Advanced Web Technologies Coursework Report

Roche Alexandra
40404291@napier.ac.uk
Edinburgh Napier University - Advanced Web Technologies (SET09103)

Abstract

The goal of this coursework is to create an online application using the Python Flask micro-framework. It must demonstrate the mastery of knowledge in Python, Flask, but also in frontend technologies (HTML, CSS, js), in database management and website structures and security. I chose to continue my application from previous coursework : JapLanguageSchools. It is a website that indexes Japanese language schools in Japan, providing global information about schools and programs and containing a login system with users and administrators. To do so, I used tools as Python Flask for the website structure, Jinja2 which is Flask's template tool and Bootstrap for the aesthetic aspect of the website. This report contains all conception and implementation steps with screenshots of the final application.

1 Introduction

Concept The purpose of **JapLanguageSchools** is to provide to students that want to study in a language school in Japan necessary information to help them to choose a school. This web-app is linked to a database containing information about Japanese language schools in Japan. This information is about schools (Name, location, website, accommodation...) and programs details (Duration, costs). From the app, you can access directly to the list of all schools, of all programs or you can use filters to specify your search. It is possible to filter results by determining a location (City), price range or course duration. You can also do a specific search of a school by entering a name, a city or a district in the research field.
JapLanguageSchools also provide a login system which allows users to register to the website. Once registered, users are able to save schools or programs as favorites and to send a review about a school registered in the app. In order to check relevance of reviews, some users ranked as administrators can accept or refuse them. Once accepted, the review is published on the school page.

Content From the home page of the app (Figure 1), you can access different pages. The first option is to display the entire school list or program list (Figure 2), and the second is to sort results by categories[1]. I chose to implement three categories: City, price range and program duration. There are different options for each. For example, by selecting to sort by duration, it is possible to display 12 18 or 24 month program's duration. Once you chose a sorting, all matching programs are displayed. More important information is displayed for each programs (School name, location, duration, accomodation type etc...). By clicking on "More information", the user is redirected on the school page[2], containing all information about the selected establishment. The same system is implemented for school sorting.
All those options are accessible from the navigation bar. Moreover, you can access the register/login page and change the current currency. Indeed, because the website is destined to be used internationally, it is important for the user to be able to select a currency to convert prices. On a school page, we see school's location and programs details. Each program is displayed in a specific tab. Under programs tabs, validated reviews are displayed. By the user's profile page, the user can visualize his favorite schools and programs and also check the status of reviews he gave, to see if it has been published. Users which have been designated as administrator are allowed to check reviews and to accept them or not.

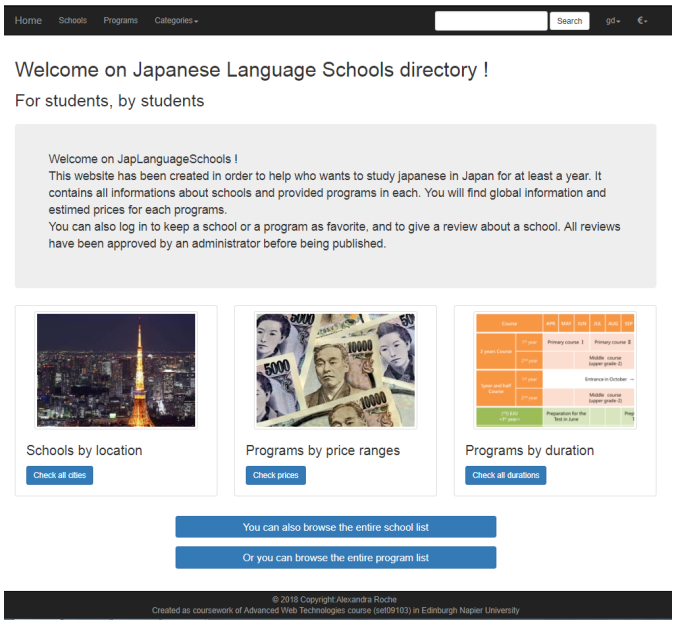


Figure 1: **Main Page -** You can select categories or chose to display all schools

