# Veronika’s favourite movies catalogue report

# 40212551

## Introduction

The aim of this coursework was to design and implement a prototype of an online catalogue web application. My online catalogue is a collection of my favourite movies. This was achieved by using the virtual server Levinux and the command line logged in using SSH. The web app was built using micro-framework Python Flask, which has many useful tools, such as debugging tools or templating using Jinja2.

## Design

URL hierarchy was designed as different genres: comedies, drama, animation and sci-fi, which are appropriate for finding and retrieving information about collection. I researched other movie sites to see how they categorize their databases, e.g. in IMDB users can search by many more parameters: actors, soundtracks, producers, etc. This categorization would be useful for bigger projects or if my collection was larger.

To style the app was used Bootstrap framework and ‘United’ free theme [1] was used to add orange styling and distinctive Ubuntu font. The use of this theme is mentioned in CSS comment to credit the authors.

To improve the design and user experience appropriate images downloaded from internet were added. These are covered by creative common license, so could not be used without permission if the app was deployed on the publically accessible server. The images, favicon and Bootstrap CSS are accessed by the Flask “url\_for” function.

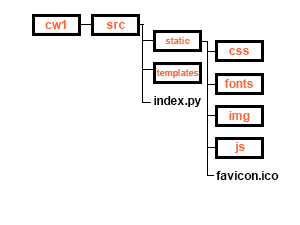


Figure 1: Folders and files structure

### Flask routing, redirects, template inheritance

The app is deployed by index.py. Each route (web address) uses appropriate template to generate HTML page and loads the data with JSON file. Folder src contains file index.py (that deploys the app) and static and templates folders. Static folder contains CSS, fonts, images, JavaScript and favicon. Templates folder stores all HTML pages. The index.py file contains different trailing slashes to redirect user to different URL pages: Films, Comedies, Drama, Animation, Sci-fi and Error. HTML is generated by render\_template function. The HTML templates include Jinja2 variable tags indicated by {{ }} for title using conditional behaviour.

### Flask error

To better fit in with the design of the app personalized error page was created. The debugging tool was set to “True” to print out debug information, which helps with designing the content. This would be changed back to “False” if the web app ran on a public server to prevent security breach.

### JSON

All data for my catalogue are saved in JSON file films.json. I used [2] suggestion to import the data to my html pages using variable tags indicated by {{ }}.

## Enhancements

To improve user experience and overall design, I would include:

* search option to easily search through the database,
* form to upload new movies,
* other searchable categories, e.g. actors, year, directors, more genres, etc.,
* different users option, e.g. sessions to hide mature content from users who are underage,
* improved template inheritance to help easily change sub-parts of individual pages.
* trailer videos or film streaming,
* social media recommender systems, such as Facebook likes,
* about me page.

## Critical evaluation

My web-app meets the criteria of the coursework. Although only 6 films were used, this is sufficient to show the level of functionality required.

If more time was allowed, more categories to choose from, more films, trailers or films streaming would be added. In this case search function would be very important to access information.

Another change would be the template inheritance to define header, footer or any other sub-part of individual pages in app to change and manage the look and feel of the app very easily.

## Personal evaluation

By completing this coursework, I learnt to use the dynamic programming language Python and its library. I constructed a web application that can run across different software platforms.

I familiarized myself with terminal command line and non-graphical text editor Vim. To learn how to use vim I used online tutorials [3], [4].

I learnt how to construct a simple JSON file and load its data to HTML pages using Python.

I used the command line application Git as my source control system, which I familiarized with at [5] and [6].

The source code was stored by pushing into my personal GitHub account repository. To easily transfer files from my computer to GitHub and Levinux respectively I tried to use a file management tool CyberDuck. This tool unfortunately did not connect to GitHub, therefore I found a different solution [7] and downloaded GitHub Desktop. This tool allowed me to save a copy of a GitHub repository on my local machine, make changes easily in Visual Studio Code and synchronize back to GitHub.

## Resources

|  |  |
| --- | --- |
| * Levinux | * Python |
| * Python Flask | * Python Jinja2 |
| * HTML/CSS/JS/JSON | * Bootstrap |
| * Vim | * GitHub Desktop |
| * Git | * GitHub |

## References

[1] "Bootswatch: United". *Bootswatch.com/united*. N.p., 2016. Web. 16 Oct. 2016.

[2] "Flask Load Local Json". *stackoverflow.com/questions/21133976/flask-load-local-json*. N.p., 2016. Web. 20 Oct. 2016.

[3] "What Is Vi?". *linuxzoo.net/page/vi\_what.html*. N.p., 2016. Web. 20 Oct. 2016.

[4] "Interactive Vim Tutorial". *Openvim.com*. N.p., 2016. Web. 20 Oct. 2016.

[5] "Git - Book". *Git-scm.com/book*. N.p., 2016. Web. 22 Oct. 2016.

[6] "Git Immersion - Brought To You By Neo". *Gitimmersion.com*. N.p., 2016. Web. 22 Oct. 2016.

[7] "The Big Github Desktop And Sourcetree Throwdown!". *blog.wppusher.com/github-and-sourcetree-throwdown/*. N.p., 2016. Web. 23 Oct. 2016.