PYTHON WEB TUTORIAL

Instructor: Joshua John

Outline

- Goal
- Prerequisites
- Getting Started
- A simple HTML Page
- HTML with CSS
- Adding Bootstrap
- Installing Python with Others
- Installing Flask
- Building our first web app
- MAIN EVENT

Goal

To develop and host a staff management app

- Features: Admin creates, updates, view and deletes staff
- Admin login and register

Prerequisites

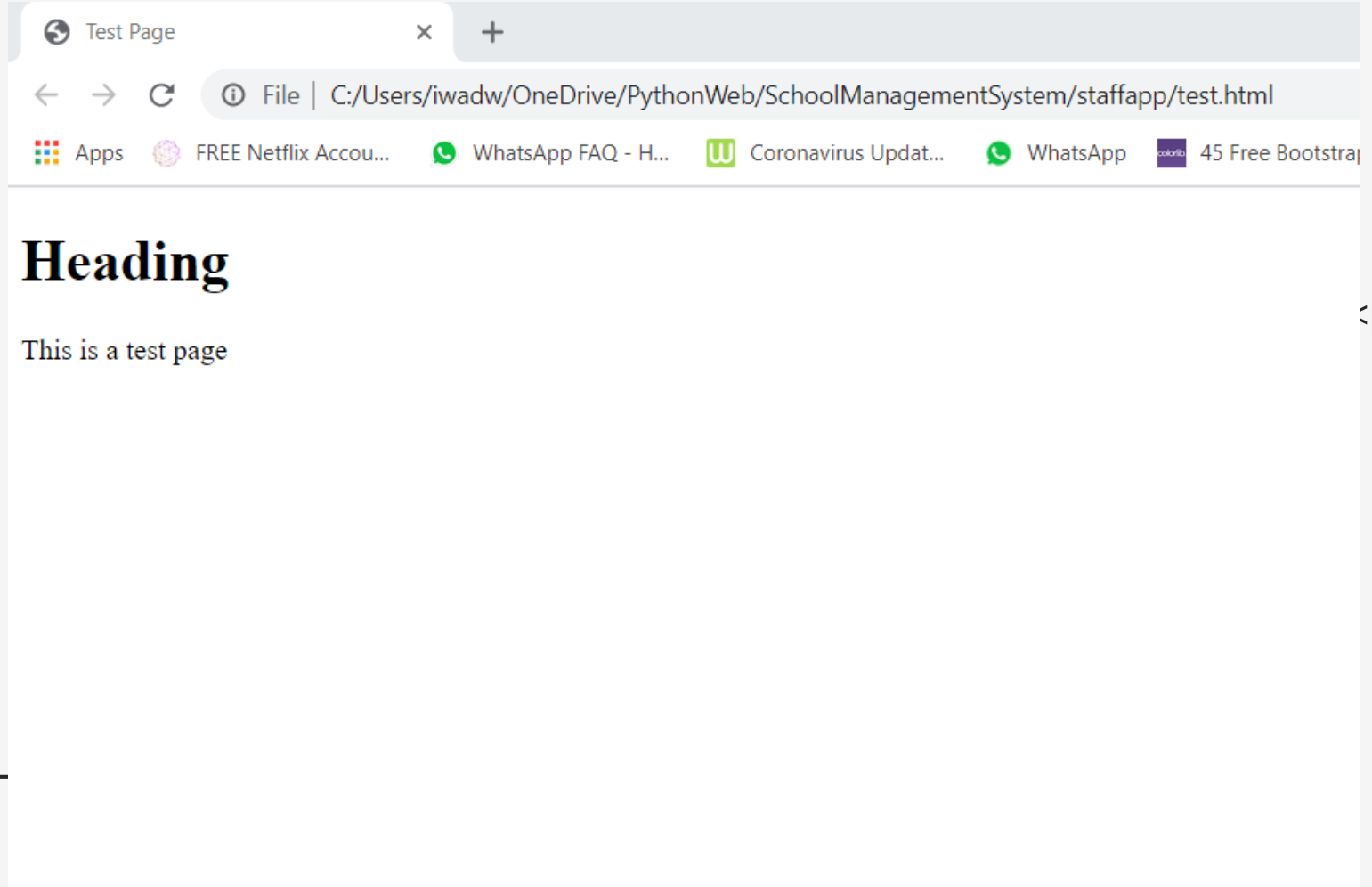
- Prior knowledge of basic python code
- How to install packages using pip
- Python interpreter (Python 3.8.2)
- An editor (Sublime Text used here)
- Prior or no knowledge in html and css
- Laptop with internet access

Getting Started

There are many frameworks for developing a website using python; the popular which are Django and Flask. In this tutorial, we will be using flask.

First we will brush through Hyper text markup language (HTML) and Cascading Style Sheet (CSS) which will be used to create static pages.

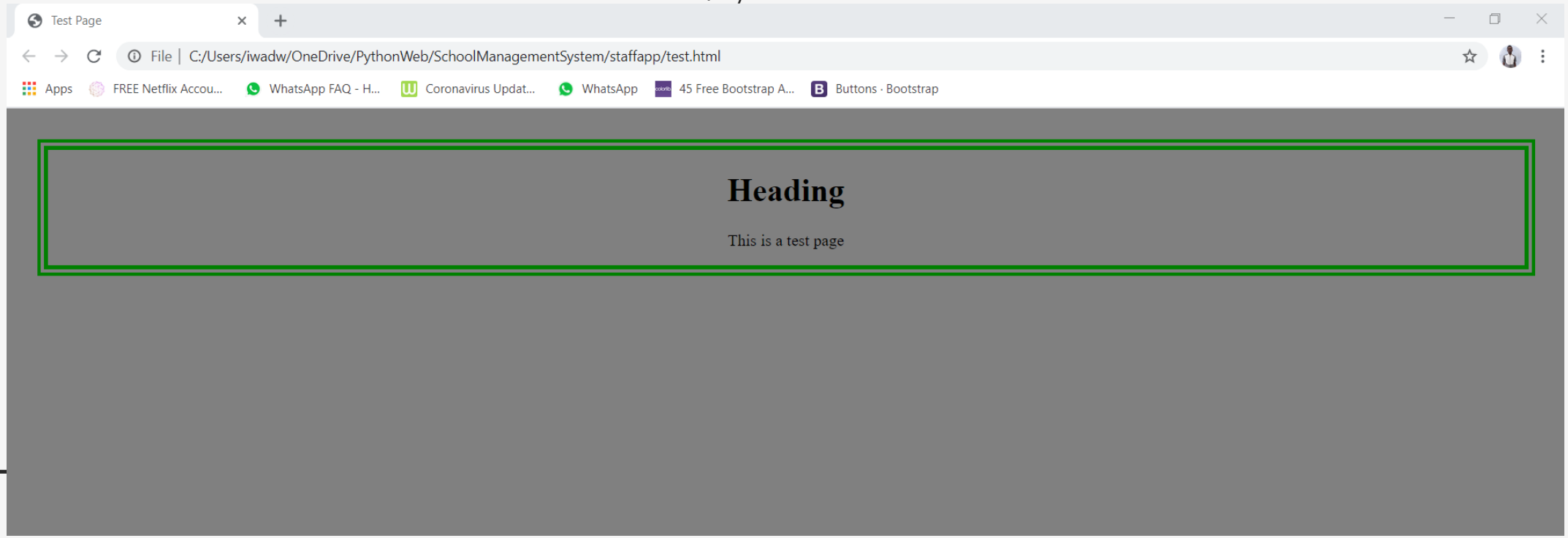
A simple HTML Page




```
<!DOCTYPE html>
<html>
<head>
<title>Test Page</title>
</head>
<body>
<h1>Heading</h1>
<p>This is a test page </p>
</body>
</html>
```

HTML with CSS

```
<body style="background-color: gray; margin: 30px;">
<style type="text/css">
.para{
border-width: 10px;
border-style: double;
border-color: green;
text-align: center;
}
</style>
```

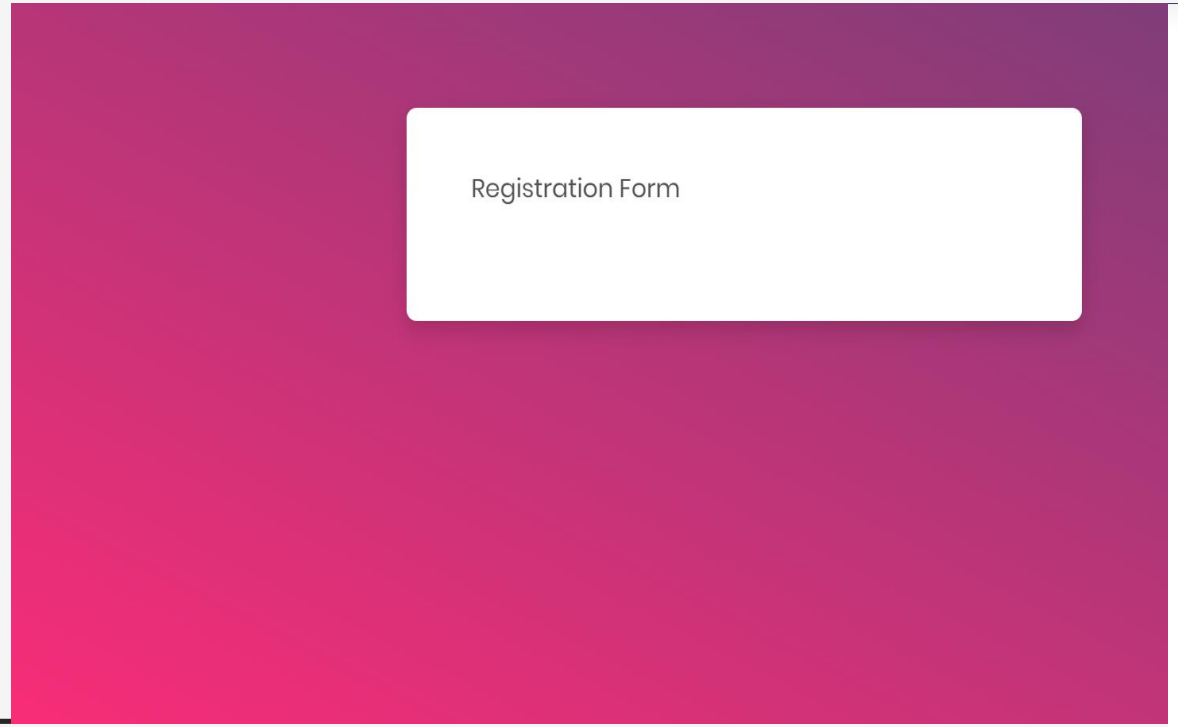


Adding Bootstrap

(https://getbootstrap.com/docs/4.4/getting-started/introduction/)

Bootstrap is a css code containing a lot of class designs used for a rich website.

It can be used offline or also served from a Content Development Network (CDN)



Starter template

Be sure to have your pages set up with the latest design and development standards. That means using an HTML5 doctype and including a viewport meta tag for proper responsive behaviors. Put it all together and your pages should look like this:

Copy

```
<!doctype html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css" integrity=

    <title>Hello, world!</title>
  </head>
  <body>
    <h1>Hello, world!</h1>

    <!-- Optional JavaScript -->
    <!-- jQuery first, then Popper.js, then Bootstrap JS -->
    <script src="https://code.jquery.com/jquery-3.4.1.slim.min.js" integrity="sha384-J6qa4849blE2+poT4WnyKhv5vZF5SrPo
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js" integrity="sha384-Q6E9RHvbIyZF
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js" integrity="sha384-wfSDF2E50Y
  </body>
</html>
```

Installing python and others

By now you should have python installed on your system else visit https://www.python.org/downloads/ to download and install.

Ensure you select Add to Path during installation.

Next, download and install sublime text (an editor) from here https://download.sublimetext.com/Sublime%20Text%20Build%203211%20x64%20Setup.exe

Next, we create a folder where the project will reside

Installing Flask

Flask is the framework that supports web development in python.

First we create a virtual environment.

