INSTALLATION MANUAL

THE CLOUD BASED ONLINE COMPILER

About: What is Online Compiler using Cloud Computing?

Online Compiler is a web application where in today's fast and competitive world everything is available on the internet, and on the web. So, we developed an online compiler using cloud computing. The main objective of this project is to develop a centralized compiler that helps to reduce problems like portability storage, cost, and space. It is the most convenient tool to compile code, remove errors and debug code. Moreover, we can run the web-based application remotely from any network connection that is independent of the platform. The challenge of installing a compiler on each machine is also avoided and one can perform online exams therefore, all these benefits make this application suitable for cloud based online compiler.

Dependencies

To run this web application requires some pre-requisites environments be installed on the user's computer:

- **Python 3.8.8 - 3.9.0**
  - **It requires latest version of python to work with the application**

Installation on MacOS

To run our application, please follow below installation steps:

Before you start installing python you need to have **Homebrew** installed on your local machine. If you don't have it installed on your local machine then please follow the below steps to install it.

Step 1: Install Homebrew

To install Homebrew, open your terminal and run this command:

```
$ /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install.sh)"
```

The script will explain what changes it will make and prompt you before the installation begins. Once you've installed Homebrew, insert the Homebrew directory at the top of your **PATH** environment variable. You can do this by adding the following line at the bottom of your ~/.profile file

```
export PATH="/usr/local/opt/python/libexec/bin:$PATH"
```

If you have OS X 10.12 (Sierra) or older use this line instead

```
export PATH=/usr/local/bin:/usr/local/sbin:$PATH
```

Step 2: Now, Install Python 3 (Mac OS X comes with Python 2.7 out of the box)

```
$ brew install python
```

Pip: you can check by typing "pip" command on terminal

Homebrew installs pip pointing to the Homebrew'd Python 3 for you. At this point, you have the system Python 2.7 available, potentially the Homebrew version of Python 2 installed, and the Homebrew version of Python 3 as well. If the Homebrew version of Python 2 is installed then pip2 will point to Python 2. If the Homebrew version of Python 3 is installed then pip will point to Python 3.

Step 3: Install Flask

Flask is one of the most popular web application frameworks written in Python. It is a microframework designed for an easy and quick start. Extending with tools and libraries adds more functionality to Flask for more complex projects.

Step 1: Install Virtual Environment

1. Install Flask in a virtual environment to avoid problems with conflicting libraries.

- Python 3 comes with a virtual environment module named *venv* preinstalled. **If you have Python 3 installed, skip to Step 2.**
- Python 2 users must install the *virtualenv* module. **If you have Python 2, follow the instructions outlined in Step 1.**

2. Install virtualenv using pip:

```
sudo python2 -m pip install virtualenv
```

Step 2: Create an Environment

1. Make a separate directory for your project

```
mkdir <project name>
```

2. Move into the directory

```
cd <project name>
```

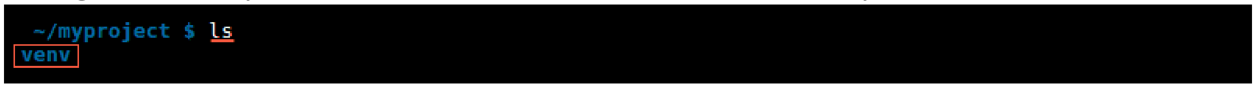
3. Within the directory, create the virtual environment for Flask. When you create the environment, a new folder appears in your project directory with the environment's name.

4. Create an Environment in Linux and MacOS

For Python 3:
To create a virtual environment for Python 3, use the *venv* module and give it a name:

```
Python3 -m venv <name of environment>
```

Listing the directory structure with the ls command shows the newly created environment:

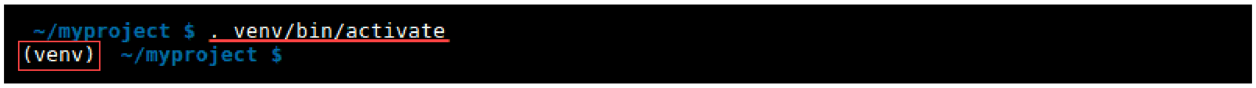


```
~/myproject $ ls
venv
```

Step 3: Activate the environment

Activate the virtual environment before installing Flask. The name of the activated environment shows up in the CLI after activation.

