

# CA: Advanced Programming

Module Title: Advanced Programming

Module Code: B8IT150

Module Leader: Paul Laird

Student Name: Wellington Nodari

Student Code: 10558156

Contents

[CA: Advanced Programming 1](#_Toc104921538)

[1. Project Overview/Scope 3](#_Toc104921539)

[2. Development 4](#_Toc104921540)

[3. Reflections on Learning 7](#_Toc104921541)

[4. References 9](#_Toc104921542)

[5. Relevant URLs 11](#_Toc104921543)

# Project Overview/Scope

In this assignment, I’ve created a information system aimed at a online courses provider, where the user/ student will be able to access public content, for selecting the course he will.

Once selected the module, user will have to enroll and create a user name and password. When logged in the user will have access to only the content has enrolled for.

The resources used for completing this project are listed below:

* Back-end: Python/ FLASK
* Front-end: HTML, CSS, JavaScript
* Database: SQLite3
* Heroku Cloud Application Platform
* Pycharm IDE
* DB Browser for SQLite
* Git GUI/ GitHub for a repository

Using Python/ Flask for creating the back-end was a great experience, where its tools, packages, and simplicity made the daily work very intuitive and enjoyable. Learning, applying and experimenting a real world tasks for a back-end engineer has opened my sight about such role/position towards a project.

For the front-end, I was able to apply most knowledge obtained on the Web Design classes, most of the concepts used from HTML and CSS I just needed to recap. What I could say at some point became challenging was the JavaScript, used for some UX/UI functions as hide/show the enrolment form.

The SQLite3 used, offered me simplicity for deploy it and was perfectly able to accomplish what was necessary for the requirements on this website. There are some considerations regarding this DB that I will describe in detail later in this document.

For the cloud service chosen, Heroku was the one that suited me better, again I have made my choices by simplicity and flexibility in getting my application deployed. With no concerns of managing servers, infrastructure and hardware, it provided me all the services and tools I needed.

Choosing the IDE for this project, maybe was the most hard choice I need to make at its very begin, maybe by lack of knowledge (in details) of more choices and what vantages each one provides. Pycharm offered me the possibility to manage the whole project, from back to the front-end, its terminal worked very well when it was needed for commands such managing the commits to the repository, connecting to the cloud, managing the DB etc.

The DB Browser for SQLite was chosen for making everything more intuitive and easy to manage on the DB. It offered flexibility for managing the tables and data when the testing and deploying phase where in place.

About the repositories I have started using Git, which is the Heroku standard. It was very simple to install, create a Heroku repo and deploy it, but after struggling a few hours in how to make my repo public, I just decided to change to GitHub, which own a simple and more user friendly interface.

# Development

In topic I will address the following subjects by dividing it into 3 parts:

1. Project/ Development Overview
2. Requirements
3. Code attributions

**Project/ Development Overview**

The project was developed under a Agile methodology, where new features and concepts where developed, tested, validated and committed to production.

My main objective in creating this app was to demonstrate a useful website for a online courses provider, allowing the user browser the subjects available as a public content. When the subject is selected the user must to enroll on it by filling the enrolment form and providing the following details:

* First Name
* Last Name
* Email address
* Course (selecting it by a drop down menu)
* Password
* Confirm Password

This action will feed the ‘myschool.db’ into 3 tables ‘students’, ‘loginCred’ and ‘std\_enroll’.

The student table receives the students personal details, keeping it in a simple way, the data stored on it are just fname, lname and email.

The loginCred tables will deal with the user login details, having the student email and password.

The last table that receives data from the user is ‘std\_enroll’ (ps. Short for student enrolment), this will be used for identify which course the student is enrolled to. The data on this table are email, course name and enroll date.

There is another table on the database called courses, where all the available course are and also duration time and prices. From this table user will access the dropdown menu on the enrolment form.

As soon the user enrolls in a given course, he can log in on it from the login page (home) providing their email address as user name and password. Once he’s in, the system will only allow the access to the specific content enrolled to.

The back-end is made on Python/ Flask and using it on this project provided me a real hands on working as developer. Of course I’ve had to make lots of researches to accomplish several desired features as the login, enroll, get and post data from DB and handle all the system routes in a properly way.

All the front-end is made with HTML, CSS and some JS. I have used almost everything that I’ve learned on the Web Design module, so I thing this part wasn’t so challenging, but the part to deal with JavaScript, what it’s not so hard, but when dealing with timeframes and goal dates, I chose not focus too much on it. The main usage for JS was the feature to hide/show the enroll form and make the user interface more approachable.

The git usage was at the very beginning quite hard (to me) to understand and to find a better manner to manage. You will observe several commits on the same day, this had happen because I haven’t realise I could use the localhost for testing and validate changes before sending to production.

It created me some issues, bringing me to roll back my app for at least two times, but for a learning perspective I think it was good for experimenting the issues that this ‘bad habit’ can cause.