A virtual environment is a container which will contain all libraries and packages used by us. This enables all requirements to be contained and can be easily sent without having to install the libraries again

On your cmd/terminal, navigate to the project folder using the 'cd {path}' command

Type:

pip install virtualenv

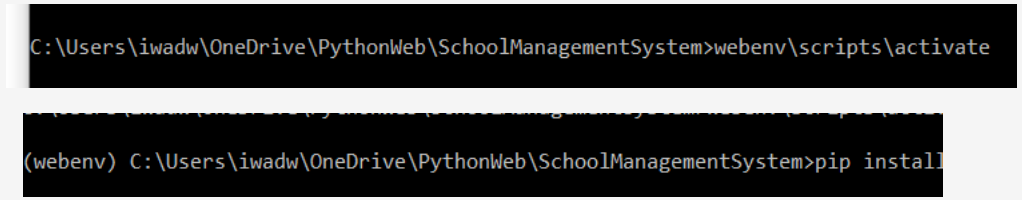
Create a virtual environment using

python –m venv virtual

Where virtual is the desired name of your environment

Installing Flask

Activate your environment by running the \scripts\activate.bat located in the virtual environment folder



After activating your virtual environment, install python using

pip install flask

Confirm by typing python on your virtual cmd, then

import flask

If no errors, then flask is installed

Building our first web app

From your project folder create your app folder, you will also see another folder which is your virtual environment.

Open sublime text, click on open folder and select your app folder.

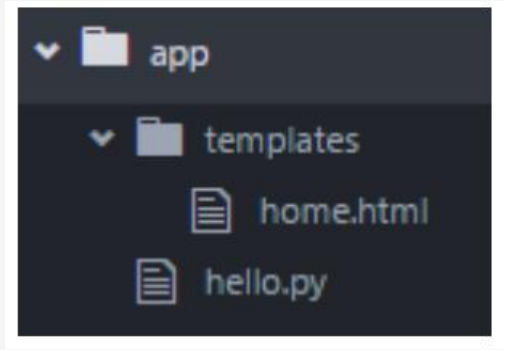
Create a new file called hello.py

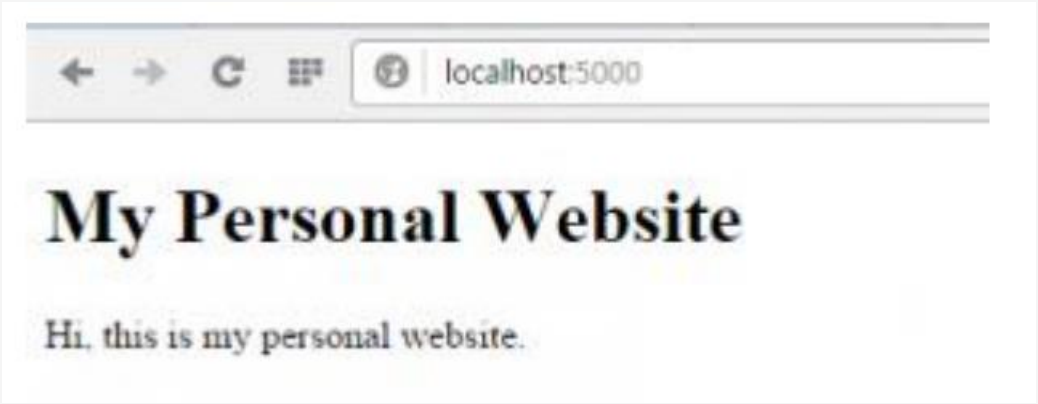
Input this

- ```
  from flask import Flask
  app = Flask(__name__)
  @app.route('/')
  def home():
  return "Hey there!"
  if __name__ == '__main__':
  app.run(debug=True)
  ```

- Each line will be explained in a voice note
- Run the hello.py file through:
  - python hello.py

Once you run the script, your website should be now live on your local machine and it can be viewed by visiting localhost:5000 in your browser.
  ```





Create an empty file, name it something like home.html and put the following HTML code inside
it:

```
<!DOCTYPE html>
<html>
<body>
<h1>My Personal Website</h1>
<p>Hi, this is my personal website.</p>
</body>
</html>
```

Save the file in a folder called templates and update your script

```
from flask import Flask, render_template
app = Flask(__name__)
@app.route('/')
def home():
return render_template('home.html')
if __name__ == '__main__':
app.run(debug=True)
```

SAVE AND REFRESH BROWSER

MAIN EVENT

Cheat: I create the page first with pure html and css then break them up into layout vs others and insert jinja2 templates

- CREATING OUR LOGIN PAGE
- We will use jinja2 templates in our html
- With this, we can break html pages by separating common design into separate file e.g loginlayout.html
- {% %} represents code blocks for keywords e.g while, for, extends, if, etc
- Block is used to insert html file into another using
- {% block content %}

  Where the other file is displayed
- {% endblock %}
- This {{ }} is used for varibles
- {{ variable_name }}
- Note all html files are created in the templates folder

```
<!DOCTYPE html>
<html lang="en">

<head>
    <!--Required meta tags-->
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

    <!--Title Page-->
    <title>{{ title }} Page</title>

    <!--Main CSS-->
    <link href="{{ url_for('static', filename='css/main.css') }}" rel="stylesheet" media="all">
        <link href="{{ url_for('static', filename='css/bootstrap.css') }}" rel="stylesheet" media="all">
</head>

<body>
    <div class="page-wrapper bg-gra-02 p-t-130 p-b-100 font-poppins">
        <div class="wrapper wrapper--w680">
            <div class="card card-4">
                <div class="card-header">{{ message }}</div>
                <div class="card-body">
                    <h2 class="title">{{ title }} Form</h2>
                    <hr>
                    <br>
                    <br>
                
                {% block content %}
                {% endblock %}
                </div>
            </div>
        </div>
    </div>

</body>
```

Create an html file called loginlayout.html

Input this

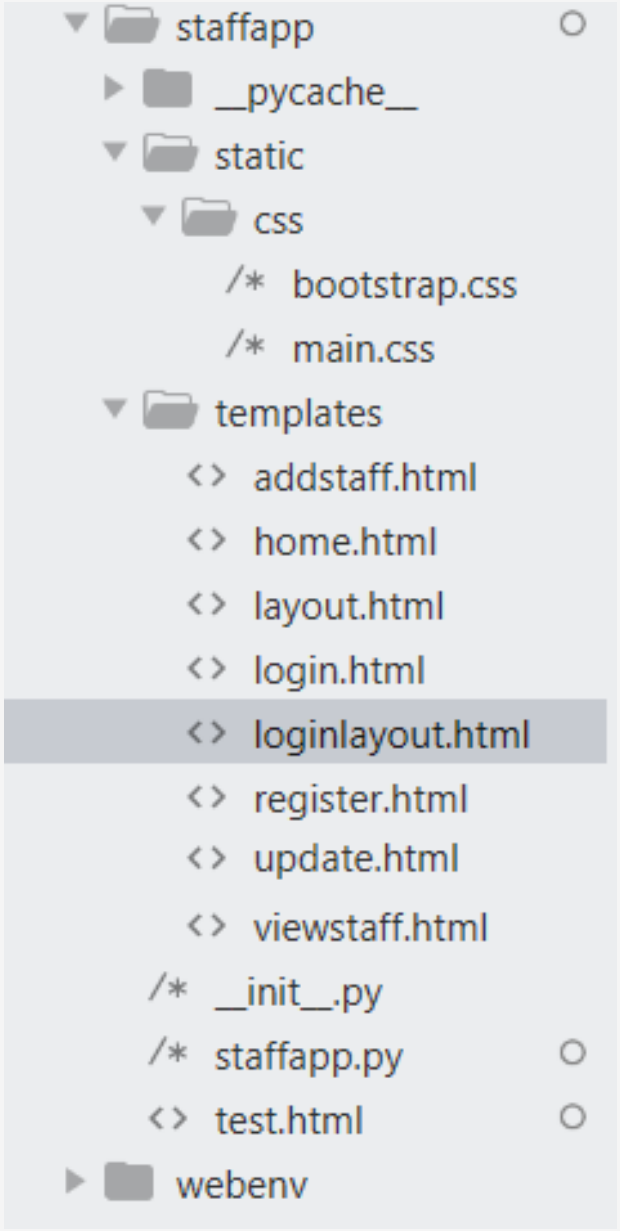
<!DOCTYPE html>

<html lang="en">

<head>

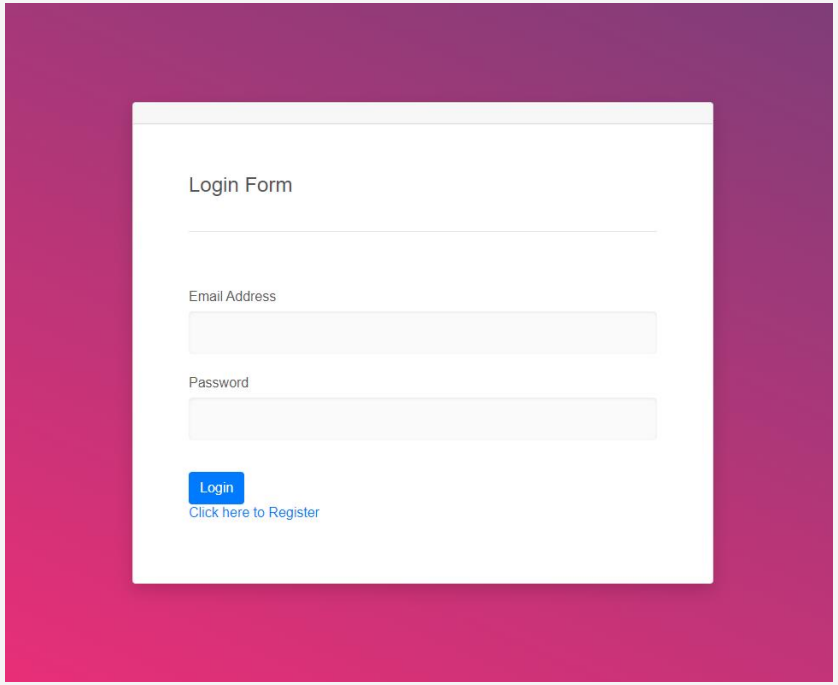
<!-- Required meta tags-->

<meta charset="UTF-8"> (complete code or file will be sent to the grp)



- Note, your folder tree should look like this
- Ignore other files for now…
- Create a folder called static
- Create a sub folder called css
- Add the two css files into it
- Bootstrap.css can be gotten online
- I will send main.css
- Create a file called the login.html
- I will send the file to the grp

Login page

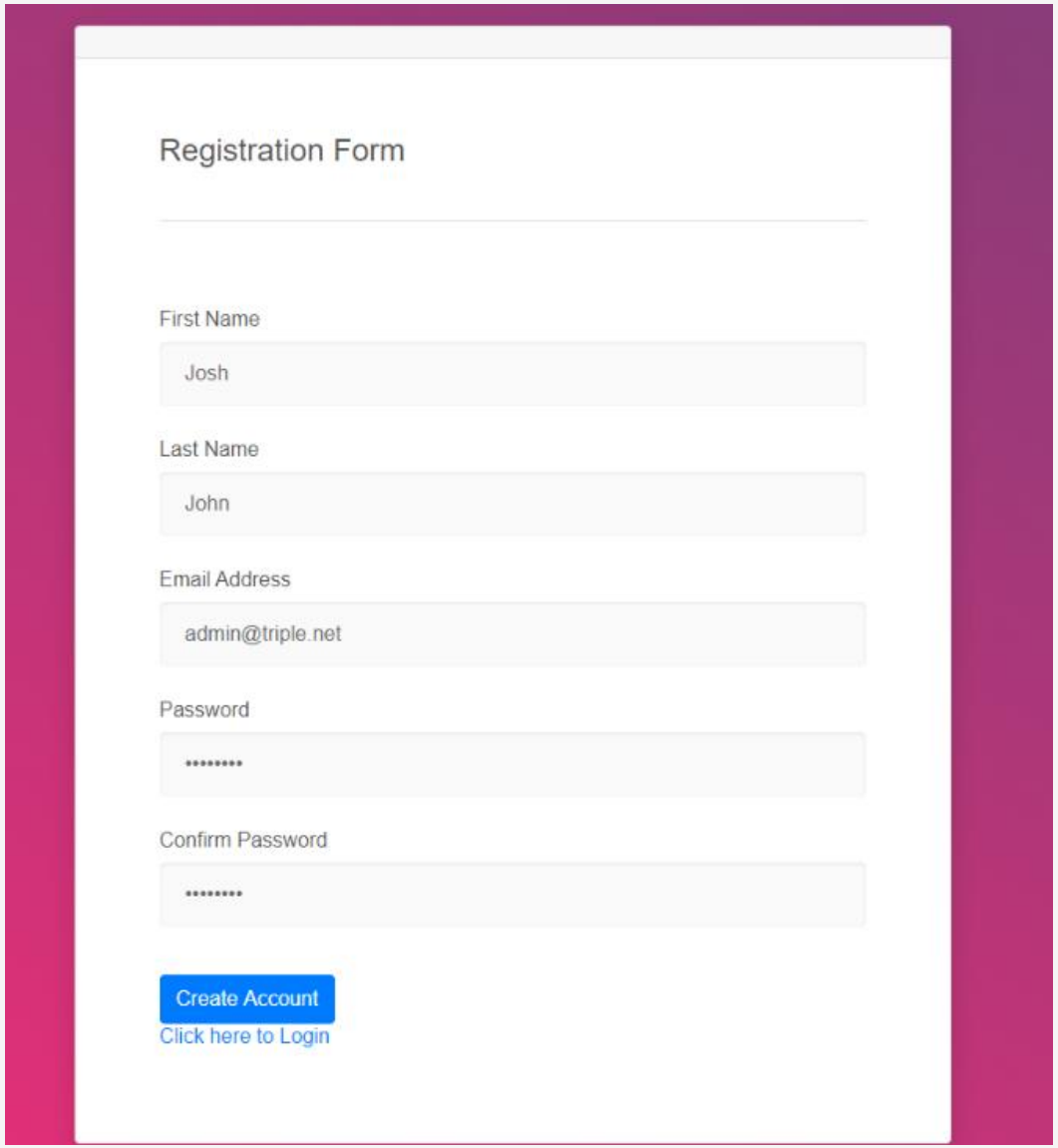


- Input this in the python script file