. <name of environment >/bin/activate



```
~/myproject $ . venv/bin/activate
(venv)  ~/myproject $
```

Step 4: Install Flask

Install Flask within the activated environment using pip:

Pip install Flask

Step 5: Test the Development Environment

1. Create a simple Flask application to test the newly created development environment.
2. Make a file in the Flask project folder called *hello.py*.
3. Edit the file using a text editor and add the following code to make an application that prints "*Hello world!*":

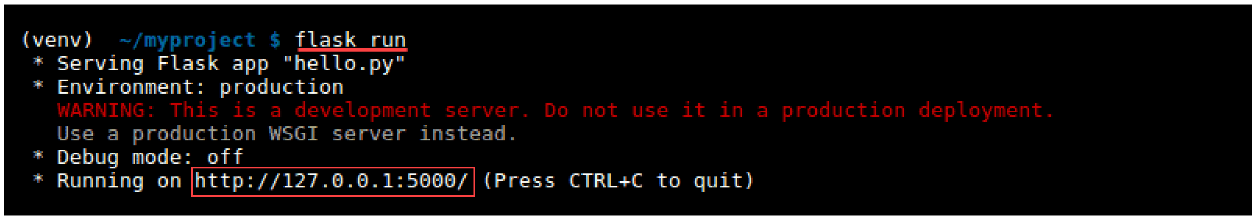
```
from flask import Flask
app = Flask(__name__)
@app.route('/')
def hello_world():
    return 'Hello world!'
```

4. Save the file and close.
5. Using the console, navigate to the project folder using the cd command.
6. Set the *FLASK_APP* environment variable.

export FLASK_APP = hello.py

7. Run the Flask Application with :

flask run



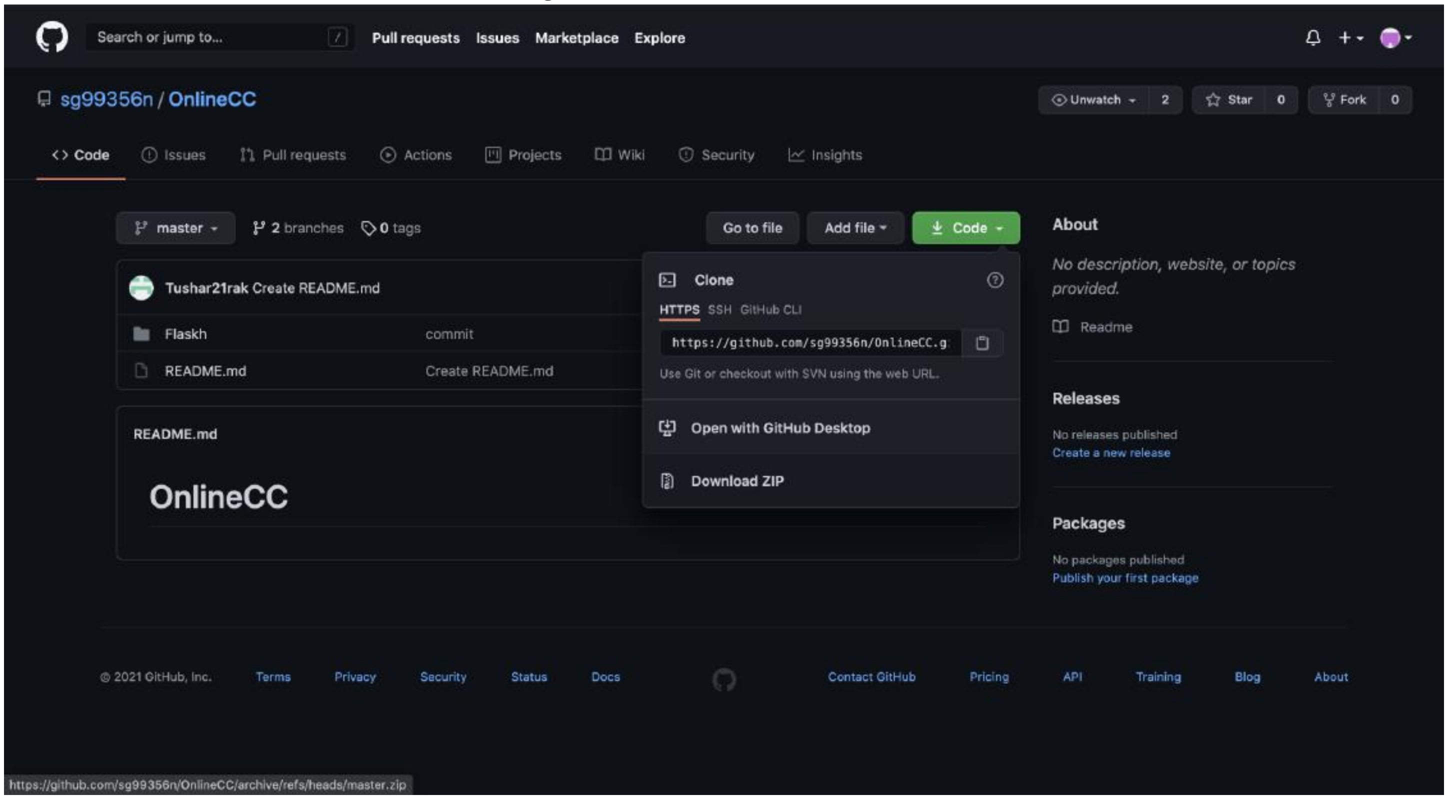
```
(venv)  ~/myproject $ flask run
 * Serving Flask app "hello.py"
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

8. Copy the address in the browser to see the project running.

Step 4: Download ONCC files from GitHub
You will need to have a copy of the application's files on your local machine in order to run the compiler. This can be achieved by going to GitHub repository at:

https://github.com/sg99356n/OnlineCC.git

and now select “Download ZIP” from the green download button above the list of files, as shown below.