After having learned the hard way, I started to define what was necessary to implement/work on, test it on the localhost and commit it, separating it by tasks accomplished and validated.

Also, when I was looking for making my repository public I was struggling a lot for many hours on how to do that. As I was using the Heroku’s default git, and it seems to be a very ‘basic’ tool or not so user friendly, I’ve decided to change the repo to github, where it is easily manageable for sharing the content for whom will assessing this project.

**Requirements**

System Functional Requirements

1. Enrolment/ subscription
2. Login/ logout – using cookies
3. Http requests – GET and POST
4. CRUD
5. Memory - 512MB RAM
6. Elasticity – up to 1GB RAM
7. Social Media link buttons
8. User’s data will be stored into a DB.

System Non-functional Requirements

1. Password and password confirmation must match in the users enroll
2. Business security - If user tries to access a content that he’s not enrolled to and not logged in, by typing it in the URL, system will raise an error message
3. Latency – Free cloud services may cause low-latency conditions
4. Stability – Application should stay stable and online
5. Sleep – The free cloud service will put the application into sleep state if there is no request after 30 min.
6. Providing data security and confidentiality for user’s data provided when enrolled.

User Functional Requirements

1. Login/ logout
2. Enroll/Subscription
3. The user email address must be unique for enrolling on a course
4. Login required – for accessing specific content

User Non-functional Requirements

1. High availability
2. User friendly interface – links and buttons hover sensitive
3. Usability
4. Shortcuts for social medias

**Code attributions**

I have used many fonts of ideas for constructing this project, as some books, websites and YouTube videos.

It was rarely the occasions where I used exactly the same code from the font, as my idea always was to understand what was the main concept and apply it into my work.

See these cases below:

myApp.py – lines 18 to 26

Text

Description automatically generated

Taken from: <https://www.youtube.com/watch?v=_pzMDIi5BuI>

index.html – lines 60 to 70

Text

Description automatically generated

Taken from: <https://bobbyhadz.com/blog/javascript-show-hide-form-on-click>

All files “\_content.html” – lines 42 to 78



Taken from: <https://developers.google.com/youtube/player_parameters>

# Reflections on Learning

This project offered me an amazing experience on how to develop a real world application from scratch, the opportunity of using and apply the concepts that were taught in this module and even on other ones was at the same time challenging and enjoyable.

It made me revisit, recap and go further on learning more concepts that when in the class I had no idea on how, when and where it should be used. I can list some of these subjects as:

* Web Design
  + HTML and CSS
  + JavaScript
* Database Design and Dev.
  + Architecture – Designing and creating tables
  + Relationships – Creating relationships between tables
  + SQL – Using statements for manage data to get from and insert to the DB
* Information Systems Dev and Management
  + Agile Methodology
  + Project management
* Web and Cloud
  + Cloud applications
  + Cloud computing
* Advanced Programming
  + Python as Back-end
    - Route & Decorators usage
    - Connecting and communicate to a DB
    - Render templates
    - Creating sessions (login and logout)
    - Creating sessions required
    - Handling GET and POST requests
    - Python Packages
      * Flask
      * Functools
      * Sqlite3
      * Datetime
  + Git/ GitHub
    - How to create a repository
    - How to manage and commit
    - How to manage and compare
    - Migrating repositories
  + Requirements
    - System/ Users
    - Functional/ Non Functional

**Things that could be improved**

In this topic, I would like to list my awareness of the weaknesses of my application, the reasons for these features not being applied, and ideas for resolving it.

* SQLite3

It is the Heroku data base by default, and also when researching about it, got informed that SQLite3 is ideal for users getting started and learning about. All the functions the DB is seemed to do is working properly, getting users data from tables for logins and inserting data into tables from enrolment form.

The main issue is that SQLite runs in memory, and backs ups the data in files on disk. This is a great strategy for development, but cannot be applied in production, because the data written on it would be lost in up to 24hrs as Heroku has an ephemeral filesystem and all contents are cleared periodically.

I tried to implement Postgres, which is a free add-on provided by Heroku, but I faced some issues in doing it so. As I had a short timeframe to deliver the project, I was afraid to lose all the DB and tables already created.

The idea for resolving this issue and have a functional DB in production would be deploy Postgres.

* Payment Method

When working on the enrolment form, I started looking for API’s for use as a payment method, checked a few as PayPal, Stripe, Google Pay.

The main situation that I haven’t used or worked on this feature was that all the providers above provide the code for integrating the API (I wasn’t sure about copying the whole thing into my code), even though I tested Stripe API on my localhost but with nothing worked properly. After some time, I’ve decided to leave the idea apart.

Having more time to work on it I am sure I can make it work, it will be implemented on my final project.

* Enrolment Form validators

When the user is filling out the enrolment form, the only validator used is the comparison of the password and password confirmation matching. This was a personal decision to focus on other features and apply these validators later on if I had time to apply and test it.

This feature will be applied to my final project too, the idea is to apply using the ‘WTForms’ Package.