```
from flask import Flask, render_template, url_for
class StaticAttr:
        LoggedUser=[]
        message = ''


@app.route('/')
@app.route('/login')
def login():
        return render_template('login.html', title = 'Login',
message = StaticAttr.message)


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



Same with the register page

- Add this for the register page

```
@app.route('/register‘)
def register():
        return render_template(‘register.html', title = ‘Registration', message = StaticAttr.message)
```

Adding database

We use sqlalchemy sqlite3 db here

Install through

Pip install flask-sqlalchemy

On your app script

```
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)

app.config['SQLALCHEMY_DATABASE_URI'] =
'sqlite:///site.db'

app.config["SQLALCHEMY_TRACK_MODIFICATIONS"] =
False

db = SQLAlchemy(app)
```

Create a model

```python
class Admin(db.Model):
    id = db.Column(db.Integer, primary_key = True)
    firstName = db.Column(db.String(120), nullable=False)
    lastName = db.Column(db.String(120), nullable = False)
    email = db.Column(db.String(120), unique=True, nullable=False)
    password = db.Column(db.String(120), nullable = False)
    staffs = db.relationship('Staff', backref = 'author', lazy = True)

    def __repr__(self):
        return f"Admin ('{self.firstName}', '{self.lastName}', '{self.email}')"


class Staff(db.Model):
    id = db.Column(db.Integer, primary_key = True)
    firstName = db.Column(db.String(120), nullable=False)
    lastName = db.Column(db.String(120), nullable = False)
    email = db.Column(db.String(120), unique=True, nullable=False)
    password = db.Column(db.String(20), nullable = False)
    function = db.Column(db.Text, nullable = False)
    admin_id = db.Column(db.Integer, db.ForeignKey('admin.id'), nullable=False)

    def __repr__(self):
        return f"Student ('{self.firstName}', '{self.lastName}', '{self.email}')"
```

- We create a class admin and staff using the db model

```
class Admin(db.Model):
        id = db.Column(db.Integer, primary_key = True)
        firstName = db.Column(db.String(120), nullable=False)
        lastName = db.Column(db.String(120), nullable = False)
        email = db.Column(db.String(120), unique=True, nullable=False)
        password = db.Column(db.String(120), nullable = False)
        staffs = db.relationship('Staff', backref = 'author', lazy = True)
        def __repr__(self):
            return f"Admin ('{self.firstName}', '{self.lastName}', '{self.email}')"
class Staff(db.Model):
        id = db.Column(db.Integer, primary_key = True)
        firstName = db.Column(db.String(120), nullable=False)
        lastName = db.Column(db.String(120), nullable = False)
        email = db.Column(db.String(120), unique=True, nullable=False)
        password = db.Column(db.String(20), nullable = False)
        function = db.Column(db.Text, nullable = False)
        admin_id = db.Column(db.Integer, db.ForeignKey('admin.id'), nullable=False)

        def __repr__(self):
            return f"Student ('{self.firstName}', '{self.lastName}', '{self.email}')"
```

Hashing our password

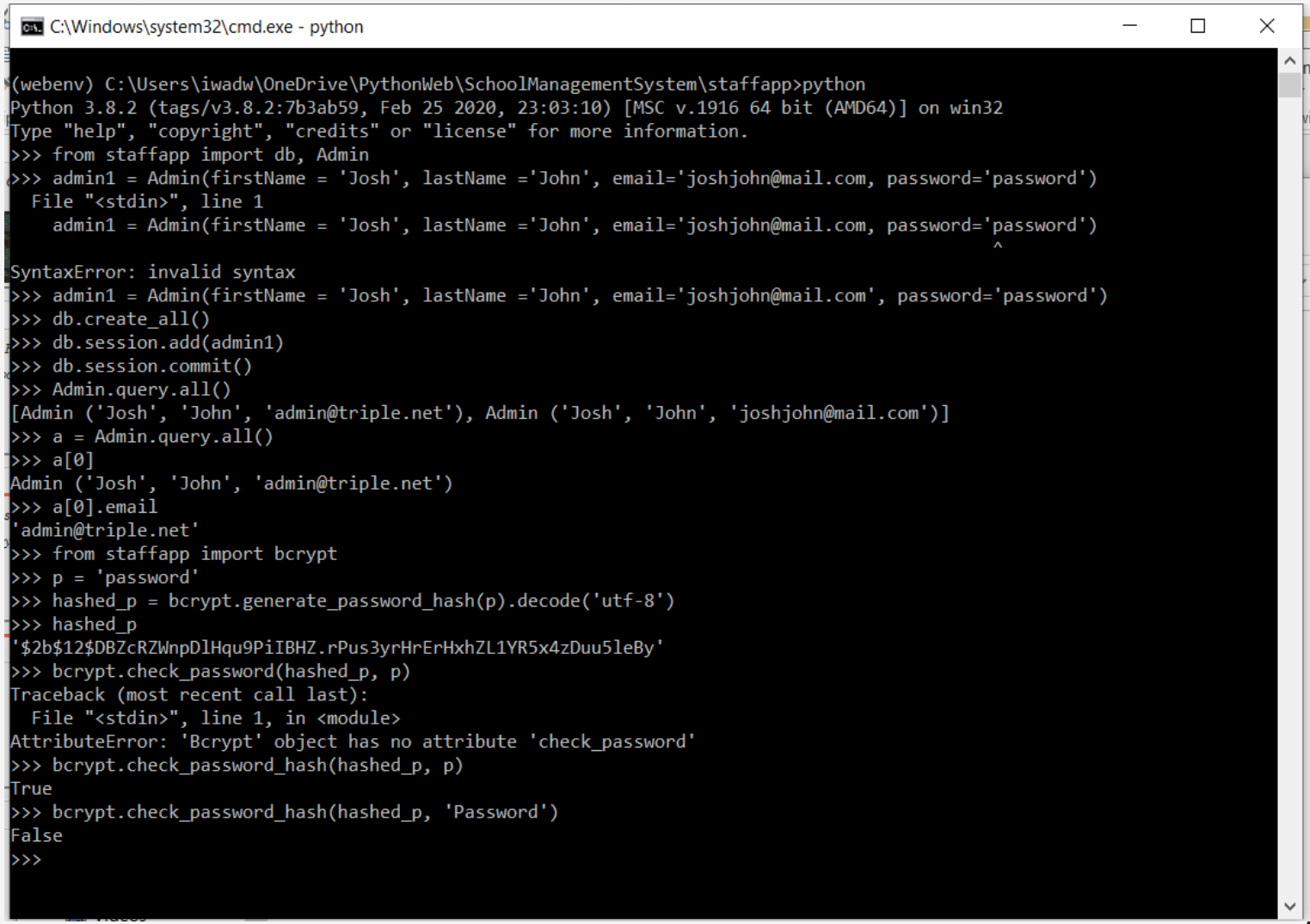
- We hash our password before storing in the database using bcrypt
- Install using
  - pip install flask-bcrypt

Update code

```
from flask import Flask, render_template, redirect, url_for, request
from flask_sqlalchemy import SQLAlchemy
from flask_bcrypt import Bcrypt


app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///site.db'
app.config["SQLALCHEMY_TRACK_MODIFICATIONS"] = False
db = SQLAlchemy(app)
bcrypt = Bcrypt(app)
```

Testing the database and bcrypt from the python console



Creating a home page

We also create a home.html file and add this to the main script

```
@app.route('/home')
def home():
        try:
                admin = StaticAttr.LoggedUser
                print(admin)
                if is_loggedin():
                        print('logged_in_admin: ', StaticAttr.LoggedUser)
                        # all_staffs = Staff.query.filter_by(author = admin)
                        staffs = Staff.query.filter_by(admin_id = admin.id)
                        print(admin, staffs)
                        StaticAttr.message=""
                        return render_template('home.html', title = 'Home',
admin = admin, staffs = staffs , message = StaticAttr.message)
                else:
                        StaticAttr.message = 'Login to continue'
        except:
                StaticAttr.message = 'An error occured'
        return redirect(url_for('login'))
```

Mumbling it all together

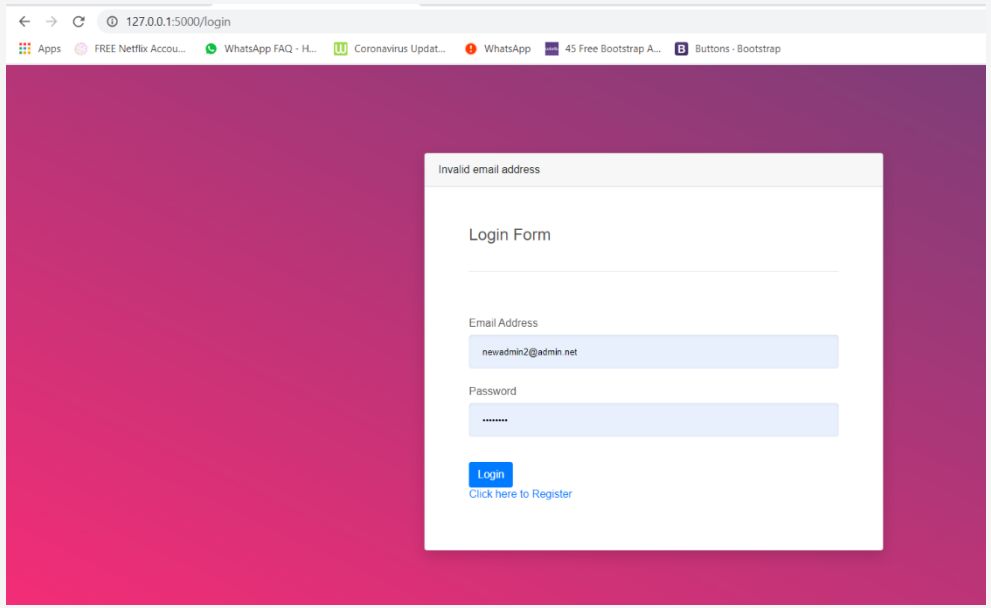
```
class StaticAttr:
        LoggedUser=[]
        message = ''
class Admin(db.Model):
        id = db.Column(db.Integer, primary_key = True)
        firstName = db.Column(db.String(120), nullable=False)
        lastName = db.Column(db.String(120), nullable = False)
        email = db.Column(db.String(120), unique=True, nullable=False)
        password = db.Column(db.String(120), nullable = False)
        staffs = db.relationship('Staff', backref = 'author', lazy = True)
        def __repr__(self):
                return f"Admin ('{self.firstName}', '{self.lastName}', '{self.email}')"
class Staff(db.Model):
        id = db.Column(db.Integer, primary_key = True)
        firstName = db.Column(db.String(120), nullable=False)
        lastName = db.Column(db.String(120), nullable = False)
        email = db.Column(db.String(120), unique=True, nullable=False)
        password = db.Column(db.String(20), nullable = False)
        function = db.Column(db.Text, nullable = False)
        admin_id = db.Column(db.Integer, db.ForeignKey('admin.id'), nullable=False)
        def __repr__(self):
                return f"Student ('{self.firstName}', '{self.lastName}', '{self.email}')"
```