```
https://github.com/sg99356n/OnlineCC.g
```

Use Git or checkout with SVN using the web URL.

No description, website, or topics provided.

No releases published
Create a new release

No packages published
Publish your first package

https://github.com/sg99356n/OnlineCC/archive/refs/heads/master.zip

Now unzip the folder and now all the files are available on your system.

Step 5: Now install all requirements for requirements.txt file

Navigate to the location where you saved your virtual environment, enable it, and then return to and enter the folder containing all of the compiler files, where you entered the command.

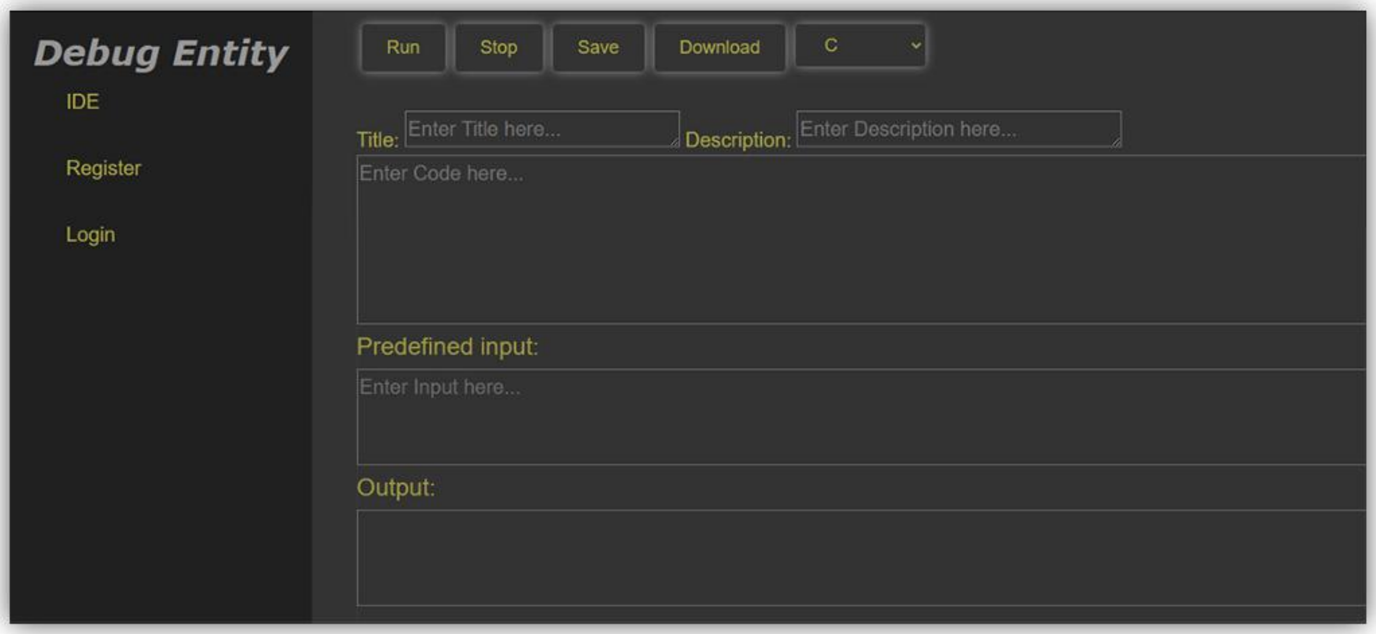
```
$ python3 -m pip install -r requirements.txt
```

Now all the dependencies required to run your application are installed.

Step 7: Running ONCC via localhost
Ensuring that you are still within the root folder containing all of ONCC’s program files, enter the following command into your terminal:

```
(pyvenv) $ python3 manage.py runserver
```

Then navigate to http://localhost:5000 in your browser, and you should see the welcome page of ONCC as shown on the next page:



Installation on Windows

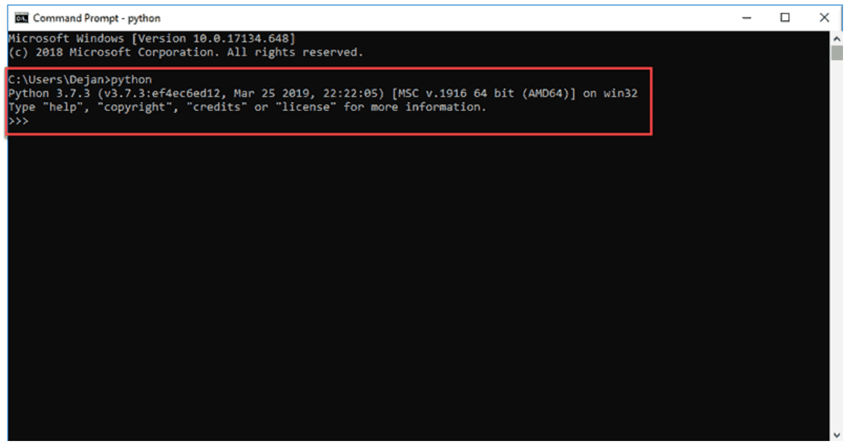
To run our application, please follow below installation steps:

Before you start installing python you need to have **Homebrew** installed on your local machine. If you don't have it installed on your local machine then please follow the below steps to install it.

Step 1: Install Python

To install python on your windows machines, please navigate to https://www.python.org/downloads/ . You need to download the files with the latest version python between 3.5 to 3.8 available on the website. You can also follow these steps on this address: https://phoenixnap.com/kb/how-to-install-python-3-windows .

Step 2: Verify python was installed on windows
Once you have installed python 3 on your local machine, you can test the installation by running-type "python" command into your command prompt. You should see the command line as below:



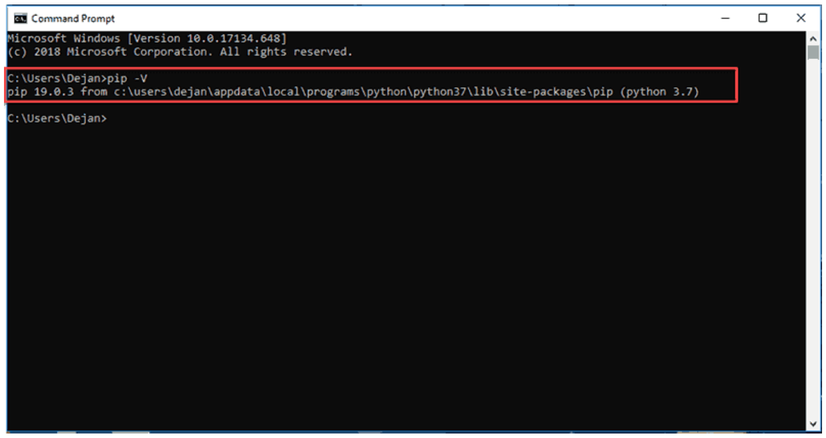
```
Microsoft Windows [Version 10.0.17134.648]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Dejan>python
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 22:22:05) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Note: you can also check whether the installation was successful by typing **python -v** in command prompt.