[1] Screenshot in Appendix 1

[2] Screenshot in Appendix 2

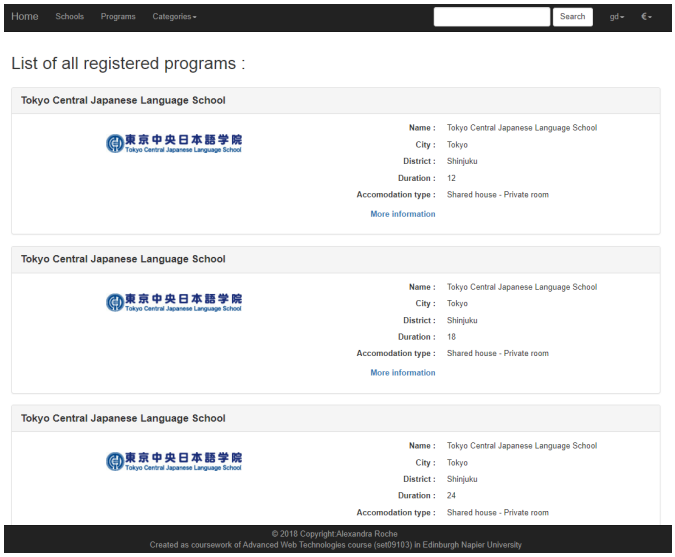


Figure 2: **Programs display page -** List all programs and display some information

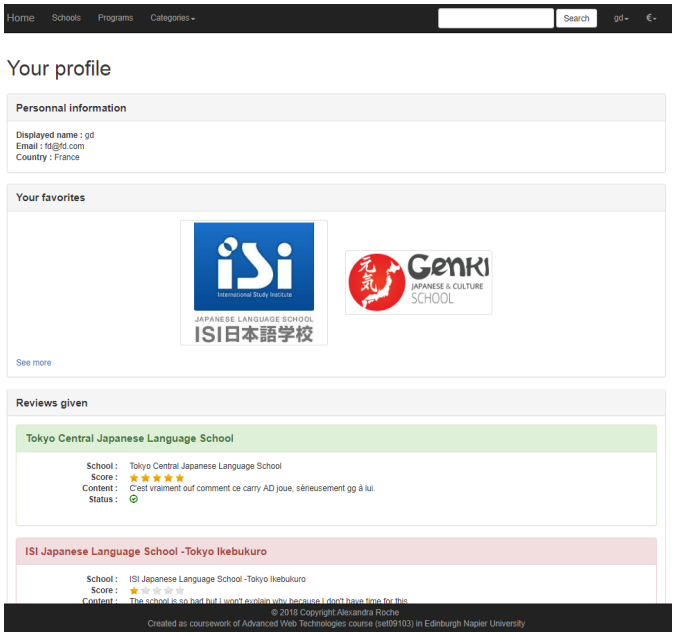


Figure 3: **User's profile page -** Contains user's information, favorites and reviews

2 Design

Website structure Because JapLanguageSchools is an online directory, it has to be efficient, that means that the user should find the requested information as simply and as fast as possible. So the website structure is fundamental in this process, it should be clear and straightforward. So the main page offers all available research options to the user: Display all schools, or chose a category. Those options are accessible from all the website's pages from the navigation bar on the top of the screen. It avoids the user to be lost on the application and to be able to change the sorting option at any moment. Category pages show all sorting options and also show by advance the number of results depending on the sorting choice. The navigation bar also contains a search field, where the user can search a specific school by entering a city, a district or a part of a school name. By searching, it redirects the user on the search page where results are displayed. From results, there is an access to the school page containing school's details and its location on a Google Map widget. It is all the more useful that it allows to save places as favorites and to check them on Google Maps application. Under school's information, tabs separate programs and give prices some features. Programs are distinguished either by the accomodation or the duration, so it allows the user to compare prices easily. To do so, I have decided to implement a currency system where the user should be able to choose between american dollars, pounds or euros to display converted prices. That option is available from the navigation bar and can be changed at any moment. This option has been made with a session variable, that contains currency name. Note that it will also affect the price sorting option.
For the reviews, I chose to save every of them in a database table and to update an attribute called "status" in order to know if they are accepted, refused or not checked. For the notation, I used a small jQuery plugin called raty[3] that display star rating. To make the email contact with administrators when a review is submited, I was based on a mail class made by Simon Wells[4] that send email to a recipient's adress given as parameter. This email is sent from a created mailbox, *japlangsch.hikaweb@gmail.com*. It also seemed important to add the checking date and the administrator who checked the review in the database, to have a better control on what has been done on the application. Application users can register and/or login from specific pages that are accessible from the navbar. From the user page, he can visualize favorite schools for a shortcut to their pages. By adding a program to his favorites, an user is then able to check most important prices and information of each program and to compare them in a global view. An user can also check the content and the status of reviews he gave. For each, it shows if it has been accepted, refused or if it still waits for a checking.

