Coursework Report

Boyang Zou
40333231@live.napier.ac.uk
Edinburgh Napier University - Advanced Web Technologies (SET09103)

Abstract

The second coursework is also a web app based on python, flsak and a series of tools and package which add some new functions based on the first coursework. Beside the use of routing (and URL hierarchy design), static files, request, redirects, responses, template and template inheritance which exerted in first coursework. Second coursework contains the use of data storage, database, session, flash, config. There add a message board, user can leave there message on this web app. However, it will only display after user log in. And if user log out, the message board will disappear.

Keywords – Boyang Zou

1 Introduction

This second coursework is a simple catalog web app relate to different kind of instruments. It contains five pages which have different functions. Two pages are to exhibit the collection about instruments including the instruction of different instruments and their characters. Users can get

The whole website is consisted of five different pages which are build by html, css and bootstrap. Two pages are to exhibit the collection about instruments including the instruction of different instruments and their characters. Users can get information they want through these pages. The third page is specialize in achieving a function of message board, user can leave their advice and some opinions about instruments. However it is only available after user log in and the message will disappear after users log out. The fourth page is for log in function. In this page, user can input default username and password to log in and get the function only can be achieved after log in. The final page is for a not completed sign up function. In addition, every single page have a different reasonable designed url. When user jump to another page, they would be able to navigate to another site. The web-app achieved transition among different pages and achieved the function of redirects requests and response.In addition, When user input a undefined url, it will display a error page which instead of the original error page to remind user is not valid url.

Every pages except index page inherit the navigation bar of index page by using template inheritance. Therefore, all the navigation bars of different pages are look the exactly the same, and it will be easier when there are some mistakes and some contents in the navigation bar need to change.

1

```
2     {% extends "index.html" %}
3 {% block content %}
4     <div class="container">
5         <div class="row">
6             <div class="col-md-offset-3 col-md-6" style="↩
      padding: 100px 0">
7                 <form method="post" action="/login">
8
9                     <div class="form-group">
10                        <input type="text" class="form-control" ↩
      name="username" placeholder="username">
11                        <i class="fa fa-user"></i>
12                    </div>
13                    <div class="form-group">
14                        <input type="password" class="form-↩
      control" name="password" placeholder="Password">
15                        <i class="fa fa-lock"></i>
16                    </div>
17                    <div>
18                        <button type="submit" class="btn btn-↩
      default">Login</button>
19                    </div>
20                </form>
21            </div>
22        </div>
23    </div>
24    {% if error %}
25
```

2 Design

There are some steps to build the web-app The first step of the coursework was using html build five basic pages and subpages and using links to connect these pages, which achieved the responses between different pages in the browser. Then searching the collections on the internet, putting different types of collection into corresponding pages and utilizing css, bootstrap to make those page look ok.

The next step was establishing two folders in Levinux, 'static' folder to contain all of my static files like css file, images and bootstrap tools, and also builded a 'templates' folder to contain all of the modules. The third step was building a app.py file to set up url for every module and using jinja2 tool to template those modules. Now it can achieve the transition from different pages and URL Hierarchy function, for example, when user click a link under the main page, they will be navigate to a new url based on the url of main page.

After achieve the transition between different pages. The next step is to achieve the message board function. Since we want to achieve the message board can will just be displayed after log in. Therefore, before doing message board, it is better to achieve log in function first. We set up config values "USERNAME"="Zouboyang", and "PASSWORD"="default" then type "app.config.from object(name)" citing config values. Then we turn to the log in page and

input default username and password. However, the web app still can not remember the user was logged in and keep user in log in state. For achieving the message board can only display in log in state, it is necessary to use a session. After user input the correct values in, it can keep user in a log in state by setting session['logged in'] = True. However, before using a session, it is also need to set a secret key as a config value to keep session safe. Through session, the message borad page can make a decision weather or not exhibit the message board by judging if the session['logged in'] = Ture. Futhermore, the "log in" button in navigation bar will convert into "log out" in logged in state. And user can log out by click that button. User will be redirect to index page after they log in successfully and get flash message in index page to remind them they have logged in. Also, if they do not input the right values, they will get a error message in log in page.

Then it is the start of building a message board. We need to put the message users leave in message board in a database. So, first, download the sqlsite3 in Levniux then set a the path of our database in config value "db location = var/zby.db", so we need to create a folder named "var" and create a file named zby.db which is the database we storage message. By define a function get db() to connect to the database. In addtion, by defining a init db() function and a schema.sql can initialize the database when excuting the init db() function in shell. It is a good way to clean the data in database.

```
SECRET_KEY = 'NIMABI'
USERNAME = 'Zouboyang'
PASSWORD = 'default'
app = Flask(__name__)
app.config.from_object(__name__)
db_location = 'var/zby.db'


def get_db():
    db = getattr(g, 'db', None)
    if db is None:
        db = sqlite3.connect(db_location)
        g.db = db
    return db

@app.teardown_appcontext
def close_db_connection(exception):
    db = getattr(g, 'db', None)
    if db is not None:
        db.close()

def init_db():
    with app.app_context():
        db = get_db()
        with app.open_resource('schema.sql', mode='r') as f:
            db.cursor().executescript(f.read())
        db.commit()

@app.before_request
def before_request():
    g.db = get_db()

@app.teardown_request
def teardown_request(exception):
    g.db.close()

```

When user logged in and submit some message in message board. The form will be psot in to another route and the data in the form will be inseert into database then redirect back to the url of message board. Then because now database have data in it, so this time the data in database will be displayed under the messageboard.

```
@app.route('/contact/')
def contact():
    cur = g.db.execute('select title, text from entries order by id desc')
    entries = [dict(title=row[0], text=row[1]) for row in cur.fetchall()]
    return render_template('contact.html', entries=entries)

@app.route('/add/', methods=['POST'])
def add_entry():

    g.db.execute('insert into entries (title, text) values (?, ?)',[request.form['title'], request.form['text']])
    g.db.commit()

    return redirect(url_for('contact'))


```

3 Enhancement

The second coursework is a still a very fundamental web-app. It is a improvement one of the first coursework. Therefore, there are many features need to be added or improved.

First, add more error pages response to different error occured and appropriate information that enables users to make an informed decision about what to do next. Because most users will be confused when some thing is wrong during they are browsing a webpage. They need some relevant guidance to tell them what is going on and induce them how to fix this. Otherwise, they might lose the interest of continue browsing the web.

Second, the web-app needs a user register function, thus user can register their own account in this web and login by using their account. When user are doing the register, they need to submit the information they inputted in the form, those data will be receipted and stored. Therefore, a specialized database needs to be established to manage those data. In addition, a database can also grant and limit access to data based upon criteria such as user name, password, region or account number, databases also enforce data integrity by ensuring that data is collected and presented using a consistent format. This means user can only register successfully by obeying the request to input their information. Currently, this web-app can just login by input a specific username and password. The establishment of database make every user can login to the web-app by using their account they registered with a consistent format. Furthermore, it would display different content in the page because different account.

Thirdly, it is great to add a simple forum to enhance the interaction among different users. Some people who are good at playing guitar can share their experience and put some tutorials on the forum to help those amateurish people. Also, people can exchange different opinions about guitar and get knowledge they do not familiar with. All the data will be stored in the database.

Then a file upload function will be set. After users log in, they will have the chance to upload some file related to guitar such as some reference audio and even some advice of how to improve the website.

Those are the features which are tend to be added and improved.

4 Critical evaluation

The second coursework have some improvements based on the first one. First it by using session the web app can remember the state of log in and display content which is only available after log in rather than there is no differents after user log in. And it exert the data storage with database and achieved the function of message board. People can leave and see their message left in message board. The web-app achieved the requests. The transition form different url which is sensible and will not confused the user. Using two types of request methods "GET" "POST". If method is post, and input the default username and password the page will redirect to the home page.If input the worng information, there will have a hint to remind the user have inputted the wrong thing. And the use of template inheritance properly reduce the instance of HTML embedded in Python code. Many static files such as pictures and CSS were cited for decorating the pages. However, this web-app is far away from a eligible website, it can not manage the data that user uploaded and even have no function to offer users a way to upload their data. Therefore, a good website need a web database to manage and store those data user uploaded. Futhermore, most of the interaction between user and server can not without a database. To sum up, it is a very simplified web-app that just achieved some basic function of a website. There are many place can be improved in next coursework.

5 Personal Evaluation

In this coursework there is a big challenge that I need use the database. It takes me a long time to understand the code how to achieve some specifical function and organized those code because it will not work if there is any mistake in you code.

To sum up, this coursework helped me enhanced my capable of using bootstrap building a website, how to use python and flask, some knowledge about database and help me strength the knowledge I have learned in the last few weeks. It is a very good experience. Helped me figure out many problems.

6 Appendix

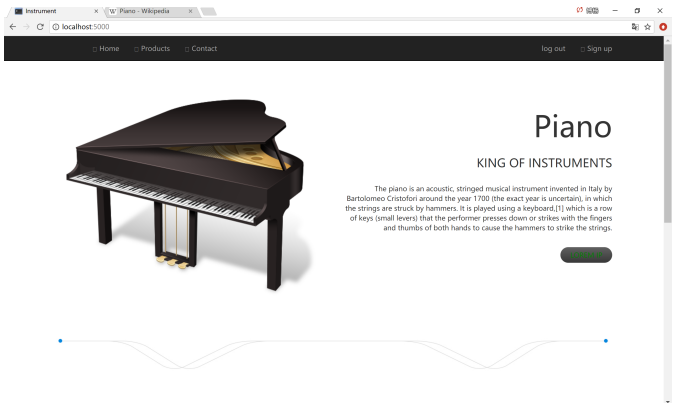

Figure 1: **ImageTitle** - Some Descriptive Text

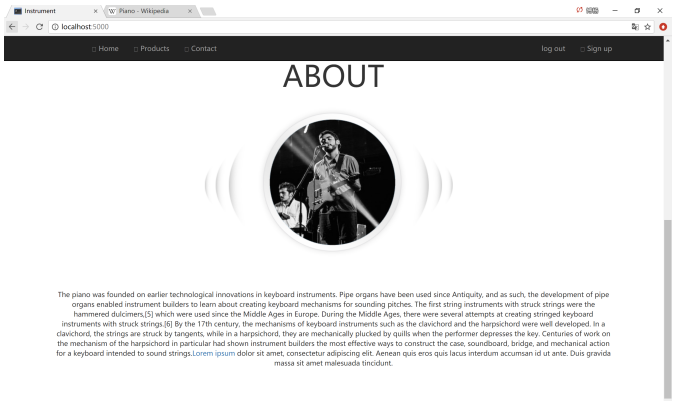

Figure 2: **ImageTitle** - Some Descriptive Text

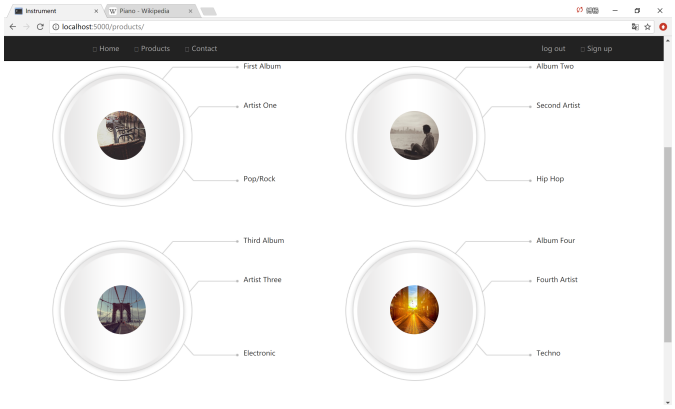

Figure 3: **ImageTitle** - Some Descriptive Text

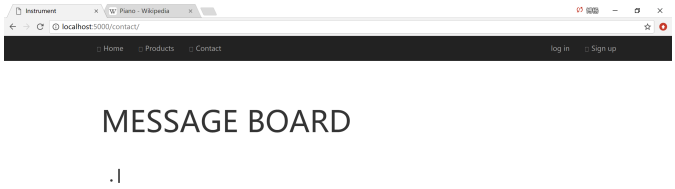

Figure 4: **ImageTitle** - Some Descriptive Text

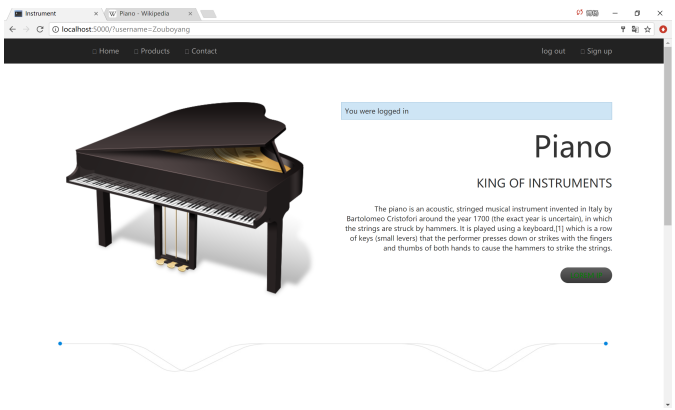

Figure 5: **ImageTitle** - Some Descriptive Text

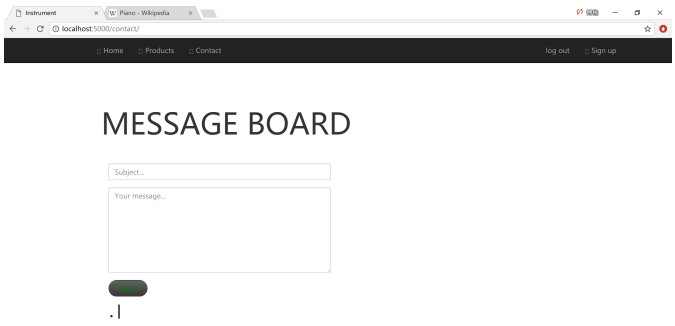

Figure 6: **ImageTitle** - Some Descriptive Text

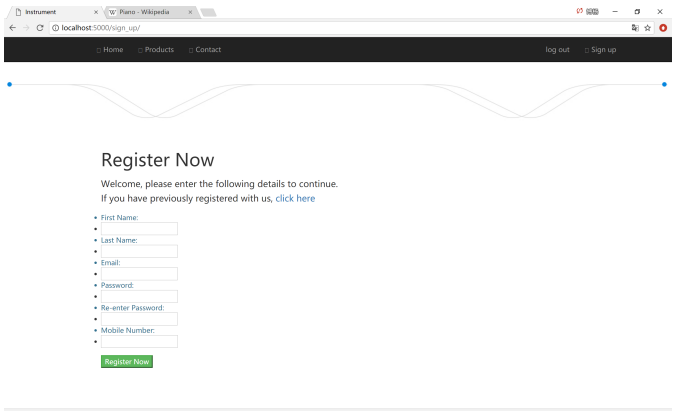

Figure 7: **ImageTitle** - Some Descriptive Text

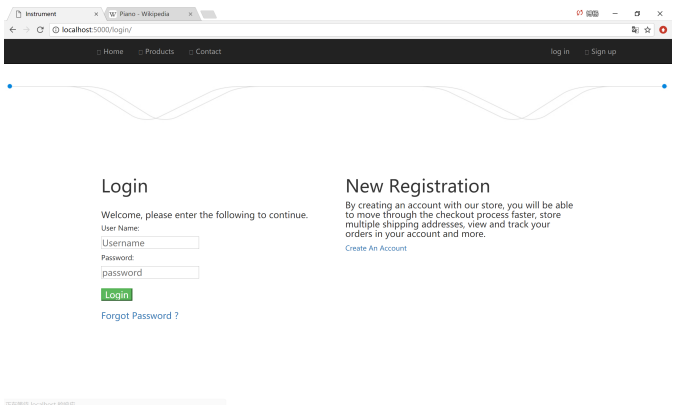

Figure 8: **ImageTitle** - Some Descriptive Text