Step 2: Verify pip was installed
If you opted to install an older version of Python, it is possible that it did not come with Pip preinstalled. Pip is a powerful package management system for Python software packages. Thus, make sure that you have it installed.
We recommend using Pip for most Python packages, especially when working in virtual environments.

To verify whether Pip was installed:

1. Open the **Start** menu and type “**cmd.**”
2. Select the **Command Prompt** application.
3. Enter **pip -V** in the console. If Pip was installed successfully, you should see the following output:



```
Microsoft Windows [Version 10.0.17134.648]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Dejan>pip -V
pip 19.0.3 from c:\users\dejan\appdata\local\programs\python\python37\lib\site-packages\pip (python 3.7)

C:\Users\Dejan>
```

Pip has not been installed yet if you get the following output:

```
'pip' is not recognized as an internal or external command,
Operable program or batch file.
```

Step 3: Install Flask

Step 1: Install Virtual Environment (it is optional because python3 comes with virtualenv installed)

- If Python 2 users must install the *virtualenv* module. **If you have Python 2, follow the instructions outlined in Step 1.**

2. Install virtualenv using pip:

py -2 -m pip install virtualenv

Step 2: Create an Environment

5. Make a separate directory for your project

mkdir <project name>

6. Move into the directory

cd <project name>

7. Within the directory, create the virtual environment for Flask. When you create the environment, a new folder appears in your project directory with the environment's name.

8. Create an Environment in Linux and MacOS

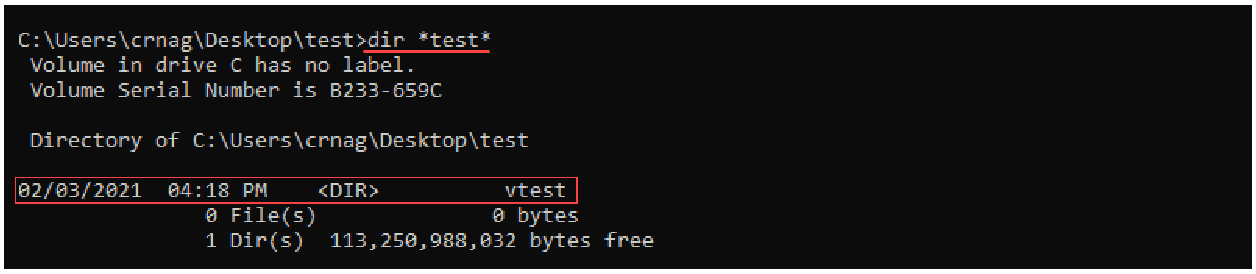
For Python 3:
To create a virtual environment for Python 3, use the *venv* module and give it a name:

py -3 -m venv <name of environment>

List the folder structure using the **dir** command:

dir <project name>

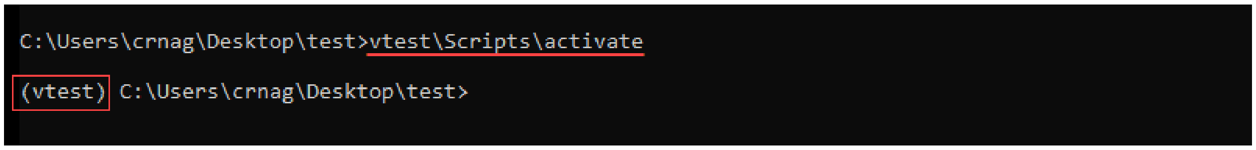
The project directory shows the newly created environment:



```
C:\Users\crnag\Desktop\test>dir *test*
 Volume in drive C has no label.
 Volume Serial Number is B233-659C

 Directory of C:\Users\crnag\Desktop\test

02/03/2021  04:18 PM    <DIR>          vtest
               0 File(s)              0 bytes
               1 Dir(s)  113,250,988,032 bytes free
```

Step 3: Activate the environment
Activate the virtual environment before installing Flask. The name of the activated environment shows up in the CLI after activation.