# References

Lutz, M. and O'reilly Media (2015). *Python pocket reference*. Sebastopol Etc.: O’reilly Media.

flask.palletsprojects.com. (n.d.). *Welcome to Flask — Flask Documentation (2.1.x)*. [online] Available at: https://flask.palletsprojects.com/en/2.1.x/.

flask.palletsprojects.com. (n.d.). *Form Validation with WTForms — Flask Documentation (2.1.x)*. [online] Available at: https://flask.palletsprojects.com/en/2.1.x/patterns/wtforms/ [Accessed 31 May 2022].

pythonbasics.org. (n.d.). *Flask HTTP methods, handle GET & POST requests - Python Tutorial*. [online] Available at: https://pythonbasics.org/flask-http-methods/.

‌

W3schools.com. (2019). *Python Tutorial*. [online] Available at: https://www.w3schools.com/python/default.asp.

‌ MDN Web Docs. (n.d.). *EventTarget.addEventListener()*. [online] Available at: https://developer.mozilla.org/en-US/docs/Web/API/EventTarget/addEventListener.

bobbyhadz.com. (n.d.). *Show/Hide a Form on button click using JavaScript | bobbyhadz*. [online] Available at: https://bobbyhadz.com/blog/javascript-show-hide-form-on-click [Accessed 31 May 2022].

www.youtube.com. (n.d.). *Discover Flask, Part 21 - User Authentication*. [online] Available at: https://www.youtube.com/watch?v=\_pzMDIi5BuI [Accessed 31 May 2022].

‌GitHub. (2022). *Discover Flask*. [online] Available at: https://github.com/realpython/discover-flask [Accessed 31 May 2022].

‌GeeksforGeeks. (n.d.). *SQL Tutorial*. [online] Available at: https://www.geeksforgeeks.org/sql-tutorial/.

W3Schools (2019). *CSS Tutorial*. [online] W3schools.com. Available at: https://www.w3schools.com/css/default.asp.

‌www.youtube.com. (n.d.). *How to install SQLite3 on Windows 10 [ 2021 Update ] SQL Programming | Complete Guide*. [online] Available at: https://www.youtube.com/watch?v=QZx6zEq9o2k&list=LL&index=2 [Accessed 31 May 2022].

‌

www.youtube.com. (n.d.). *Flask Tutorial (part 3) - sessions, login\_required decorator, debugging*. [online] Available at: https://www.youtube.com/watch?v=WCpNvteLCDI&t=330s [Accessed 31 May 2022].

‌www.google.com. (n.d.). *hex color - Google Search*. [online] Available at: https://www.google.com/search?q=hex+color&rlz=1C1JJTC\_enIE952IE952&oq=hex&aqs=chrome.2.69i57j69i59l2j46i131i199i433i465i512j0i131i433i512j69i60j69i61l2.6583j0j4&sourceid=chrome&ie=UTF-8 [Accessed 31 May 2022].

www.sqlite.org. (n.d.). *Command Line Shell For SQLite*. [online] Available at: https://www.sqlite.org/cli.html [Accessed 31 May 2022].

www.youtube.com. (n.d.). What is Data Science? | Great Learning. [online] Available at: https://www.youtube.com/watch?v=Nrfht\_c3T7w [Accessed 31 May 2022].

CareerFoundry (2021). *UX Design vs UI Design - What’s The Difference? (2021)*. *YouTube*. Available at: https://www.youtube.com/watch?v=5CxXhyhT6Fc [Accessed 25 May 2021].

‌ Testaholic (2016). *Software Development Lifecycle in 9 minutes!* *YouTube*. Available at: https://www.youtube.com/watch?v=i-QyW8D3ei0.

‌

www.youtube.com. (n.d.). *DevOps In 5 Minutes | What Is DevOps?| DevOps Explained | DevOps Tutorial For Beginners |Simplilearn*. [online] Available at: https://www.youtube.com/watch?v=Xrgk023l4lI.

Simplilearn (2020). *Cyber Security In 7 Minutes | What Is Cyber Security: How It Works? | Cyber Security | Simplilearn*. *YouTube*. Available at: https://www.youtube.com/watch?v=inWWhr5tnEA.

Simplilearn (2019). *Blockchain In 7 Minutes | What Is Blockchain | Blockchain Explained Simply | Blockchain |Simplilearn*. *YouTube*. Available at: https://www.youtube.com/watch?v=yubzJw0uiE4.

‌

Player Parameters (2018). *YouTube Embedded Players and Player Parameters  |  YouTube IFrame Player API  |  Google Developers*. [online] Google Developers. Available at: https://developers.google.com/youtube/player\_parameters.

‌

## 5. Relevant URLs

Application URL:

<https://cawellington.herokuapp.com/>

GitHub Project Public Repository:

<https://github.com/Wellington-Nodari/CA-AdvProgmng.git>

Heroku URL (Paul Laird is added as collaborator in this project):

<https://dashboard.heroku.com/apps/cawellington/>