URL structure The website's homepage is on the root (URL *'/'*). From this page, the user can access on the website's five main pages, as shown in figure 3. From the school display page, you can access to each school pages. Those school pages have a URL which specified the school number (this number corresponding to the index of the school in the database). For the sorting options pages, the URL contains the value of the selected category (price, city or duration). When an option is selected, it appears in the URL as a parameter. When the user clicks on the button to get more information on a school from a sorting results page, he is redirected to the school details page on the URL *'/school/<schId>*'
Users can access thier profiles by the simple url */profile/*. This url is protected by a verification function that makes that only authentified users can access that page. A spe-

[3]https://github.com/wbotelhos/raty
[4]https://github.com/siwells

cific page for favorites is available from the url */profile/favorites/*.
About the reviews, url are based on actions. To submit a review, the url will be on the same base than the school page : */school/<schId>/submit-review*. The link sent to administrators to check reviews is a more direct url from the root containing the review id */check-review/<reviewId>*
Some urls are used in the application but only have a redirecting role. For exemple, to change the currency the user will go to the url */change-currency/<currency>* but will be redirected by the app to the page where the request comes from. So the user will just think that the page has been reloaded and set the currency. It the same case for submit a review, accept or refuse a review etc.

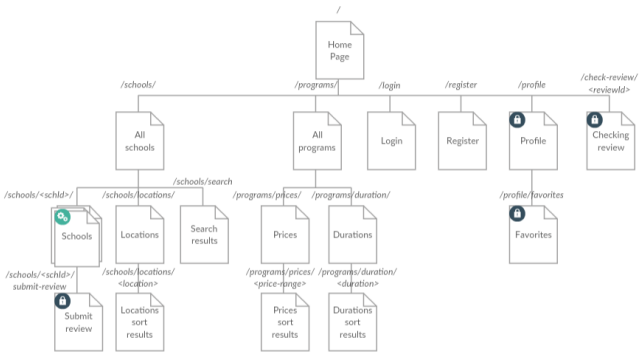


Figure 4: **URL Hierachy** - From the level 0 (Root) to level 3. Lockers represent a rank checking before access to the page

3 Enhancements

Because this work is in continuity with the first coursework, we can take a look of improvments that I suggested at the end of the previous work and that I have implemented.

- I changed my database structure. In the first coursework, there was only one table, "*schools*" and now I have separated schools from programs in order to be closer to the user's needs and to make comparaisons available. The database now also contains "*users*", "*favorites*" and "*reviews*" tables. The database representation is given on Appendix 3.
- I have added a Google Widget in school pages to show their locations.
- I have implemented a login system where users can register and access to new features such as the favorite system and the review system. Every access to the database is made by prepared SQL statements which reduce risk of SQL injections and a function checks that only authentificated users have access to some pages.
- I have added a currency option to users by a session value.

However, there is still a lot of possible improvments that can be made in oder to improve user experience on JapLanguageSchool.