. <name of environment >Scriptsactivate



```
C:\Users\crnag\Desktop\test>vtest\Scripts\activate

(vtest) C:\Users\crnag\Desktop\test>
```

Step 4: Install Flask

Install Flask within the activated environment using pip:

Pip install Flask

Step 5: Test the Development Environment

1. Create a simple Flask application to test the newly created development environment.
2. Make a file in the Flask project folder called *hello.py*.
3. Edit the file using a text editor and add the following code to make an application that prints "*Hello world!*":

```
from flask import Flask
app = Flask(__name__)
@app.route('/')
def hello_world():
    return 'Hello world!'
```

4. Save the file and close.
5. Using the console, navigate to the project folder using the cd command.
6. Set the *FLASK_APP* environment variable.

setex FLASK_APP "hello.py"

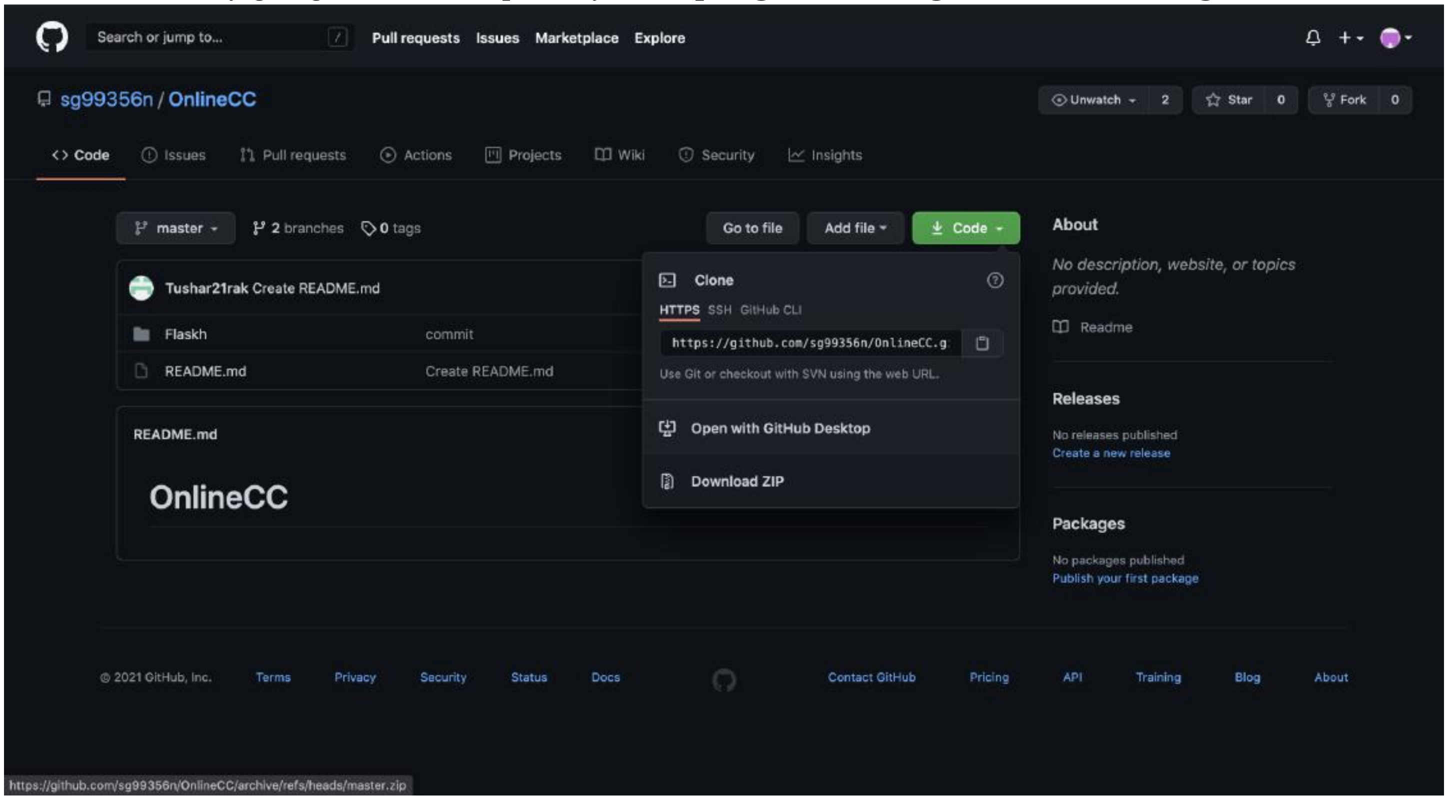
7. Run the Flask Application with:

flask run

8. Copy the address in the browser to see the project running.

Step 4: Download ONCC files from GitHub

You will need to have a copy of the application's files on your local machine in order to run the compiler. This can be achieved by going to GitHub repository at: **https://github.com/sg99356n/OnlineCC.git**



and now select “Download ZIP” from the green download button above the list of files, as shown below. Now unzip the folder and now all the files are available on your system.

