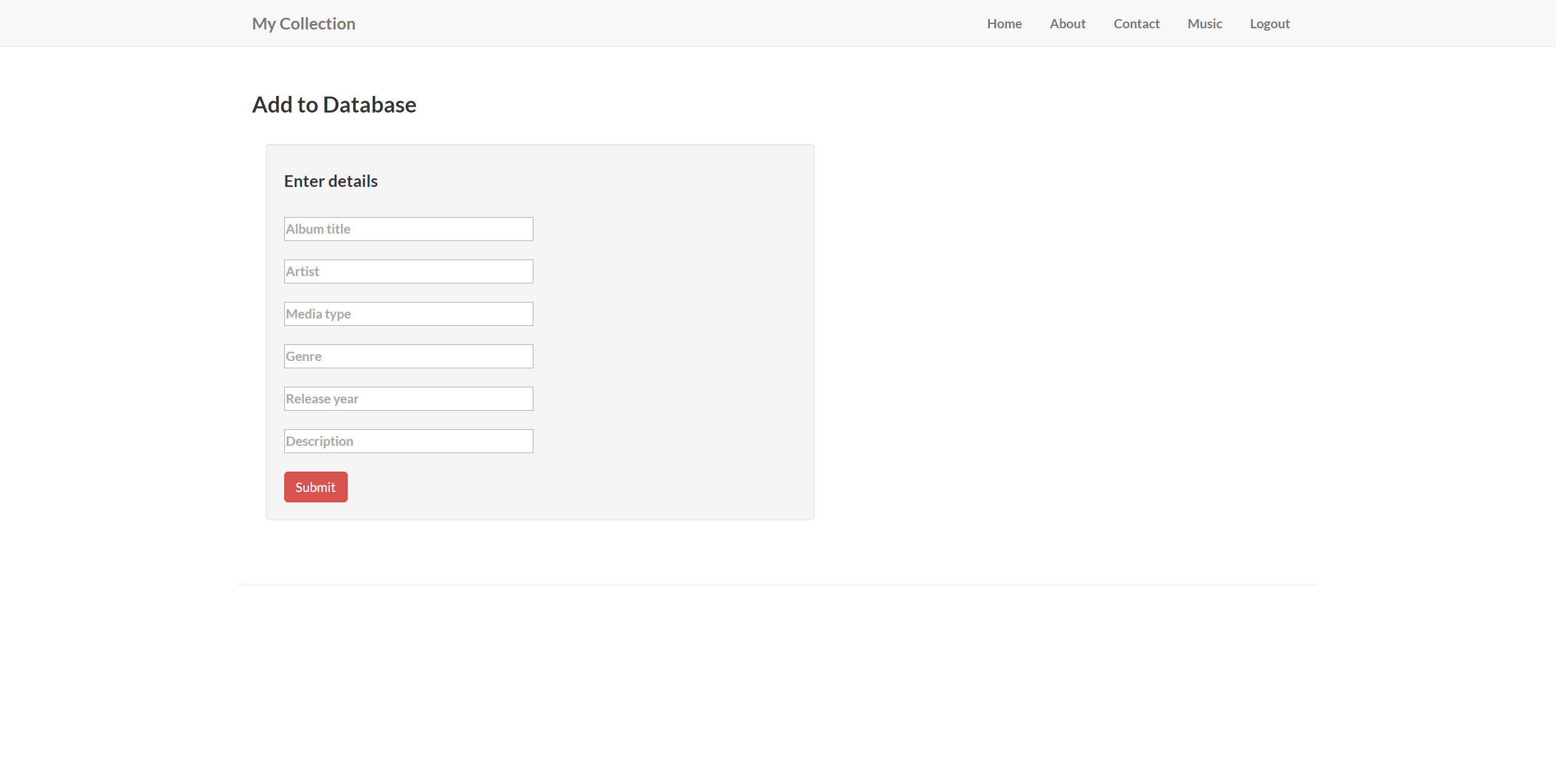
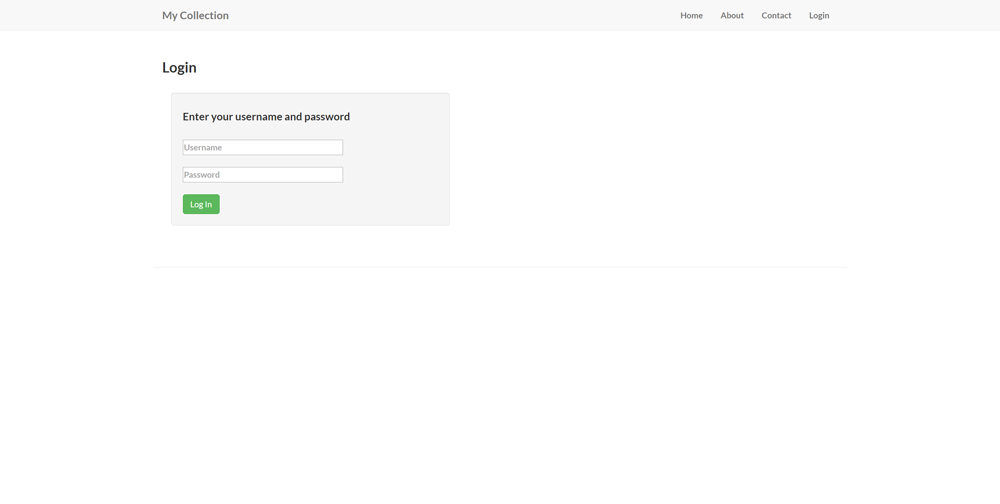
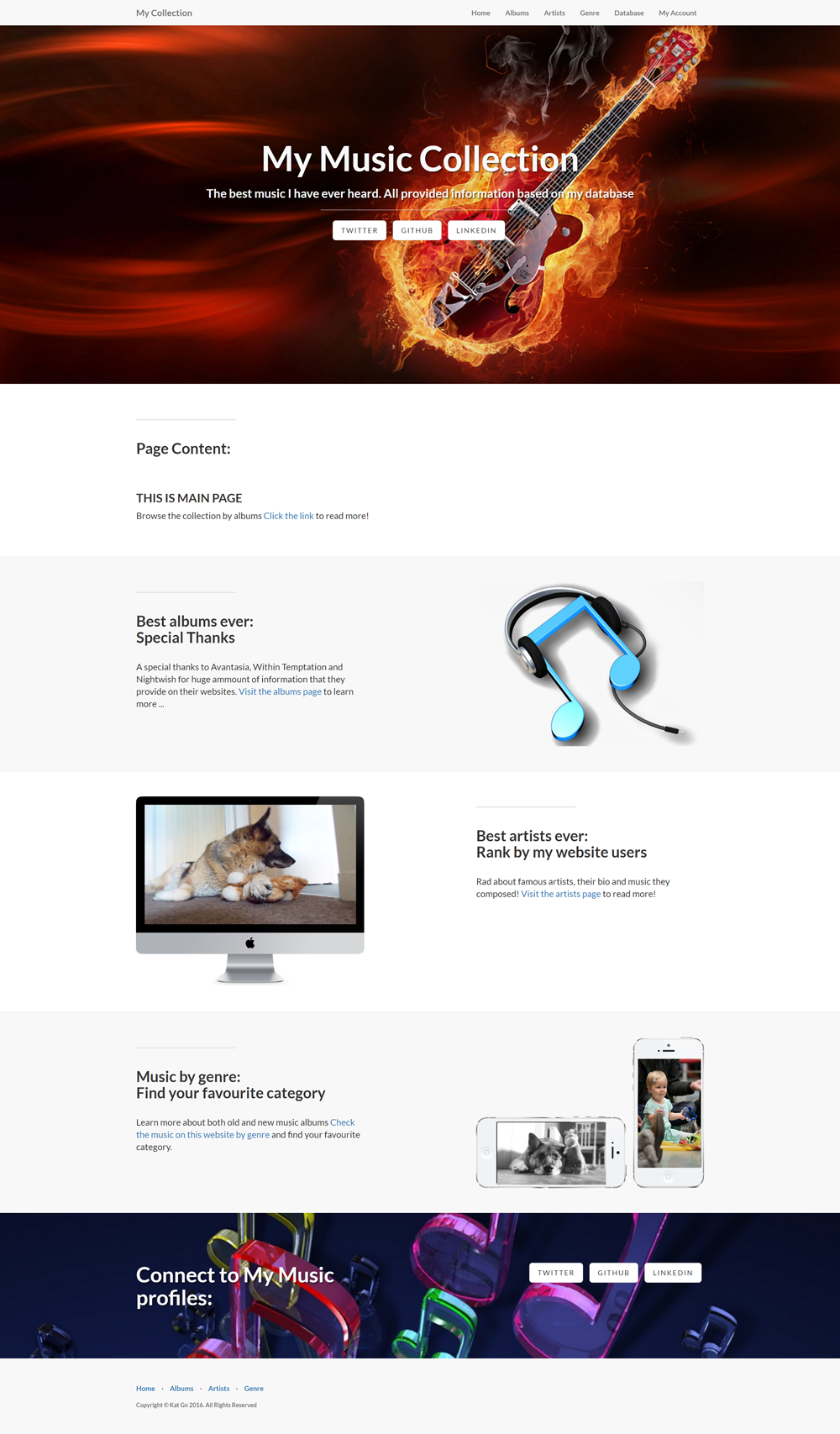
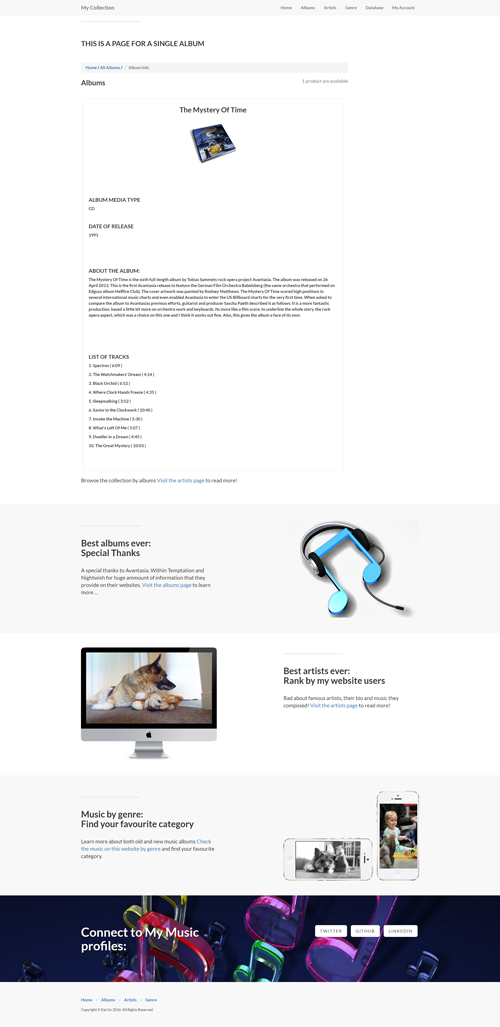
# Application Screenshots

#### Github





#### Pages



|  |  |
| --- | --- |
|  | C:\Users\Kat\AppData\Local\Microsoft\Windows\INetCacheContent.Word\albums-page.png |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# References and Resources

1. ***Flaming Guitar Wallpaper, Accessed 15 October 2016***

*https://s-media-cache-ak0.pinimg.com/originals/a3/3b/05/a33b052156de55435826cfd918a55a56.jpg*

1. ***Course source code, Accessed 01 October 2016***

*https://github.com/siwells/teaching\_set09103*

1. ***Advanced Technologies Course website, Accessed 25 September 2016***

*http://siwells.github.io/teaching\_set09103/*

1. ***SQLite3 Tutorial, Accessed 10 October 2016***

*https://www.tutorialspoint.com/flask/flask\_sqlite.htm*

1. ***Flask Web Development with Python Tutorial - 7 - Passing Objects into Templates, Accessed 05 October 2016***

*https://www.youtube.com/watch?v=AOboS0RESt4&feature=youtu.be*

1. ***Creating a Web App From Scratch Using Python Flask and MySQL: Part 6, Accessed 28 September 2016***

*https://code.tutsplus.com/tutorials/creating-a-web-app-from-scratch-using-python-flask-and-mysql-part-6--cms-23402*

1. ***Working with JSON in Python Flask, Accessed 25 September 2016***

*http://codehandbook.org/working-with-json-in-python-flask/*

1. ***Learning Flask tutorial on Lynda.com, accessed 20 September 2016***

*http://codehandbook.org/working-with-json-in-python-flask/*