Coursework Report

Danni Xu
40332963@napier.ac.uk
Edinburgh Napier University - Advanced Web Technologies (SET09103)

Abstract

With the rapid development of Internet, the various of website has become the main method for people to obtain information conveniently and quickly. Therefore, the establishment of the website is significant in today's society, people are devoted themselves to building popular websites with powerful function and innovation. The purpose of this report is to introduce a prototype music web application which was created by using python flask. It also used HTML, CSS, JavaScript, bootstrap, etc. Next, basic introduction, design architected, the functions and envisioned implementation of this application will be described. With the rapid development of Internet, the various of website has become the main method for people to obtain information conveniently and quickly. Therefore, the establishment of the website is significant in today's society, people are devoted themselves to building popular websites with powerful function and innovation. The purpose of this report is to introduce a prototype music web application which was created by using python flask. It also used HTML, CSS, JavaScript, bootstrap, etc. Next, basic introduction, design architected, the functions and envisioned implementation of this application will be described.

Keywords – Python, Flask, Route, Music ,Navigation, Templates, Static Files, HTML, CSS

1 Introduction

The theme of this web application is music, Music Space. The core element of Music Space is the best music library, the regular music classification and the social function. It focus on discovery and sharing to provide online music services, according to the user's browsing records to predict user preferences, and recommend the song list which they may love, so as to create a new and convenient music life for users.The theme of this web application is music, Music Space. The core element of Music Space is the best music library, the regular music classification and the social function. It focus on discovery and sharing to provide online music services, according to the user's browsing records to predict user preferences, and recommend the song list which they may love, so as to create a new and convenient music life for users.

2 Design

2.1 Layout Analysis:

Home page: includes 'Navigation bar, Carousel map, body content' three parts.
Navigation part: includes Find Music bottom (homepage), My Music, Login in. When you narrow the page, a button appears on the right, forming a dropdown navigation, and it can connect to the three pages above. Besides, there is a search bar on the right side of the navigation bar.
Carousel map: it will display the latest music information, including the latest release of the song, MV, or music related activities.
Body content: includes 'Billboard, Singer, Classification, FM, MV, HOT! ' six parts.
Billboard page: The latest ranking list of songs will be shown.
Singer page: On the left is the singer classification, the right shows singers, click the singer name, you can get the details about the singer.
Classification page: The left is a variety of categories, such as language classification, genres classification, theme classification, etc. The right is song lists which is according to the classification list.
The child pages in the body content module can click on the Find music in the navigation bar, or the 'back' button or 'Music space' of the footer can be returned to the home page.
My music: includes personal accounts, personal collections, social information and so on.
Login in page: the registration section on the left and the login part on the left.

2.2 Technical Analysis:

The main technology used in the web-app: python-flask, HTML, CSS, JavaScript, Bootstrap.

Using app.route to add more routes to the web-app, and write functions then add decorator for each one to make the function into routs, when a web-flask is running, that app will be accessible in a web browser on a particular network.

From the homepage to the singer page and then to the 'Taylor Swift' page, it shows the URL hierarchy clearly.

Using HTML templates with Conditional Arguments, redirect, request, responses achieved the function that in the landing page under the address bar, you can enter any words and it can be transferred to the welcome page, and display the specific name just input after 'welcome'. If the input is 'none', it will be on the login in page.

Next, the source code of the implementation will be displayed.

Listing 1: Example Code

```
from flask import Flask, render_template, redirect, url_for, request
from datetime import datetime
app = Flask(__name__)

@app.route('/loginin/<username>', methods = ['GET', 'POST'])
def welcome(username):
    user = {'name':username}
    if username == 'none':
        return redirect(url_for('loginin'))
    else:
        return render_template('welcome.html', user=user)
if __name__ == ''__main__'':
    app.run(host='0.0.0.0', debug=True)
```

Using template inheritance allows each page to have the same navigation bar and footer.
Next, take the singer page as an example.

Listing 2: Example Code - HTML file

```
{% extends ''inheritance/base.html" %}

{% block content %}
<div class=''page-header" style=''padding: 100px 0 0 0 ; color:red">
  <font size=''12", align=''center"><h1>Not Found !!!</h1></font>
</div>

{% endblock %}
```

Listing 3: Example Code - Flask file

```
from flask import Flask, render_template, redirect, url_for, request
from datetime import datetime
app = Flask(__name__)

@app.route('/singer/')
def inherits_three():
    return render_template('singer.html')

if __name__ == ''__main__'':
    app.run(host='0.0.0.0', debug=True)
```

It also provides own error page, If you enter a path that doesn't exist, it will jump to the error page.
Next, the source code of the implementation will be displayed.