Step 5: Now install all requirements for requirements.txt file

Navigate to the location where you saved your virtual environment, enable it, and then return to and enter the folder containing all of the compiler files, where you entered the command.

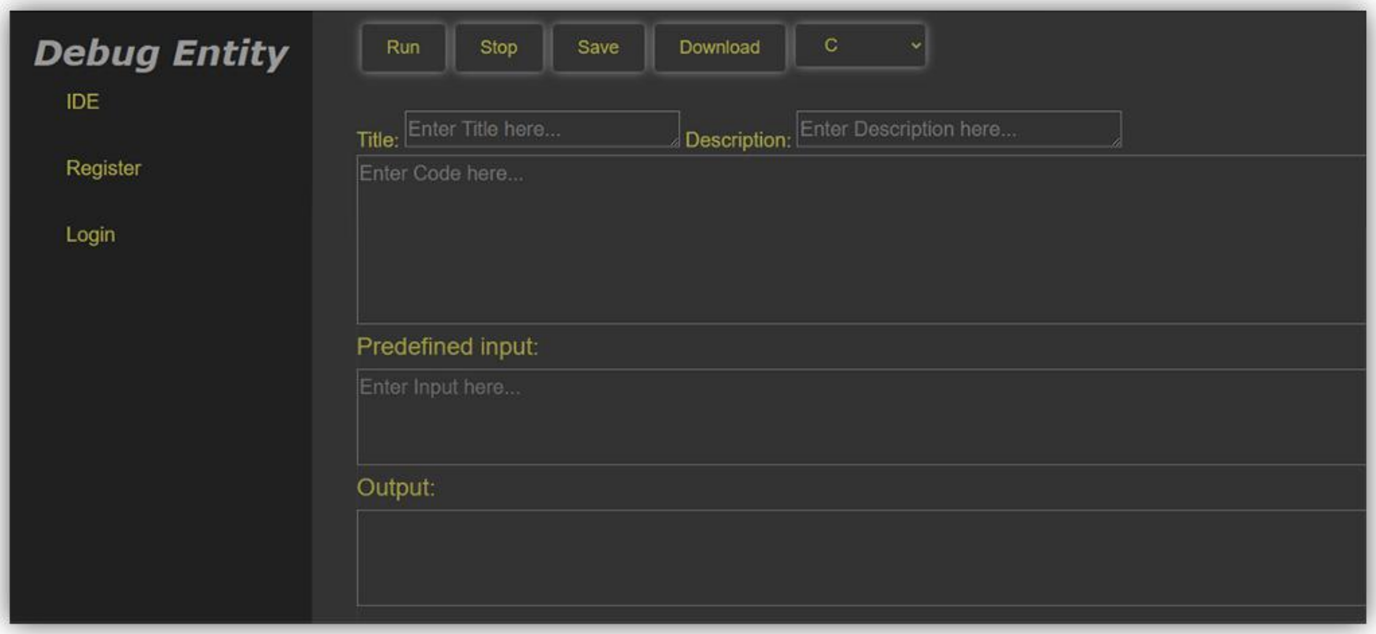
```
$ python3 -m pip install -r requirements.txt
```

Now all the dependencies required to run your application are installed.

Step 7: Running ONCC via localhost
Ensuring that you are still within the root folder containing all of ONCC’s program files, enter the following command into your terminal:

```
(pyvenv) $ python3 manage.py runserver
```

Then navigate to http://localhost:5000 in your browser, and you should see the welcome page of ONCC as shown on the next page:


Debug Entity
IDE
Register
Login
Run
Stop
Save
Download
C
Title:
Enter Title here...
Description:
Enter Description here...
Enter Code here...
Predefined input:
Enter Input here...
Output: