A. Local Deployment (C:\Users\Ritahani\OneDrive\Documents\GitHub)

1. Copy and paste files from CSM Local to the desired machine. Make sure file hierarchy looks like this:

```
└── CSM Local/
    ├── static
    ├── uploads
    ├── CSM.py
    ├── utilsCSM.py
    └── templates/
        ├── baseCSM.html
        ├── CSM.html
        ├── evaluate.html
        ├── result.html
        └── upload.html
```

2. Open CSM.py
3. Click Explorer (Ctrl + Shift + E)
4. Press 'Open Folder'
5. Press 'Select Folder' (make sure its pointing to the same directory)
6. Run the Python file
7. Localhost is running on http://127.0.0.1:5000

B. Online Deployment

If the linux server is set up already, you can skip to instruction 23. If not, you can start by setting up the Linux server.

1. Download and Install Putty. Putty will be used as the Command Line Interface for interacting with the server.
2. Open Putty and fill up the hostname with your server's IP Address.. The other settings can be left as the default setting.
3. Click Open. Fill up your username ("root") and password (the one you registered previously on Linode).
4. Once you're logged in, update the server.

```
sudo apt update
```

5. Download and install Python.

```
sudo apt install python3
```

6. Install Python Pip.

```
Sudo apt-get install python3-pip
```

7. Install Flask.

```
Pip install flask
```

8. Make sure you are in the root folder. Create a new folder to setup your Flask files named “csm-server”.

```
Mkdir csm-server
```

9. Open your new folder.

```
Cd csm-server
```

10. As a rule of thumb, create the same file inside of the file we just created.

```
Mkdir csm-server
```

11. Open your new folder.

```
Cd csm-server
```

12. Open WinSCP. Login using IP Address as hostname, root as username and fill up the password.
13. Now, locate the csm-server directory in which we made earlier. Simply transfer all the Flask files that we made into the csm-server directory that we’ve made.
14. Rename the “app.py” file into “__init__.py”.

```
Sudo apt install nginx
```

15. It’s time to set up the NGINX gateway so that your app can be accessed from the browser.

```
Sudo nano /etc/nginx/sites-enabled/csm-server
```

16. Write the following, replace the server-ip with your own IP Address.:

```
server {

    listen 80;
```

```
    server_name [server-ip];


    location / {

        proxy_pass http://127.0.0.1:8000;

        proxy_set_header Host $host;

        proxy_set_header X-Forwarded-For
$proxy_add_x_forwarded_for;

    }

}
```

17. Check for syntax errors.

```
Sudo nginx -t
```

18. Restart the NGINX server.

```
Sudo systemctl restart nginx
```

19. Gunicorn will be the gateway between NGINX and Flask. Install Gunicorn.

```
Sudo apt-get install gunicorn -y
```

20. Go to the home directory of your __init__.py (one folder up).

```
Cd ..
```

21. Copy and paste files from “CSM Local” folder and make sure your file hierarchy looks like this:

```
└── root/
    └── csm-server/  Note: Change directory here to execute Instruction 22
        ├── uploads
        ├── utilsCSM.py
        └── csm-server/
            ├── CSM.py
            ├── utilsCSM.py
```

```
├── static
└── templates/
    ├── baseCSM.html
    ├── CSM.html
    ├── evaluate.html
    ├── result.html
    └── upload.html
```

22. Finally, run the gunicorn command in order to deploy your model. It is important that in the __init__ file, the flask instance is assigned to “app”.

```
Gunicorn -w 5 csm-server:app
```

23. Right now, your website will end as soon as you close Putty. To avoid that, run these two commands.

```
Sudo apt-get install screen -y
```

```
Screen gunicorn -w 5 csm-server:app
```

24. Congratulations, if everything was configured correctly, you are now able to access your application online at http://[your-server-ip].