- I would like to improve the way to update the database. For exemple, put a way for administrators to update schools or programs data. It will avoid to modify it directly with SQL commands.
- Then, I would like to add a contact form. In the same way than the reviews submission system, it will allows schools to contact application's administrators and to transfer price details or school information by email. In this way, contact form will also allows the sender of email to join attached files such as pricing brochures.
- Another improvment is in an aesthetic aspect. I would like to spend time to improve visual of my application, by more personnal css (And not just rely on Bootstrap) and js animations for exemple.
- We can also add a translation system for reviews to allow users to read reviews depending on their connection's location's language.

4 Critical Evaluation

When I chose the kind of directory I wanted to implement, I was pleased to find a topic which I am concerned by. That allows me to know what the user's purpose is when using this website. From this point, I deduce how to build my application.
In my opinion, information displaying is a strength of my application. The preview of each school in sorting pages or search pages shows key point information, and it can reduce the necessity for the user to visualize the entire school page or can help for comparison between schools. Also, the search system works well and allows a search by name (or partial name), city or district. Some towns in Japan are so big that just giving only the city do not provide enough information to the user. That is another example of implementation that I have made because I am interested in the topic and because I know that it is useful in some cases.
I also feel that my code is a bit poor because it is unclear and not readable by someone else. I need to be more focus on this point on my future works because when the project grows bigger and bigger, I am getting lost in my code because the structure is not good enough. I need to separate accesses to the database for exemple and to create more functions.

5 Personal Evaluation

Process and challenges For this project, I have chosen to work only by the Linux terminal to access the server. Even if I know bash language for two years, doing an entire project with VIM and without any IDE provided with a graphic interface was challenging. I already used some provided tools such as Bootstrap or Python, but I never used Python to create a website (I used Php most often). Thanks to practical classes, Python Flask was easy to

handle. The most complicated part was to transfer data between Python flask code and templates using Jinja2. Jinja2, in general, was a new tool for me, and it took me some time to know how to use it. Hopefully, thanks to all documentation available on the internet, I discovered all Jinja content pretty fast, and it didn't block me that much. One example of challenge that I met was when I have implemented Google Maps Widget on my schools pages. This widget uses parameters that we can find on a Google Maps url to find and define a place. More often, those parameters are place's name. But because most part of school names are in Japanese, it caused encoding problems in my database. It can seems like an esay task to add a Google Maps widget but not everytime. To fix this point, I had to use an url encoder to change Japanese characters into an UTF-8 encoding and it worked.
Visual aspect of the app is a bit unsatisfactory because it rests on Bootstrap composants. With more time, it can be improved and be more attractive to the user. I also realized that I need to look more into Javascript, JQuery and all JS libraries. I think that I miss something and that I will soon get a closer look to it.

Personnal Feedback First and foremost, I was pleased to see that we had the choice of our topic. I spent a really good time working on JapLanguageSchools, mostly because I needed to research language schools in Japan in the first place. So this work was a way to do a "two-in-one" and maybe this application will allow me to share more easily my research with my family and friends to choose my future studies.
I think that I will continue that project after the end of Advanced Web Technologies. I really want to continue improving it and to make as functionnal as possible. I may put it in production once I will be sure that everything works. For now, I have some feedbacks from my family and my friends that told me that I should put it in production and to let other people use it as well. That encourages me to work on it. Maybe one day that website will help people to choose their language school, and in a future far far away, it will also contains schools from other contries and become a reference platform.
For my futures web-applications, it is possible that I will use Python to create them. After those coursework, I feel confident about working with Flask but I will also work with a Python IDE (Not with vim as I did during courseworks) in order to have more features as database visualization for exemple.
My work may be less impressive or less original than others, but I always wanted to make useful tools that maybe not everyone is going to use, but that miss to a community or that actual tools are not efficient enough. It is what I have done with this work, and in general I am really proud of what I have made during those 3 months.

6 Bibliography

References

[1] *StackOverflow*.
https://stackoverflow.com

[2] Armin Ronacher
Jinja2 Documentation.
http://jinja.pocoo.org/docs/2.10/

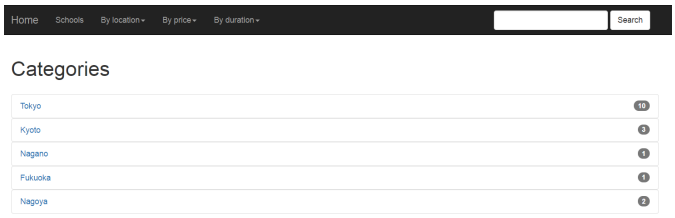
[3] Armin Ronacher
Flask documentation
http://flask.pocoo.org

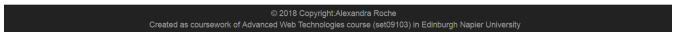
[4] Henrik Huttunen
Vim tutorial
https://www.openvim.com

[5] *Python tutorial (French)*
https://openclassrooms.com/fr/courses/235344-apprenez-a-programmer-en-python

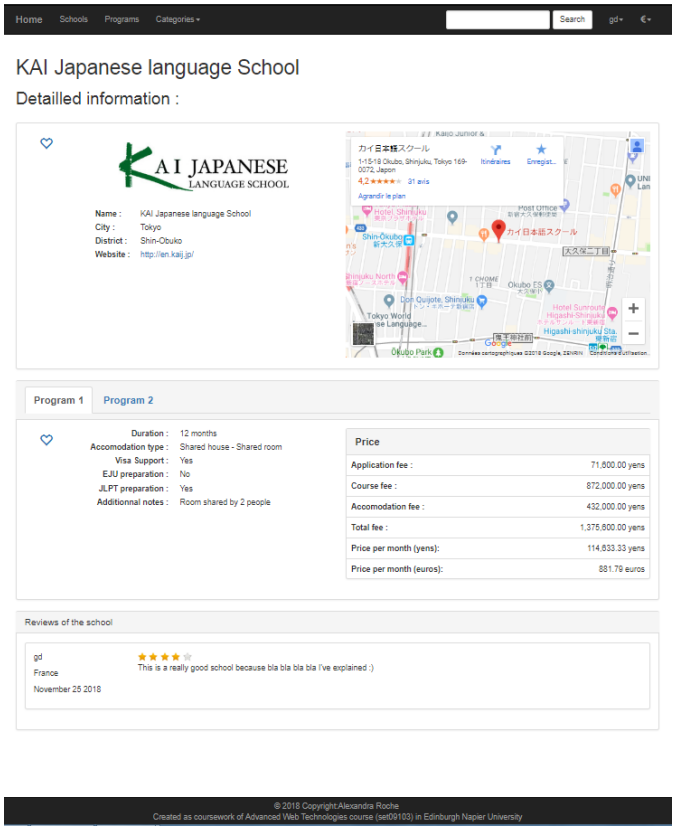
[6] Washington Botelho
JQuery Raty documentation
https://github.com/wbotelhos/raty/blob/master/README.md

7 Appendix

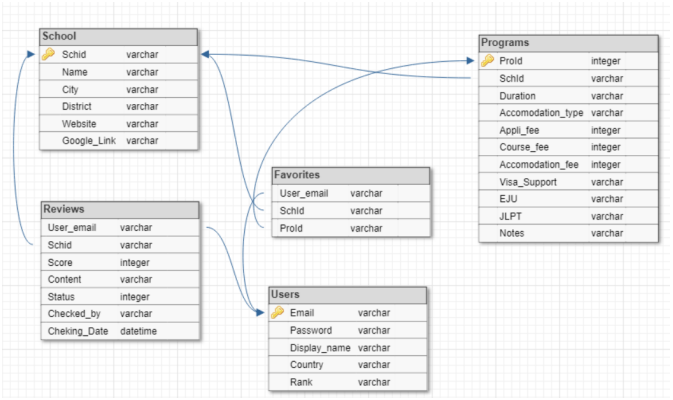




Appendix 1 - Page where categories are displayed



Appendix 2 - School page with information



Appendix 3 - Application's database's structure

```
1    pip install --user flask
2    pip install --user bycrypt
3    python -c "import flask"
4    export FLASK_APP = website.py
5    python -m flask run --host=0.0.0.0 --port 9176
6
```

Appendix 4 - Application's running commands from Napier server