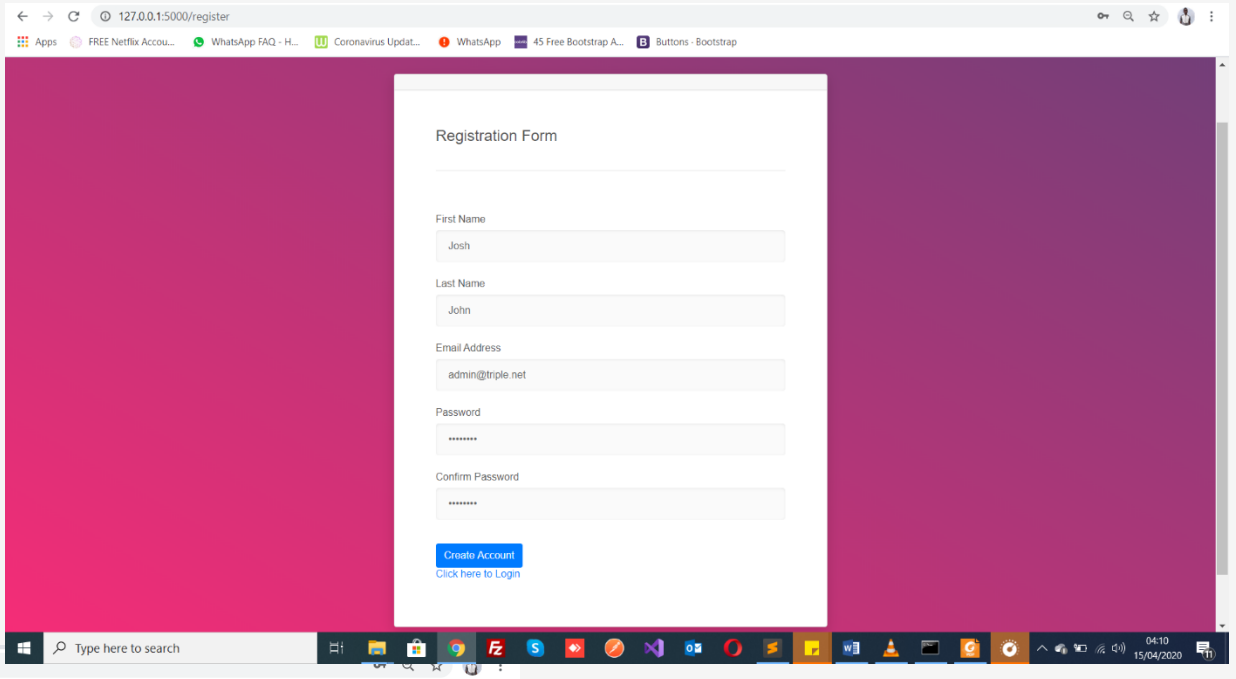
```
@app.route('/register', methods=['GET','POST'])
def register():
        try:
                if request.method == 'POST':
                        first = request.form['first_name']
                        last = request.form['last_name']
                        email = request.form['email']
                        password = request.form['password']
                        hashed_password =
bcrypt.generate_password_hash(password).decode('utf-8')
                        if email_exist(email) == False:
                                admin = Admin(firstName = first, lastName =
last, email= email, password = hashed_password)
                                db.session.add(admin)
                                db.session.commit()
                                StaticAttr.message = 'Your account has been
created!, Login to continue'
                                return redirect(url_for('login'))
                        StaticAttr.message = 'Email already exists'
        except:
                db.session.remove()
                StaticAttr.message = 'An error occured'
        return render_template('register.html', title = 'Registration', message = StaticAttr.message)
```

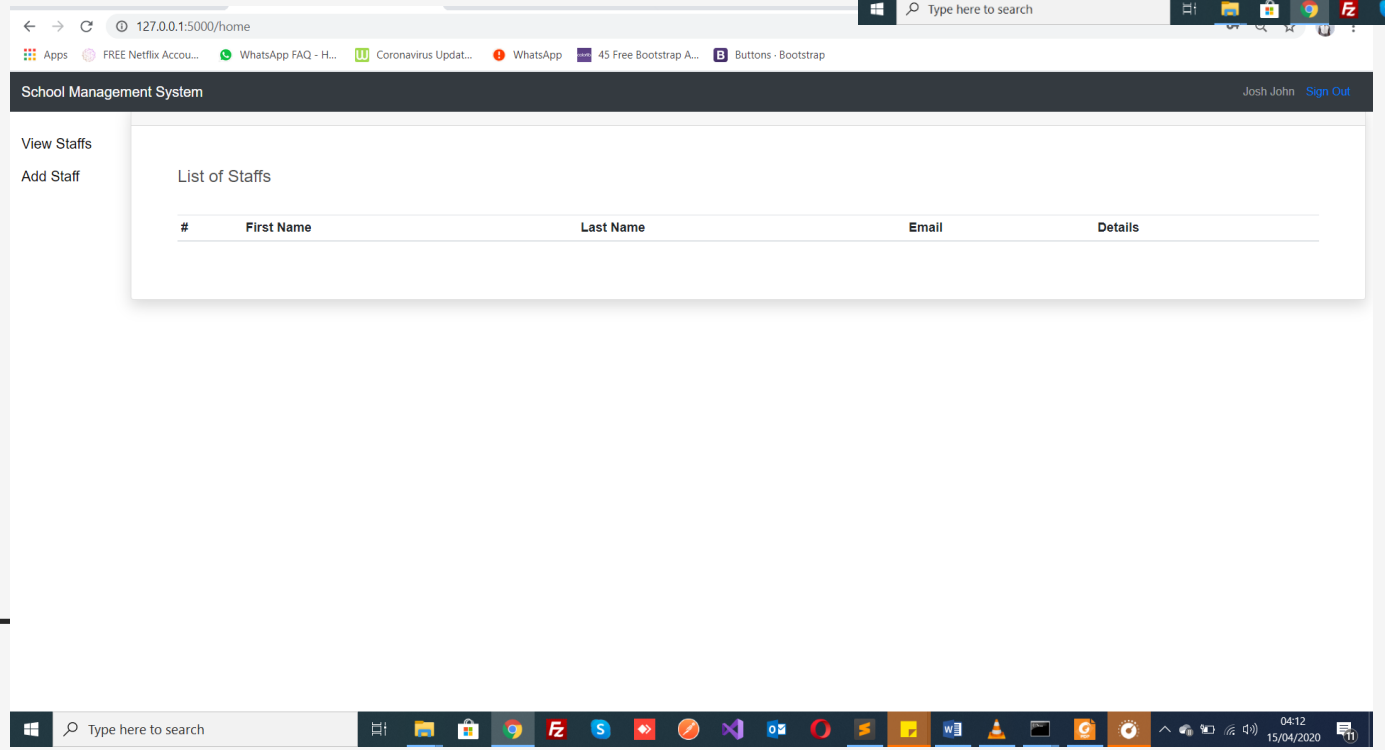
Mumbling it all together 2

```
@app.route('/')
@app.route('/login', methods=['GET','POST'])
def login():
        try:
        if request.method == 'POST':
                email = request.form['email']
                password = request.form['password']
                admin = Admin.query.filter_by(email = email).first()
                if admin:
                if bcrypt.check_password_hash(admin.password,password):
                        StaticAttr.LoggedUser = admin
                        print('login level: ', StaticAttr.LoggedUser)
                        return redirect(url_for('home'))
                        else:
                        StaticAttr.message = 'Invalid password'
                else:
                        StaticAttr.message = 'Invalid email address'
        except:
                StaticAttr.message = 'An error occured'
        return render_template('login.html', title = 'Login', message = StaticAttr.message)
```

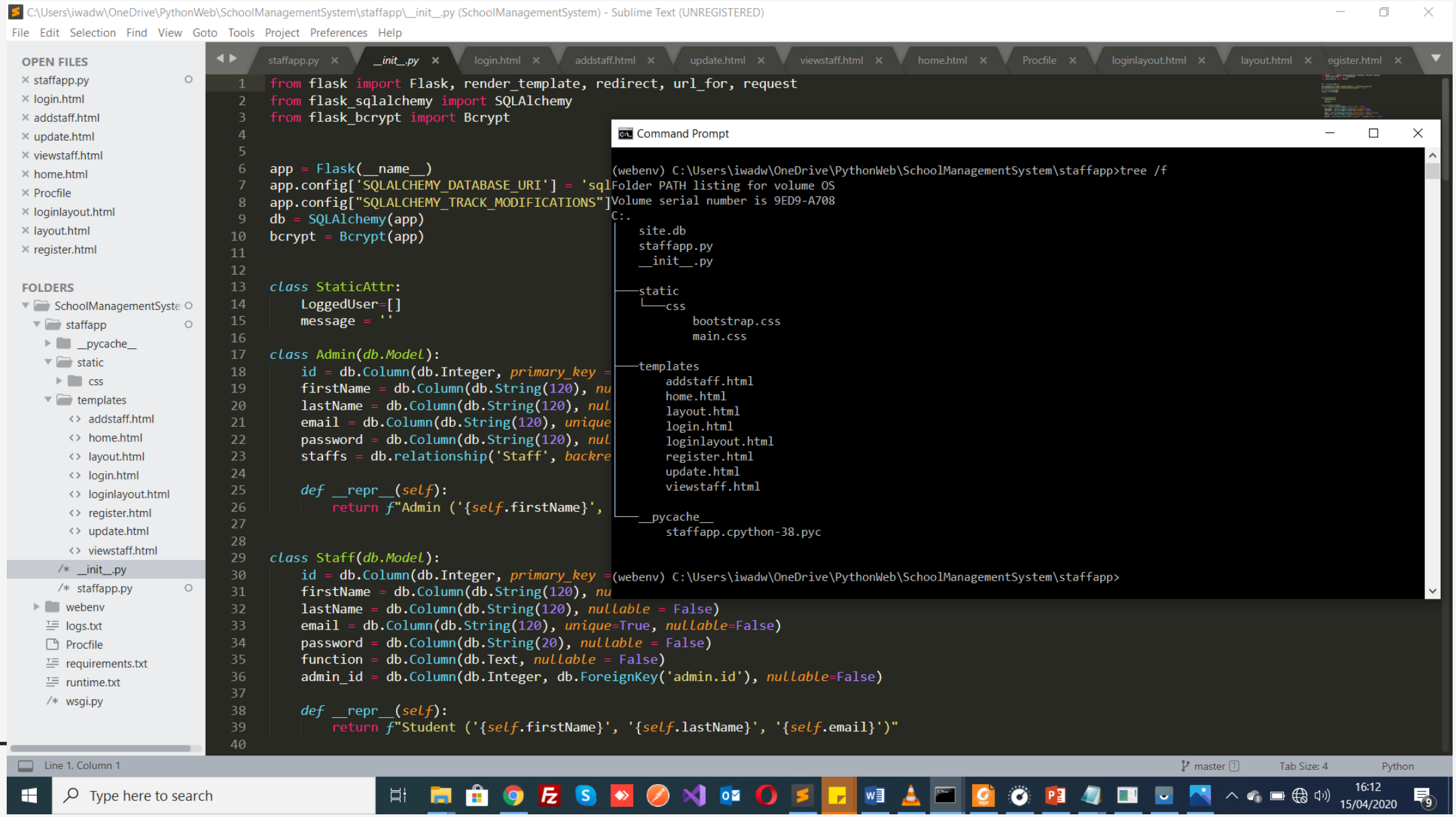
```
@app.route('/logout')
def logout():
        StaticAttr.LoggedUser=[]
        StaticAttr.message = 'Logout successful'
        return redirect(url_for('login'))
@app.route('/home')
def home():
        try:
        admin = StaticAttr.LoggedUser
        print(admin)
        if is_loggedin():
                print('logged_in_admin: ', StaticAttr.LoggedUser)
                # all_staffs = Staff.query.filter_by(author = admin)
                staffs = Staff.query.filter_by(admin_id = admin.id)
                print(admin, staffs)
                StaticAttr.message=""
                return render_template('home.html', title = 'Home', admin = admin, staffs =
staffs , message = StaticAttr.message)
        else:
                StaticAttr.message = 'Login to continue'
        except:
        StaticAttr.message = 'An error occured'
        return redirect(url_for('login'))
```


127.0.0.1:5000/login
Apps
FREE Netflix Accou...
WhatsApp FAQ - H...
Coronavirus Updat...
WhatsApp
45 Free Bootstrap A...
Buttons · Bootstrap
Invalid email address
Login Form
Email Address
newadmin2@admin.net
Password
Login
Click here to Register

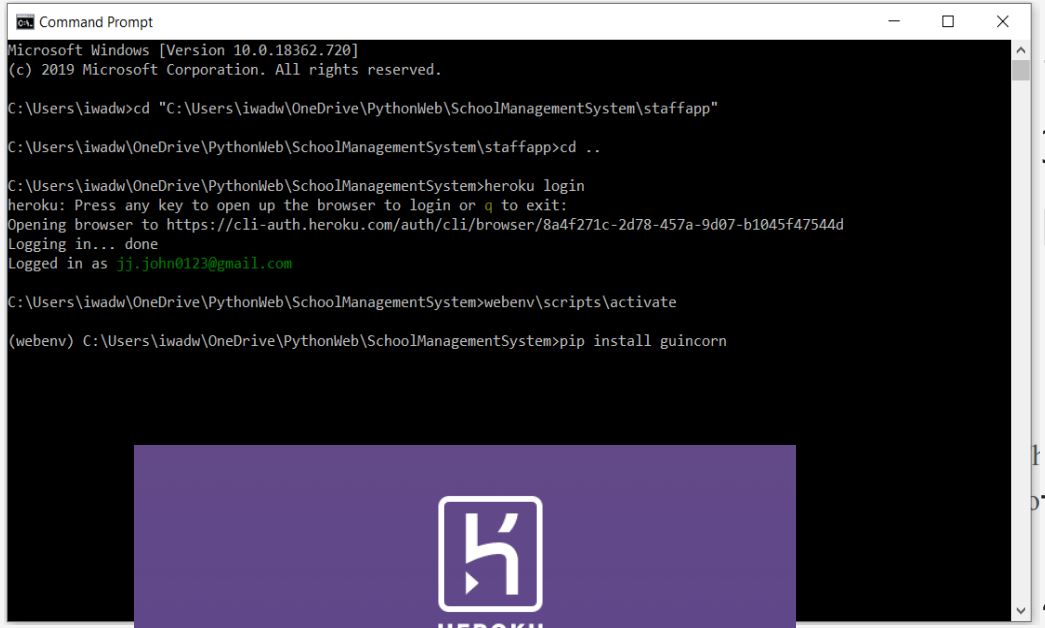

127.0.0.1:5000/register
Apps
FREE Netflix Accou...
WhatsApp FAQ - H...
Coronavirus Updat...
WhatsApp
45 Free Bootstrap A...
Buttons · Bootstrap
Registration Form
First Name
Josh
Last Name
John
Email Address
admin@triple.net
Password
Confirm Password
Create Account
Click here to Login
Type here to search
04:10
15/04/2020

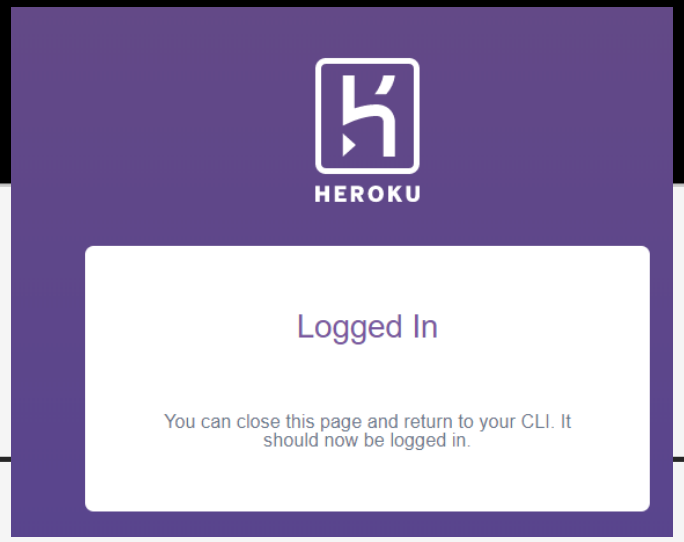

127.0.0.1:5000/home
Apps
FREE Netflix Accou...
WhatsApp FAQ - H...
Coronavirus Updat...
WhatsApp
45 Free Bootstrap A...
Buttons · Bootstrap
School Management System
Josh John
Sign Out
View Staffs
Add Staff
List of Staffs
#
First Name
Last Name
Email
Details
Type here to search
04:12
15/04/2020

Tree view of overall project



Publish online





So, let's go and publish our website online. Here are the steps:

1. We will be using the *git* tool to send our local files to the cloud. So, the first thing to do is to download git from https://git-scm.com/downloads and install it.

2. Sign up for a free account on Heroku: heroku.com.

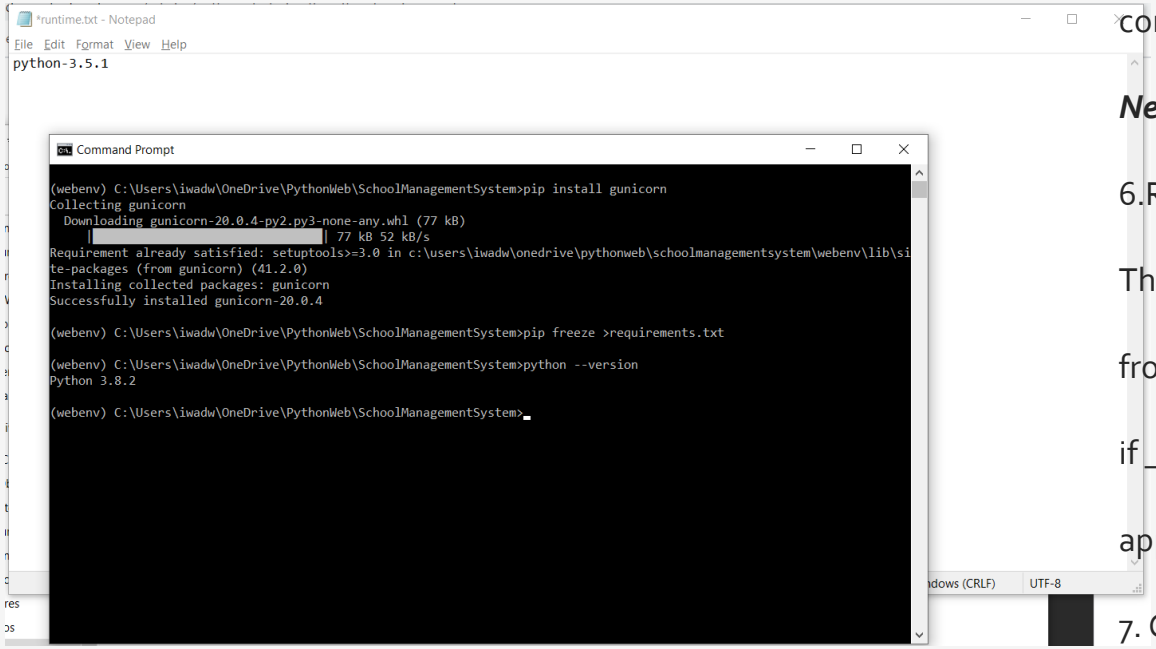
3. Download and install Heroku Toolbelt from https://toolbelt.heroku.com/.

Heroku Toolbelt is a package that allows you to interact with the Heroku cloud through your computer command line and it needs *git* to do that. You already installed *git* in step 1.

4. Start interacting with your Heroku account through the command line. Open your command line while you are inside project folder and type: ***heroku login***

you will be redirected to where you login on the browser

Publish online



5. Heroku doesn't have a webserver. Instead, it expects the application to use its own webserver.

A Python webserver to be used is ***gunicorn***. Gunicorn comes as a Python library, so you need to install it with *pip*. Gunicorn 'Green Unicorn' is a Python WSGI HTTP Server for UNIX. It's a pre-fork worker model ported from Ruby's Unicorn project. The Gunicorn server is broadly compatible with various web frameworks, simply implemented, light on server resource usage, and fairly speedy.

command line, type: ***pip install gunicorn***

Next is to CONVERT APP TO PACKAGE/MODULE

6.Rename your app.py to _ _init_ _.py,

Then in your root folder, create a file named **wsgi.py** and type this

```
from app import app

if __name__ == ("__main__"):

app.run(debug=True)
```

7. Create an empty file named ***Procfile*** in your current folder.

Then enter this line inside the empty file: **web: gunicorn wsgi:app** The file shouldn't have any extension, so make sure the file name is not getting a *.txt* extension.

Publish online

8. Create a *requirements.txt* file by typing:

pip freeze > requirements.txt

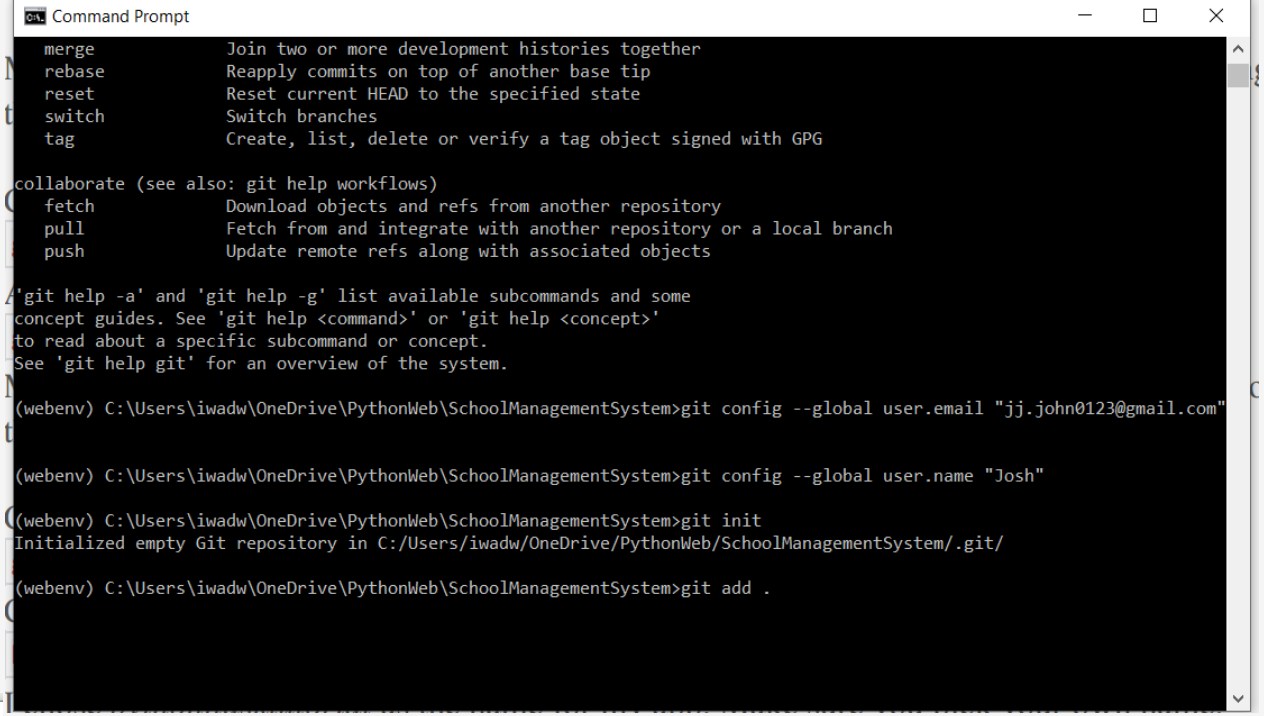
That will generate a list of the Python libraries that are installed in your virtual environment and write the list inside the *requirements.txt* file. That file will then be sent and read by the webserver so that the webserver knows what libraries to install so that application runs correctly.

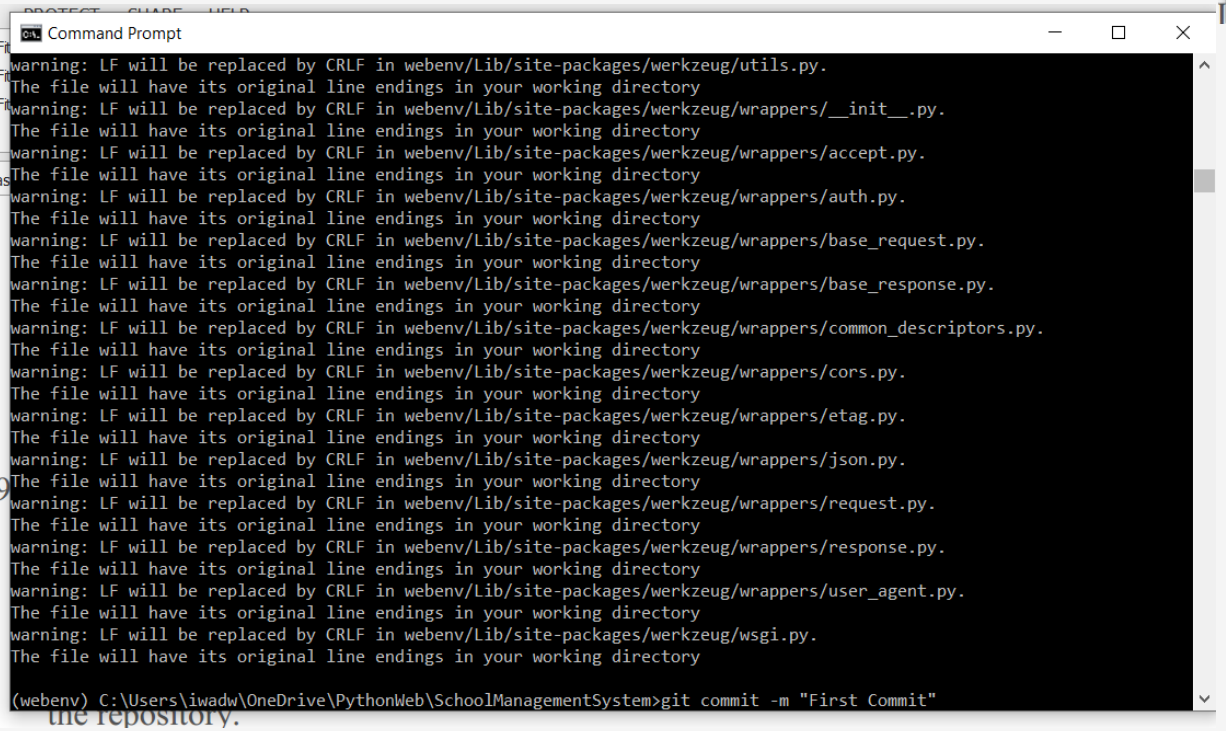
9. Great! Now, Heroku may run Python 2 by default for the applications that are sent to it.

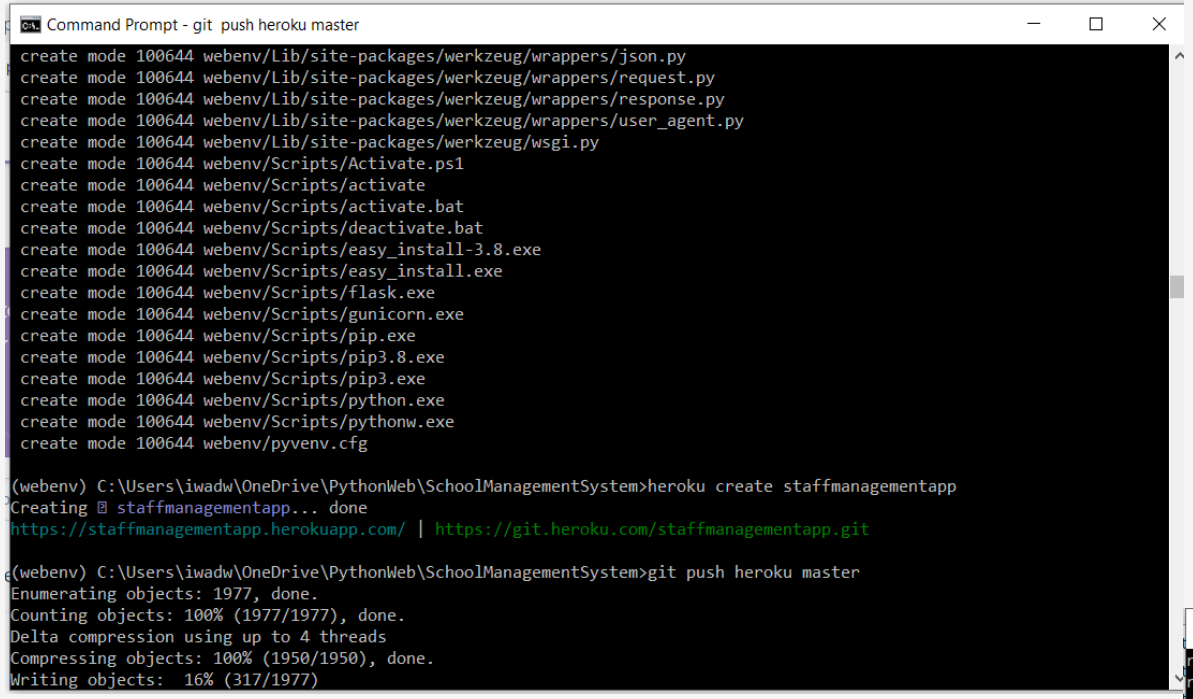
Therefore, it would be a good idea to declare what Python version your app has been designed to work with. I built my app using Python 3.8.2. To do that, you need to create a **runtime.txt** file and insert the this line in there: python-3.8.2

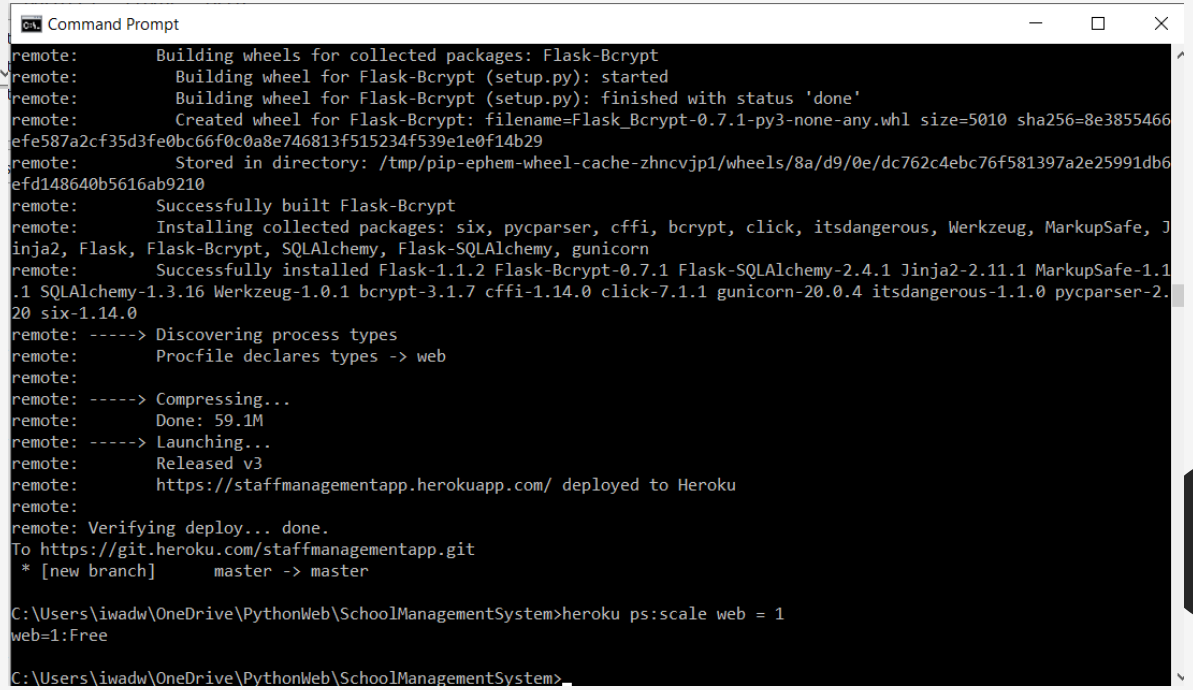
Check your python version using: python --version

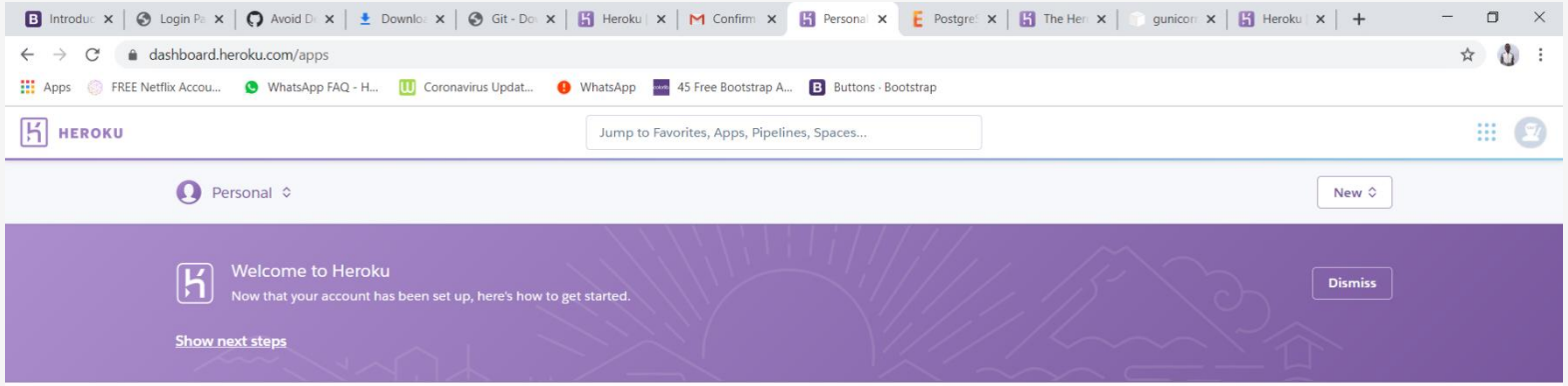
Final round

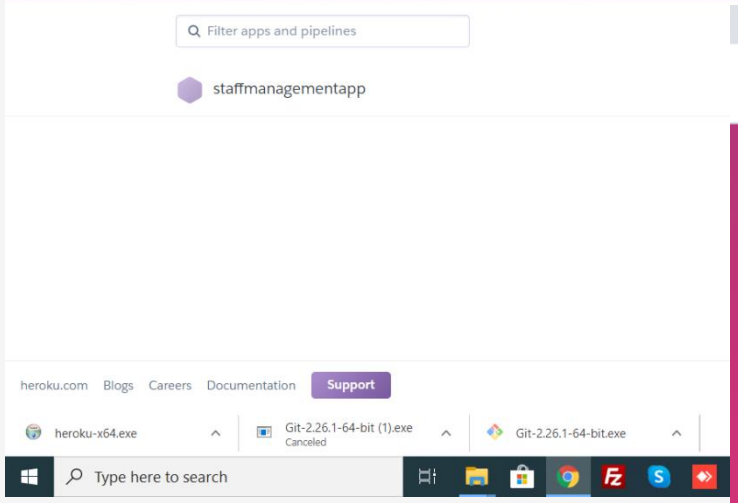

Command Prompt
merge Join two or more development histories together
rebase Reapply commits on top of another base tip
reset Reset current HEAD to the specified state
switch Switch branches
tag Create, list, delete or verify a tag object signed with GPG
collaborate (see also: git help workflows)
fetch Download objects and refs from another repository
pull Fetch from and integrate with another repository or a local branch
push Update remote refs along with associated objects
'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
See 'git help git' for an overview of the system.
(webenv) C:\Users\iwadw\OneDrive\PythonWeb\SchoolManagementSystem>git config --global user.email "jj.john0123@gmail.com"
(webenv) C:\Users\iwadw\OneDrive\PythonWeb\SchoolManagementSystem>git config --global user.name "Josh"
(webenv) C:\Users\iwadw\OneDrive\PythonWeb\SchoolManagementSystem>git init
Initialized empty Git repository in C:/Users/iwadw/OneDrive/PythonWeb/SchoolManagementSystem/.git/
(webenv) C:\Users\iwadw\OneDrive\PythonWeb\SchoolManagementSystem>git add .

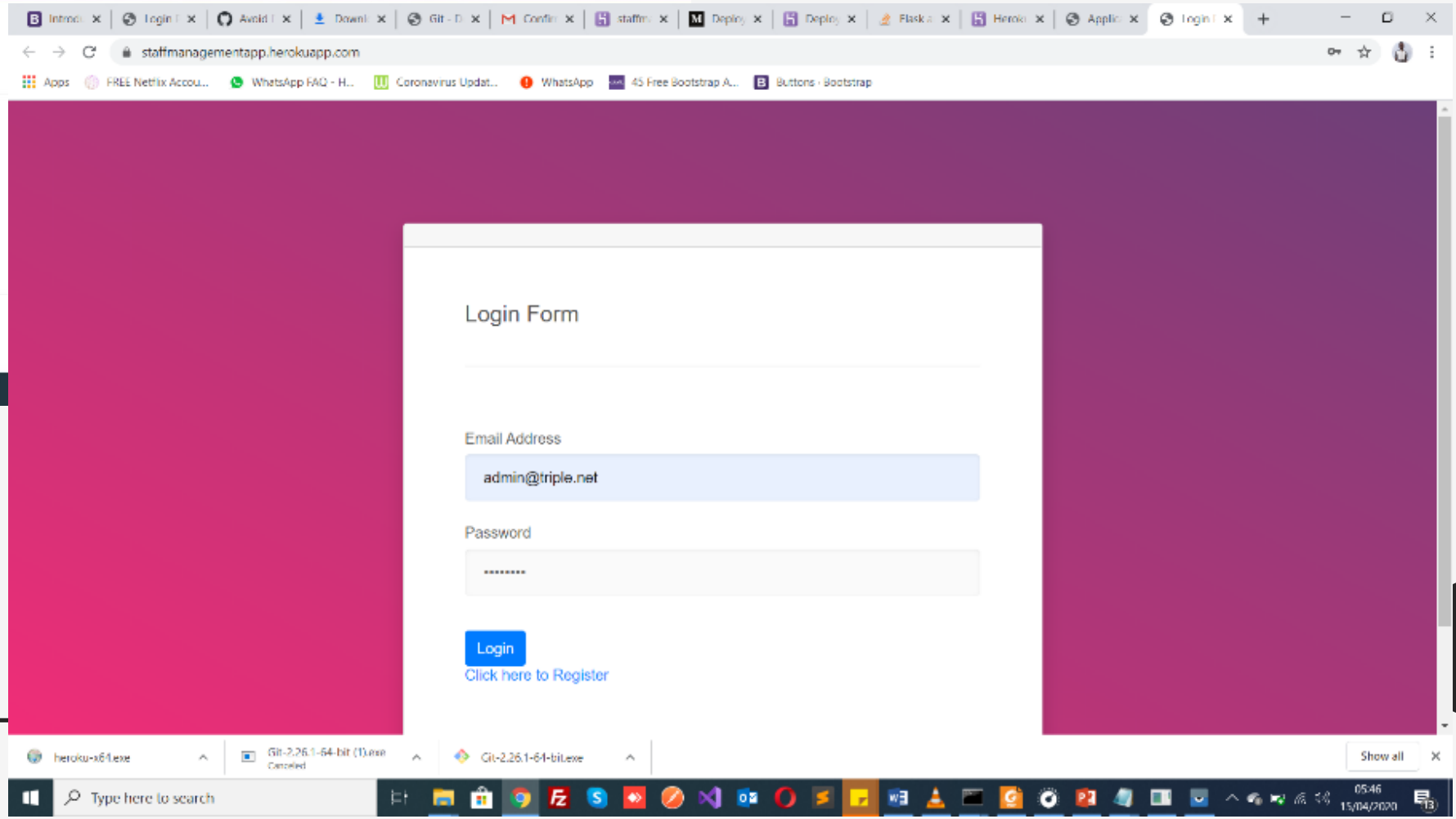

Command Prompt
warning: LF will be replaced by CRLF in webenv/Lib/site-packages/werkzeug/utils.py.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in webenv/Lib/site-packages/werkzeug/wrappers/__init__.py.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in webenv/Lib/site-packages/werkzeug/wrappers/accept.py.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in webenv/Lib/site-packages/werkzeug/wrappers/auth.py.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in webenv/Lib/site-packages/werkzeug/wrappers/base_request.py.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in webenv/Lib/site-packages/werkzeug/wrappers/base_response.py.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in webenv/Lib/site-packages/werkzeug/wrappers/common_descriptors.py.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in webenv/Lib/site-packages/werkzeug/wrappers/cors.py.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in webenv/Lib/site-packages/werkzeug/wrappers/etag.py.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in webenv/Lib/site-packages/werkzeug/wrappers/json.py.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in webenv/Lib/site-packages/werkzeug/wrappers/request.py.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in webenv/Lib/site-packages/werkzeug/wrappers/response.py.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in webenv/Lib/site-packages/werkzeug/wrappers/user_agent.py.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in webenv/Lib/site-packages/werkzeug/wsgi.py.
The file will have its original line endings in your working directory
(webenv) C:\Users\iwadw\OneDrive\PythonWeb\SchoolManagementSystem>git commit -m "First Commit"

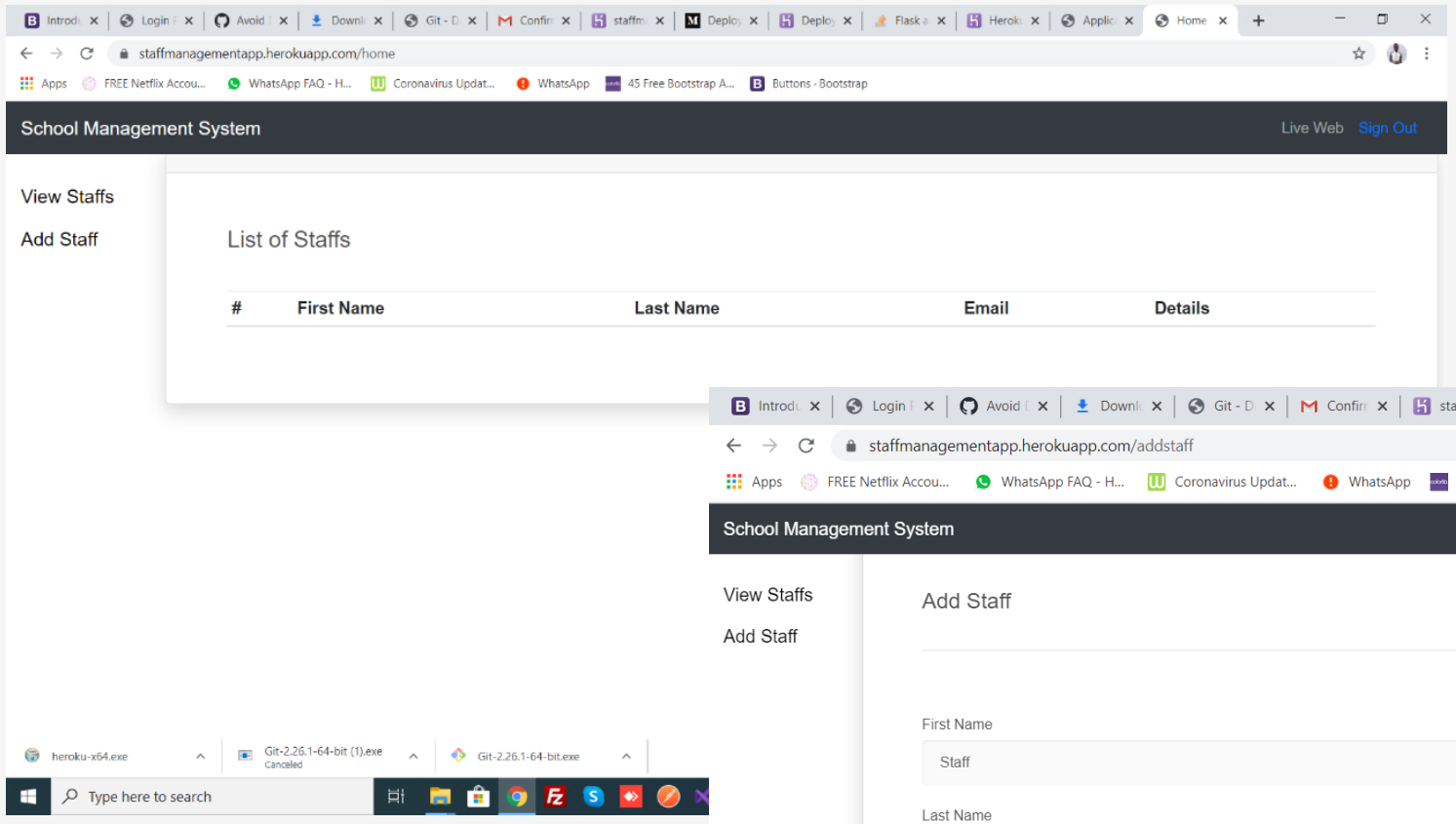

Command Prompt - git push heroku master
create mode 100644 webenv/Lib/site-packages/werkzeug/wrappers/json.py
create mode 100644 webenv/Lib/site-packages/werkzeug/wrappers/request.py
create mode 100644 webenv/Lib/site-packages/werkzeug/wrappers/response.py
create mode 100644 webenv/Lib/site-packages/werkzeug/wrappers/user_agent.py
create mode 100644 webenv/Lib/site-packages/werkzeug/wsgi.py
create mode 100644 webenv/Scripts/Activate.ps1
create mode 100644 webenv/Scripts/activate
create mode 100644 webenv/Scripts/activate.bat
create mode 100644 webenv/Scripts/deactivate.bat
create mode 100644 webenv/Scripts/easy_install-3.8.exe
create mode 100644 webenv/Scripts/easy_install.exe
create mode 100644 webenv/Scripts/flask.exe
create mode 100644 webenv/Scripts/gunicorn.exe
create mode 100644 webenv/Scripts/pip.exe
create mode 100644 webenv/Scripts/pip3.8.exe
create mode 100644 webenv/Scripts/pip3.exe
create mode 100644 webenv/Scripts/python.exe
create mode 100644 webenv/Scripts/pythonw.exe
create mode 100644 webenv/pyvenv.cfg
(webenv) C:\Users\iwadw\OneDrive\PythonWeb\SchoolManagementSystem>heroku create staffmanagementapp
Creating staffmanagementapp... done
https://staffmanagementapp.herokuapp.com/ | https://git.heroku.com/staffmanagementapp.git
(webenv) C:\Users\iwadw\OneDrive\PythonWeb\SchoolManagementSystem>git push heroku master
Enumerating objects: 1977, done.
Counting objects: 100% (1977/1977), done.
Delta compression using up to 4 threads
Compressing objects: 100% (1950/1950), done.
Writing objects: 16% (317/1977)

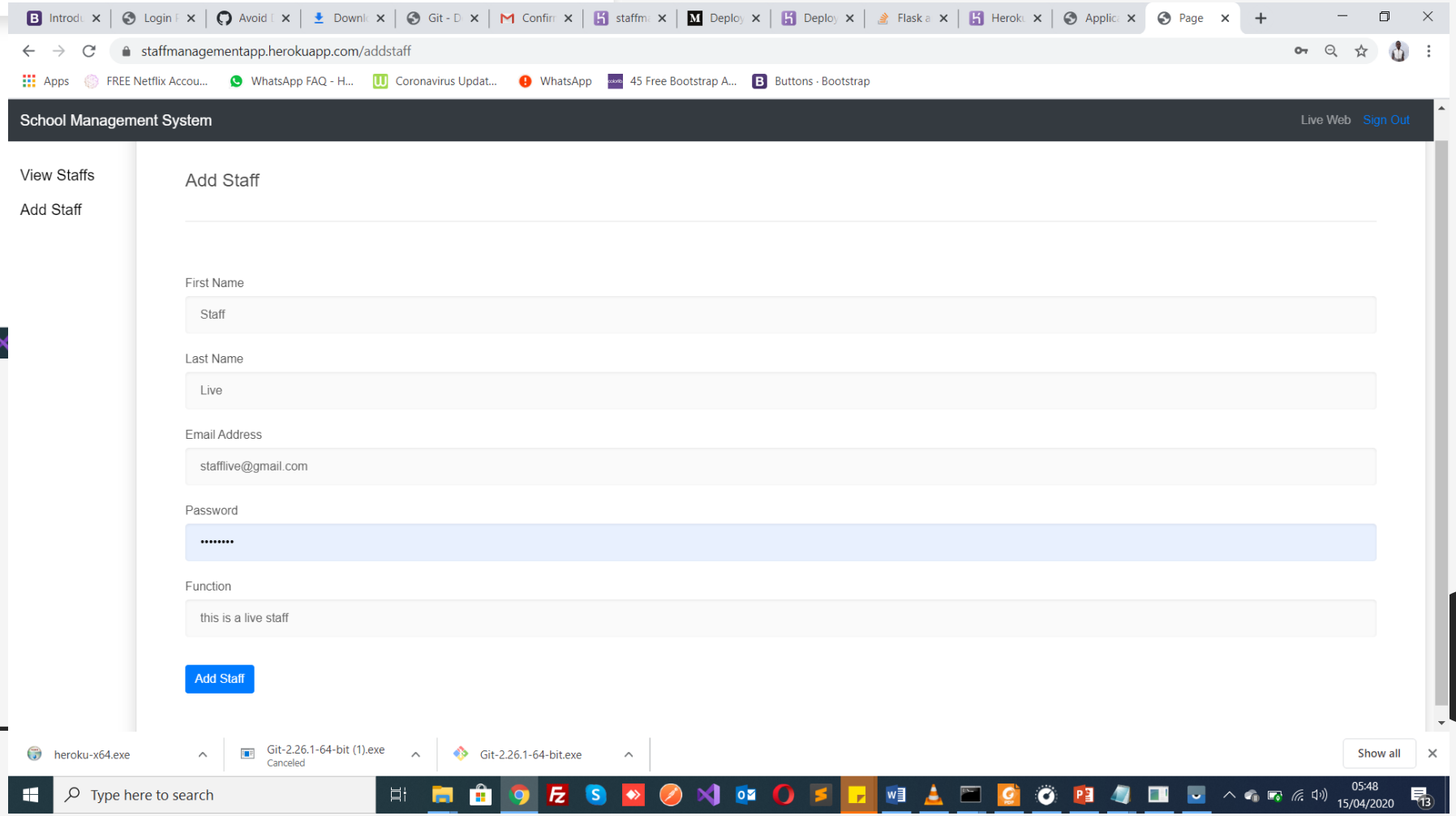

Command Prompt
remote: Building wheels for collected packages: Flask-Bcrypt
remote: Building wheel for Flask-Bcrypt (setup.py): started
remote: Building wheel for Flask-Bcrypt (setup.py): finished with status 'done'
remote: Created wheel for Flask-Bcrypt: filename=Flask_Bcrypt-0.7.1-py3-none-any.whl size=5010 sha256=8e3855466efe587a2cf35d3fe0bc66f0c0a8e746813f515234f539e1e0f14b29
remote: Stored in directory: /tmp/pip-ephem-wheel-cache-zhncvjp1/wheels/8a/d9/0e/dc762c4ebc76f581397a2e25991db6efd148640b5616ab9210
remote: Successfully built Flask-Bcrypt
remote: Installing collected packages: six, pycparser, cffi, bcrypt, click, itsdangerous, Werkzeug, MarkupSafe, Jinja2, Flask, Flask-Bcrypt, SQLAlchemy, Flask-SQLAlchemy, gunicorn
remote: Successfully installed Flask-1.1.2 Flask-Bcrypt-0.7.1 Flask-SQLAlchemy-2.4.1 Jinja2-2.11.1 MarkupSafe-1.1.1 SQLAlchemy-1.3.16 Werkzeug-1.0.1 bcrypt-3.1.7 cffi-1.14.0 click-7.1.1 gunicorn-20.0.4 itsdangerous-1.1.0 pycparser-2.20 six-1.14.0
remote: -----> Discovering process types
remote: Procfile declares types -> web
remote:
remote: -----> Compressing...
remote: Done: 59.1M
remote: -----> Launching...
remote: Released v3
remote: https://staffmanagementapp.herokuapp.com/ deployed to Heroku
remote:
remote: Verifying deploy... done.
To https://git.heroku.com/staffmanagementapp.git
 * [new branch]   master -> master
C:\Users\iwadw\OneDrive\PythonWeb\SchoolManagementSystem>heroku ps:scale web = 1
web=1:Free
C:\Users\iwadw\OneDrive\PythonWeb\SchoolManagementSystem>

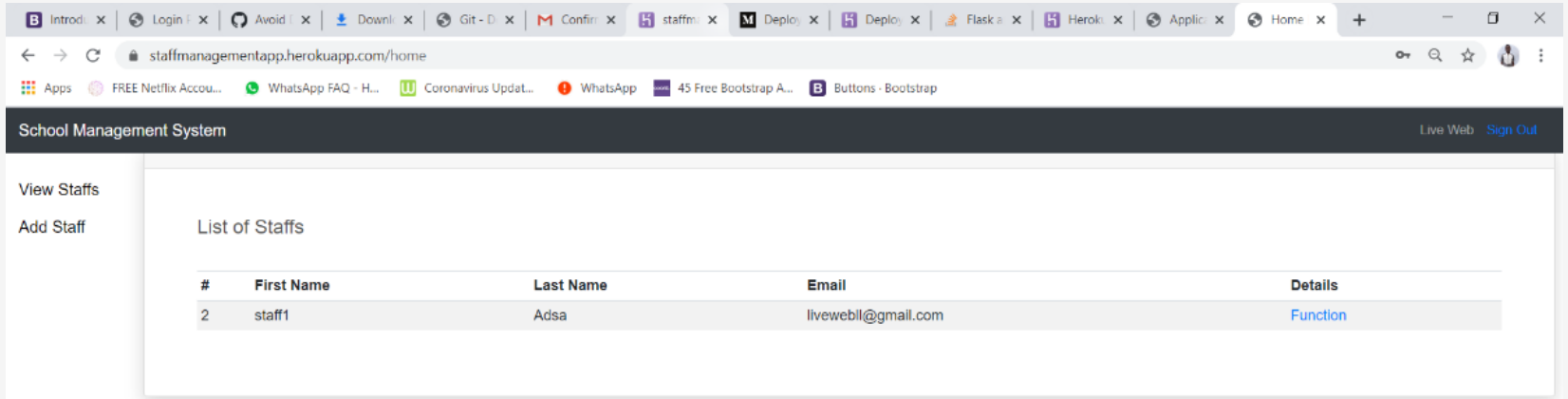

dashboard.heroku.com/apps
HEROKU
Jump to Favorites, Apps, Pipelines, Spaces...
Personal
New
Welcome to Heroku
Now that your account has been set up, here's how to get started.
Dismiss
Show next steps

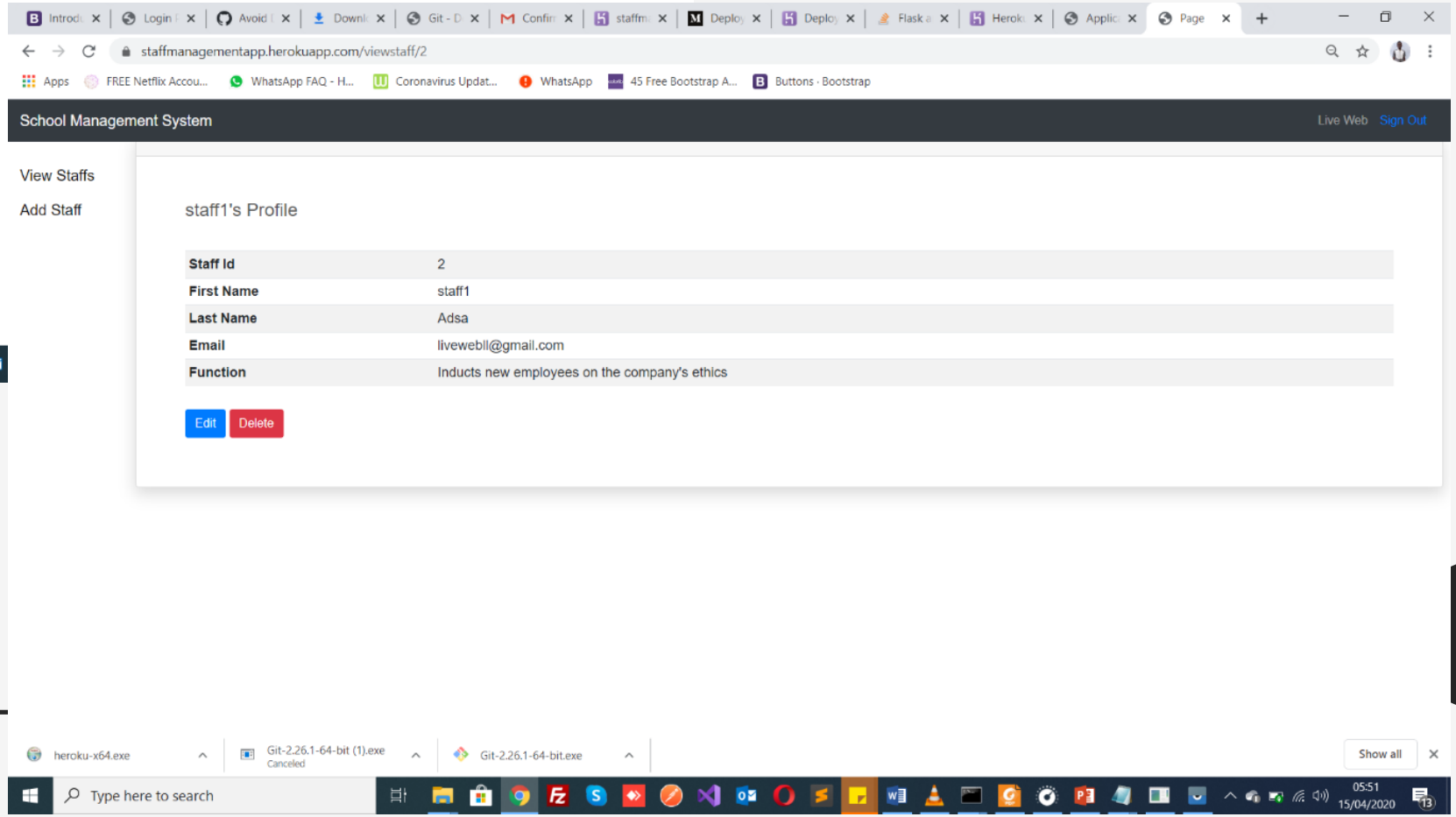

Filter apps and pipelines
staffmanagementapp
heroku.com
Blogs
Careers
Documentation
Support
heroku-x64.exe
Git-2.26.1-64-bit (1).exe
Canceled
Git-2.26.1-64-bit.exe
Type here to search


staffmanagementapp.herokuapp.com
Login Form
Email Address
admin@triple.net
Password
Login
Click here to Register
heroku-x64.exe
Git-2.26.1-64-bit (1).exe
Canceled
Git-2.26.1-64-bit.exe
Show all
Type here to search
05:46
15/04/2020


staffmanagementapp.herokuapp.com/home
School Management System
Live Web
Sign Out
View Staffs
Add Staff
List of Staffs
First Name Last Name Email Details


staffmanagementapp.herokuapp.com/addstaff
School Management System
Live Web
Sign Out
View Staffs
Add Staff
Add Staff
First Name
Staff
Last Name
Live
Email Address
stafflive@gmail.com
Password
Function
this is a live staff
Add Staff
heroku-x64.exe
Git-2.26.1-64-bit (1).exe
Canceled
Git-2.26.1-64-bit.exe
Show all
Type here to search
05:48
15/04/2020


staffmanagementapp.herokuapp.com/home
School Management System
Live Web
Sign Out
View Staffs
Add Staff
List of Staffs
First Name Last Name Email Details
2 staff1 Adsa livewebll@gmail.com Function


staffmanagementapp.herokuapp.com/viewstaff/2
School Management System
Live Web
Sign Out
View Staffs
Add Staff
staff1's Profile
Staff Id 2
First Name staff1
Last Name Adsa
Email livewebll@gmail.com
Function Inducts new employees on the company's ethics
Edit
Delete

Note: project includes pages more than what what explained here

For production scale; Postgrel db should be used and debug should be false

Crashes was avoided in the project so as to used the sqlite3 db

For the LoggedUser, a Static property is used here.. For optimality, we use login, authentication and session managing modules.

Others are freestlyes

Thank you

Project files and presentation is access via GITHUB

https://github.com/successfuljosh/pythontutorial