Listing 4: Example Code

```
from flask import Flask, render_template, redirect, url_for, request
from datetime import datetime
app = Flask(__name__)

@app.errorhandler(404)
def not_found(error):
    return render_template('404.html'), 404

if __name__ == ''__main__'':
    app.run(host='0.0.0.0', debug=True)
```

Using redirect, request, responses and some HTML element achieved that jump to the home page by entering the name in the form of the login in page. If you do not complete it, you can not jump it. If you put nothing in the form, and submit, the bottom of the form will show the word 'Try Again, please!'.
Next, the source code of the implementation will be displayed.

Listing 5: Example Code

```
from flask import Flask, render_template, redirect, url_for, request
from datetime import datetime
app = Flask(__name__)

@app.route('/loginin', methods=['GET', 'POST'])
def loginin():
    error = ''
    try :
        if request.method == ''POST'':
            attempted_username = request.form['username']
            attempted_username = request.form['password']
            if attempted_username == ''admin'' and attempted_password == ''password'':
                return redirect(url_for('homepage.html'))
            else:
                error=''Try Again, please !''
        return render_template(''loginin.html'', error = error)
    except Exception as e:
        return render_template(''loginin.html'', error = error)

if __name__ == ''__main__'':
    app.run(host='0.0.0.0', debug=True)
```

2.3 Style analysis:

The site is minimalist design style, with black and white colour, not pursuit of exaggeration, but simple and practical. So that users can quickly understand the website architecture, quickly find what they need, give users a good experience.

3 Enhancement:

It's just a prototype, finished the basic page. Firstly, what I want to add most is music play function, though I try to add music, it still doesn't work. Besides, video play function also required.
Secondly, I try to add the function of showing the current time and date, hoping to give the user a better experience, but this function has not yet been implemented. Thirdly, it needs to add the page for songs play and add the comment function below the songs, through the other people's comments, experience the story of the song, share their feelings and find resonance in the comments.
Thirdly, for 'FM' page, It can be divided into two parts, one is personal FM, which can be recommended according to the user's preferences, the other is some programs produced by other users and you can upload your own programs. For example, it can include Music stories, talk shows, emotional topics and other modules, so that everyone can easily express themselves and communicate with others.
What is more, for 'MV' page, it can be similar to the singer page, on the left is the classification, and on the right is the video. Besides, under the video, it can add some related videos and discussion area. As for 'HOT!' module, it can include the most popular songs, singers, MV, FM. Put in some of the hottest and most popular things in current. Finally, it can connect database to improve and make it

more similar to a real site.

4 Critical:

4.1 Advantages:

Using template inheritance to make each page have the same navigation part, so that the site is unified and coherent. In addition, the web pages are static and dynamic, not rigid and full of vitality. Besides, the most basic functions of Python flask are realized.

4.2 Disadvantages:

Firstly, the pages are simple and crude, due to not having a good grasp of the knowledge of web page decorated. Secondly, there are too few functions to implement because of the lack of relevant knowledge at present. Thirdly, due to the limited time, only a part of the page is displayed, and other envisioned pages have not yet been implemented. In addition, there is no website security aspect involved.

5 Personal evaluations:

Through this course and the coursework, I learned how to use Python flask to build a website, although it is still rudimentary. Then, I have a better understanding of the HTTP protocol, API and some other the basic concepts via practice. I met a lot of challenges in the process of doing this coursework. Firstly, it is the first time for me to touch python, I did not even know the most basic commands at the beginning. Secondly, because my knowledge of HTML, CSS, JavaScript is not strong enough, I spent a lot of time on it. Thirdly, because of the language problem, I could not understand the introduction of workbook very well. Besides, in technical aspects, when I built a website, there were always displaying some errors or the program cannot run.

Although there were many difficulties, I looked for various methods to solve them. For example, I found some of the relevant video to supplement, when there was an error in the page, I searched the error shown on the page in the web to find answers or ask my friends for advice. What is more, I found that YouTube is a good learning platform.

In the end, it is a fantastic and challenging experience for me. Although there are a lot of shortcomings, I have tried my best to do it well. What is more, it stimulates my interest in learning python via this coursework. I hope that through the next study, I can build a site with more beautiful appearance and excellent functions.

A Appendix

-app screenshots

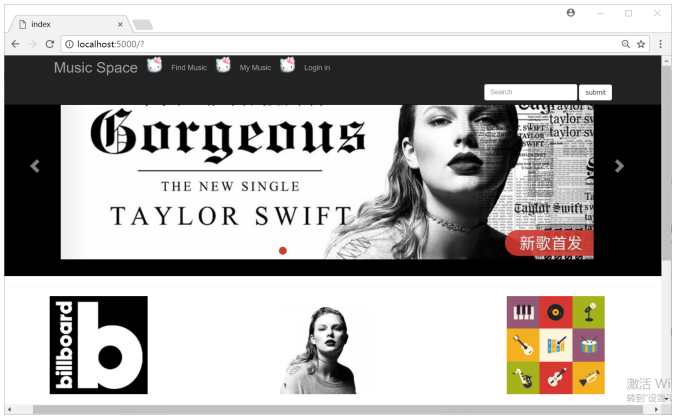

Figure 1: **Homepage** - The lower part of the page

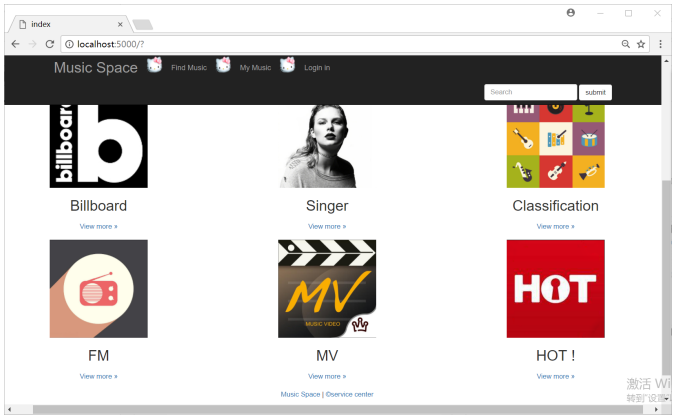

Figure 2: **Homepage** - The top half of the page

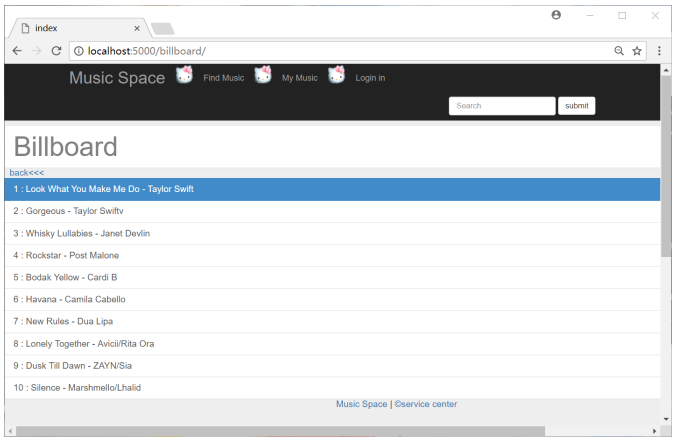

Figure 3: **Billboard page** - Ranking list

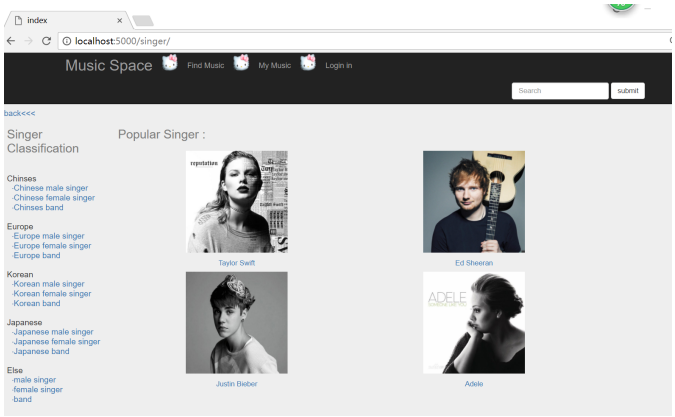

Figure 4: **Singer page** - Singer classification

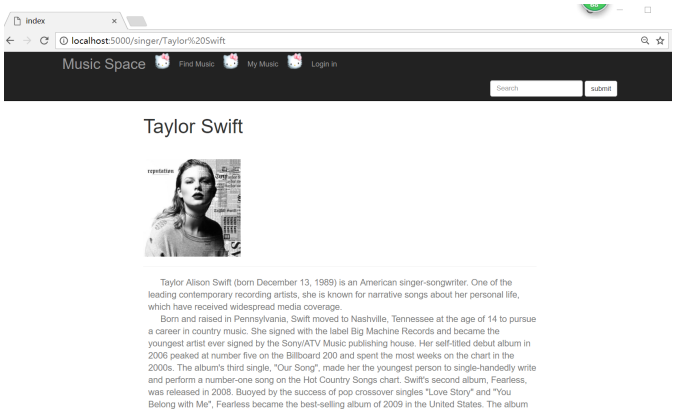

Figure 5: **Singer-Taylor** - A sub page of the singer page

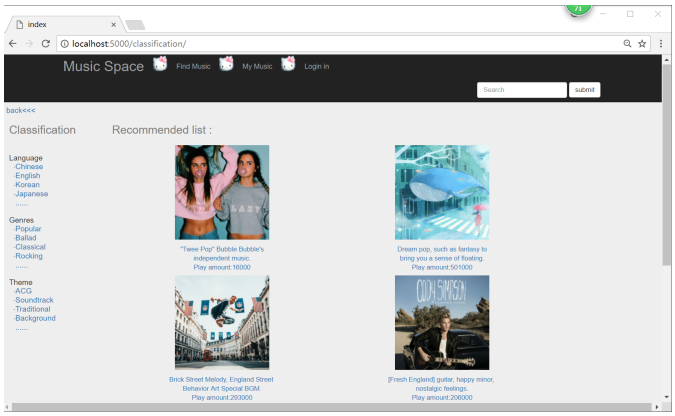

Figure 6: **Song classification** - Various forms of classification

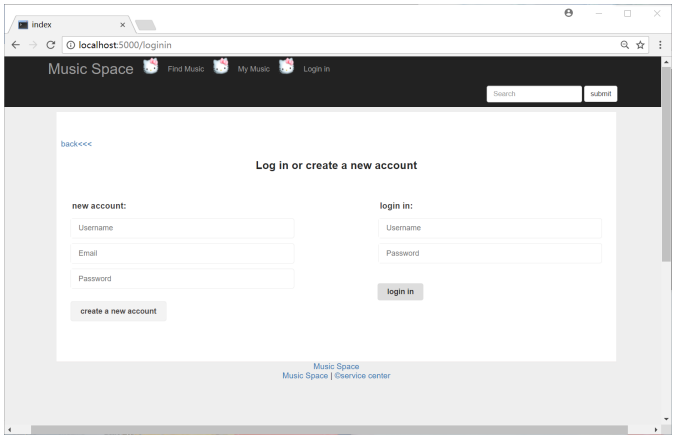

Figure 7: **Login in page**

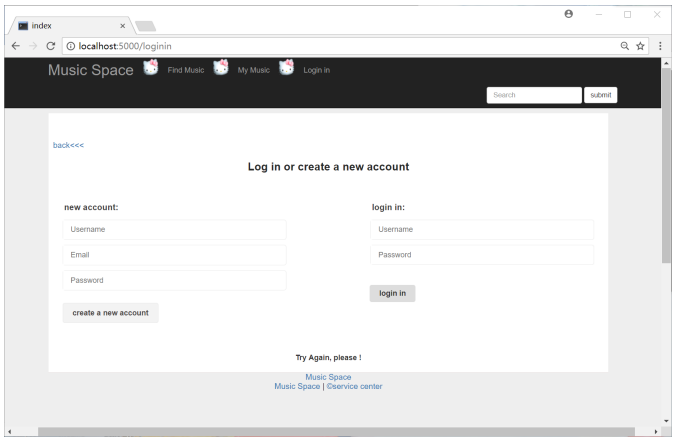

Figure 8: **Remind**

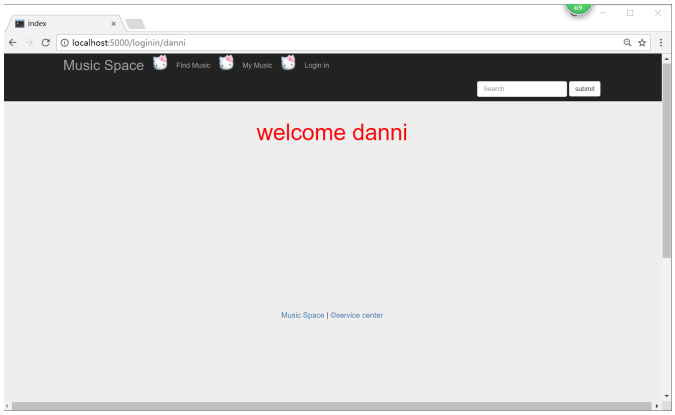

Figure 9: **Welcome page** - Enter the name in the address bar to show welcome

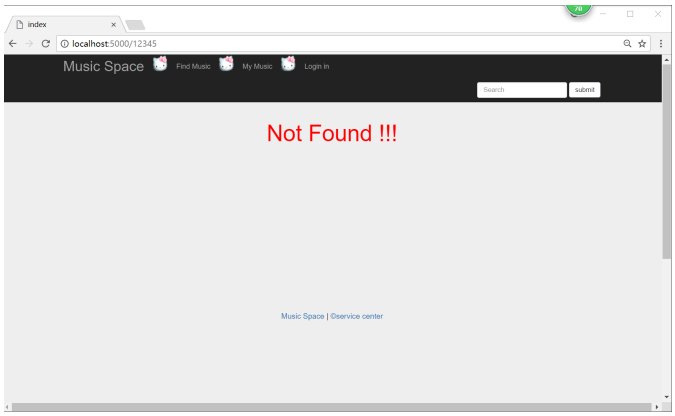

Figure 10: **Error page**

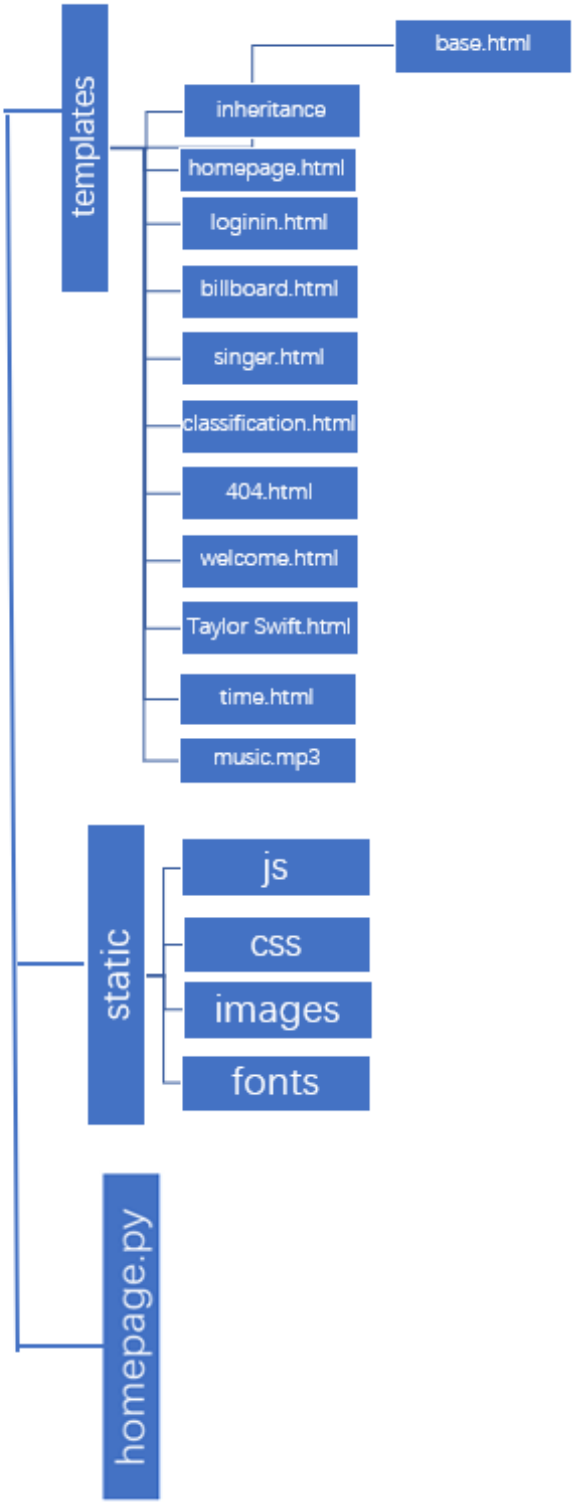

Figure 11: **File structure** - clear display of file structure