

CCL Assignment No. 2

Name- Rohit Anirudha Pendse

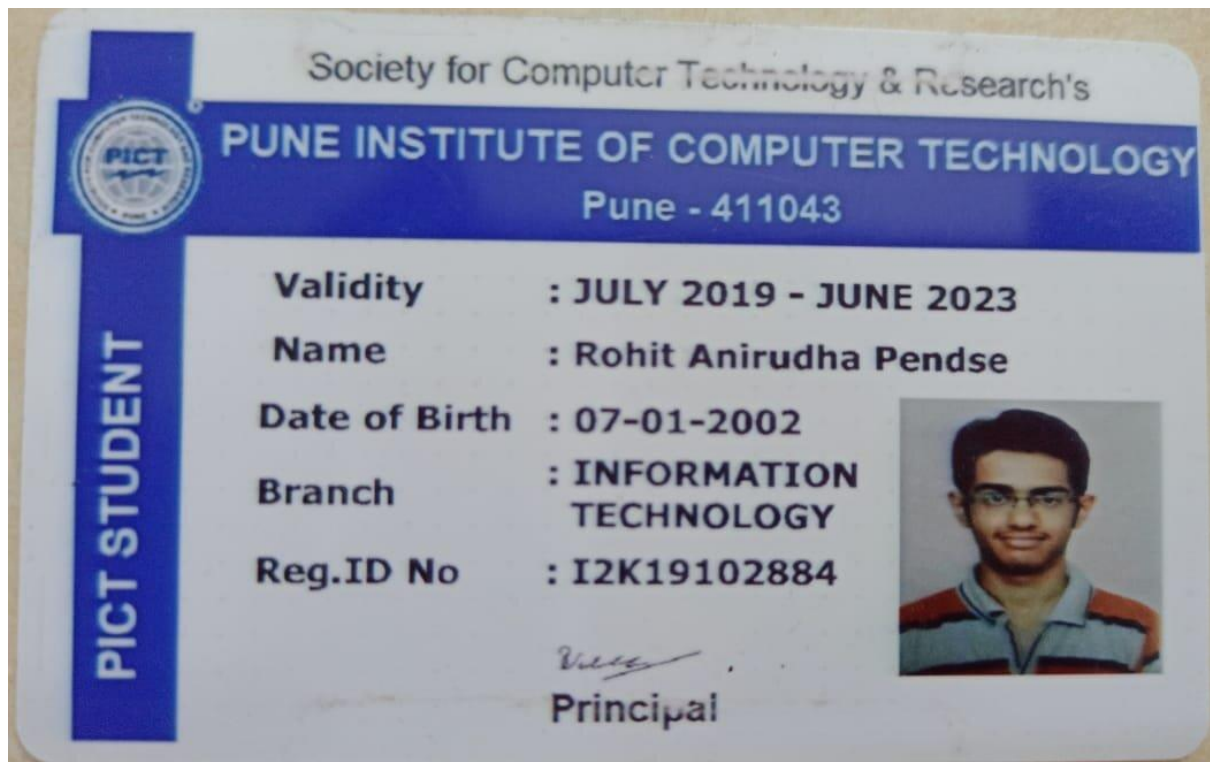
Roll No.- 33358

Batch- N11

Contact No.- 8766925932

Email Id- rapendse2002@gmail.com

ID card-



Name : Rohit Pendse

Roll no: 33358

Cloud Computing - Assignment 2

Aim: Use GAE launcher to launch the web application.

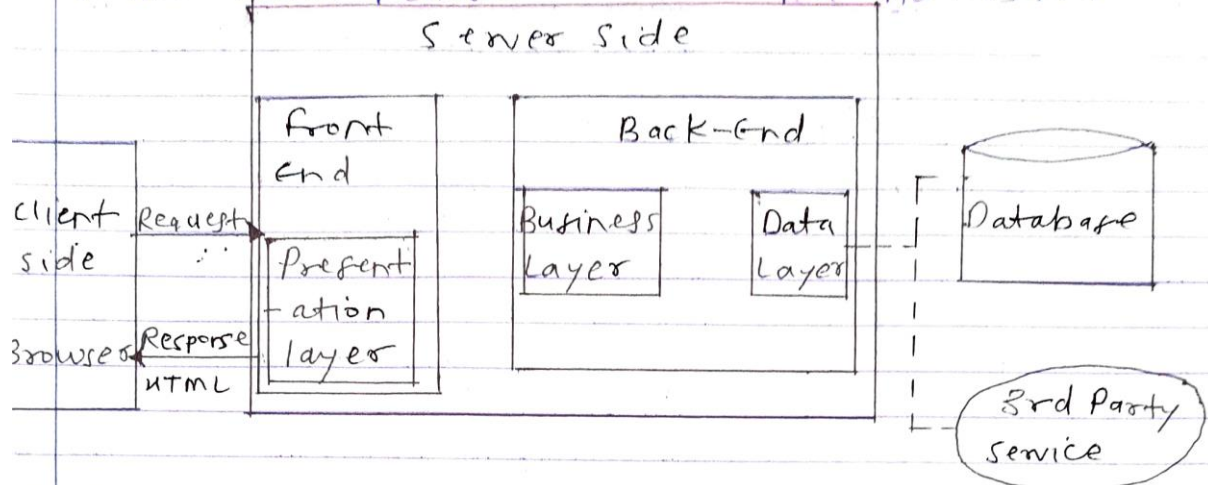
Theory:

1. Mention structure of GAE applications with their tile structure & description.

- ① Each version of your app engine service is defined in app.yaml tile
- ② For similar apps, the minimum requirement for deployment is to define app.yaml tile
- ③ If you are deploying several versions of a service, you can create multiple yaml tiles in same directory to represent configuration for each versions.
- ④ For each service, you can create separate directories in root of your app when deploying locally.
- ⑤ Each directory/repository should represent a single service & contain that service's app.yaml tile alongwith associated source code.
- ⑥ The other configuration tiles should reside in the rest root directory of default service of your app.
- ⑦ Our project directory includes app.yaml, index.html, main.py, result.html

Name: Rohit Pendse

Roll no: 33358



2. What are the main components of web application?

→ All web-based applications have three primary components: A web browser, a web-application server and a database server.

- ① Web-based applications rely only on database server which provides data for application.
- ② Database sometimes provides data for application with business logic in the form of stored procedure.
- ③ The clients handle presentation logic which controls the way with which users interact with the application.
- ④ In some cases, the client validates user-provided inputs.

In terms of layers, they can be categorized as:

- ① view layer: It gives an interface to the application. It is a bridge for getting data in and out of the application.
- ② Data access layer: This layer is built to keep the code you use to put data from the database.

Name: Rohit Pendse

Roll no: 33358

(iii) Data Access Layer: This layer is built to keep the code you use to put the data from your database

(iv) Error handling, security, logging: It is vital part of any application & user experience. It can leave your users feel informed and properly considered

3. what is procedure to develop a simple web app?

→ ① Define the problem

② Plan the workflow of your web application

③ Wireframe your web application

④ Receive validation

⑤ Choose tech-stack to be used in the app

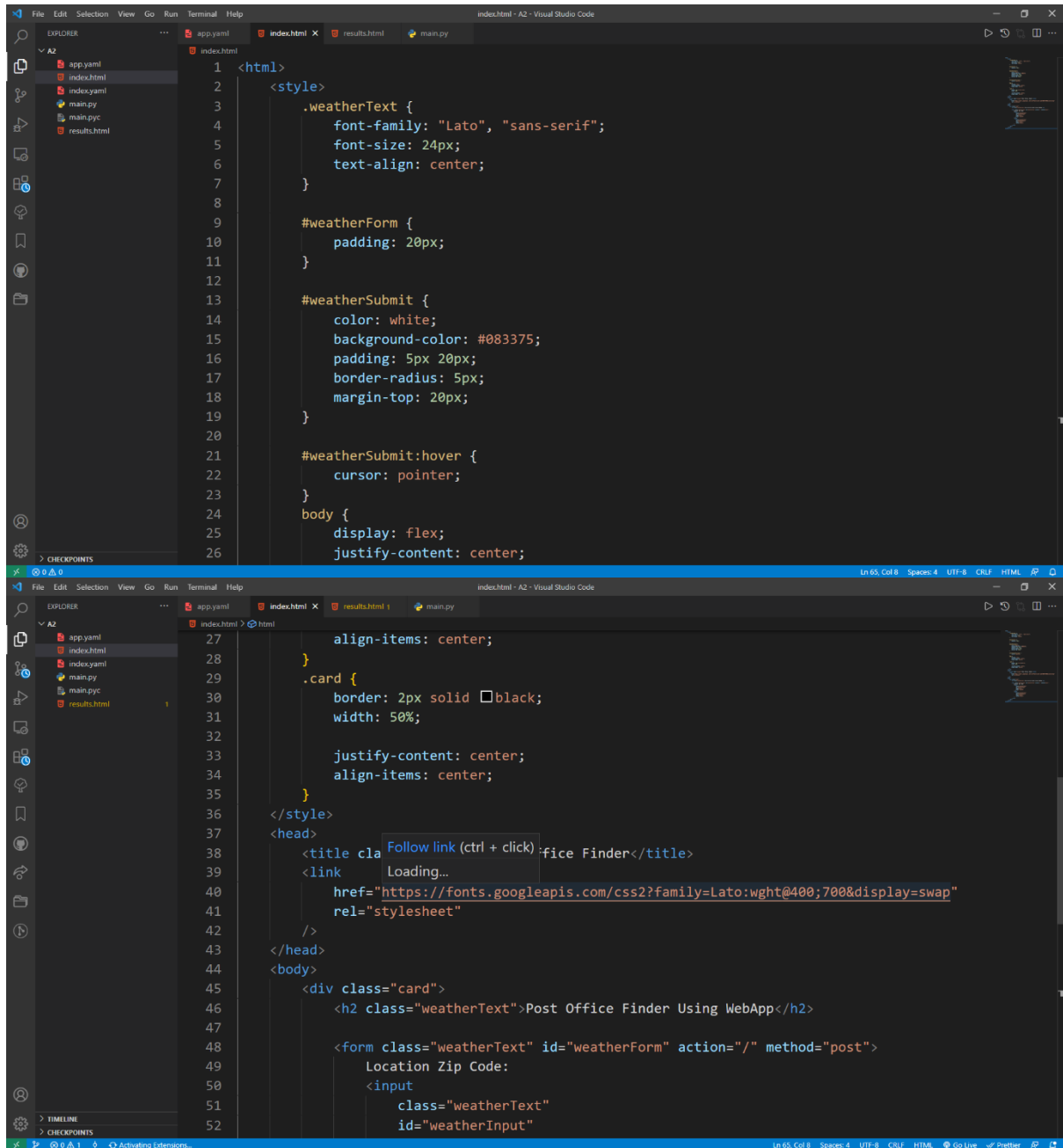
⑥ Start building the application based on design

⑦ Test app after regular intervals.

⑧ Host & deploy your web application

OUTPUT:

a) index.html :



```
1 <html>
2 <style>
3   .weatherText {
4     font-family: "Lato", "sans-serif";
5     font-size: 24px;
6     text-align: center;
7   }
8
9   #weatherForm {
10    padding: 20px;
11  }
12
13  #weatherSubmit {
14    color: white;
15    background-color: #083375;
16    padding: 5px 20px;
17    border-radius: 5px;
18    margin-top: 20px;
19  }
20
21  #weatherSubmit:hover {
22    cursor: pointer;
23  }
24  body {
25    display: flex;
26    justify-content: center;
27
28    align-items: center;
29  }
30  .card {
31    border: 2px solid black;
32    width: 50%;
33
34    justify-content: center;
35    align-items: center;
36  }
37 </style>
38 <head>
39   <title class="weatherText">Post Office Finder Using WebApp</title>
40   <link
41     href="https://fonts.googleapis.com/css2?family=Lato:wght@400;700&display=swap"
42     rel="stylesheet"
43   />
44 </head>
45 <body>
46   <div class="card">
47     <h2 class="weatherText">Post Office Finder Using WebApp</h2>
48
49     <form class="weatherText" id="weatherForm" action="/" method="post">
50       Location Zip Code:
51       <input
52         class="weatherText"
53         id="weatherInput"
54       />
55     </form>
56   </div>
57 </body>
58 </html>
```

```
41     rel="stylesheet"
42   />
43 </head>
44 <body>
45   <div class="card">
46     <h2 class="weatherText">Post Office Finder Using WebApp</h2>
47
48     <form class="weatherText" id="weatherForm" action="/" method="post">
49       Location Zip Code:
50       <input
51         class="weatherText"
52         id="weatherInput"
53         type="text"
54         name="zipCode"
55       /><br />
56       <input
57         class="weatherText"
58         id="weatherSubmit"
59         type="submit"
60         value="Submit"
61       />
62     </form>
63   </div>
64 </body>
65 </html>
```

b) Results.html:

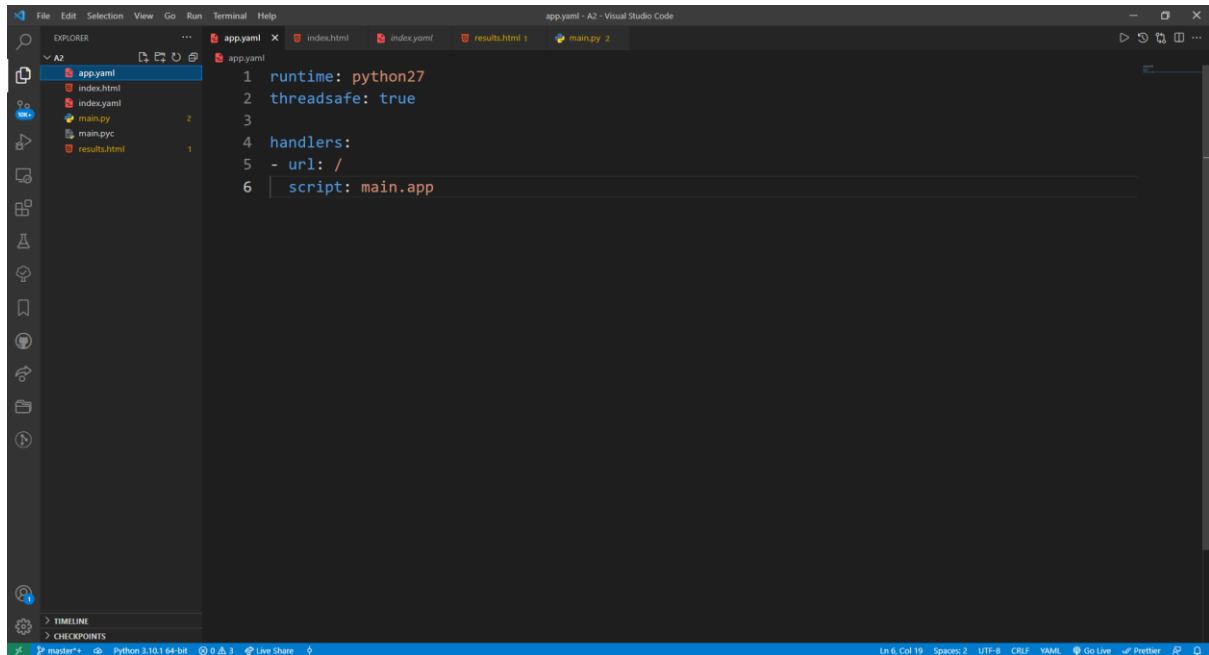
```
1 <!DOCTYPE html>
2 <html lang="en">
3   <style>
4     body {
5       display: flex;
6       justify-content: center;
7       align-items: center;
8     }
9     #weatherResults {
10      background-color: #83e9c2;
11      font-family: "Lato", sans-serif;
12      font-size: 24px;
13      padding: 30px;
14      display: inline-block;
15      text-align: center;
16      margin: 20px;
17      margin-top: 10%;
18      border: 2px solid black;
19      border-radius: 5px;
20    }
21  </style>
22  <head>
23    <meta charset="UTF-8" />
24    <title>Post Office Information</title>
25    <link
26      href="https://fonts.googleapis.com/css2?family=Lato:wght@400;700&display=swap"
```

```
File Edit Selection View Go Run Terminal Help
results.html - A2 - Visual Studio Code

EXPLORER
  Az
  app.yaml
  index.html
  index.yaml
  main.py
  main.pyc
  results.html 1

26 href="https://fonts.googleapis.com/css2?family=Lato:wght@400;700&display=swap"
27 rel="stylesheet"
28 />
29 </head>
30 <body>
31   <div id="weatherResults">
32     <table>
33       <tr>
34         <th>
35           <h3>State of Post Office :</h3>
36         </th>
37         <th>
38           <h3>{{ post_office }}</h3>
39         </th>
40       </tr>
41       <tr>
42         <th>
43           <h3>Name of Post Office :</h3>
44         </th>
45         <th>
46           <h3>{{ name }}</h3>
47         </th>
48       </tr>
49       <tr>
50         <th>
51           <h3>Block of Post Office:</h3>
52         </th>
53         <th>
54           <h3>{{ block }}</h3>
55         </th>
56       </tr>
57       <tr>
58         <th>
59           <h3>District of Post Office:</h3>
60         </th>
61         <th>
62           <h3>{{ district }}</h3>
63         </th>
64       </tr>
65     </table>
66   </div>
67 </body>
68 </html>
```

c) app.yaml :

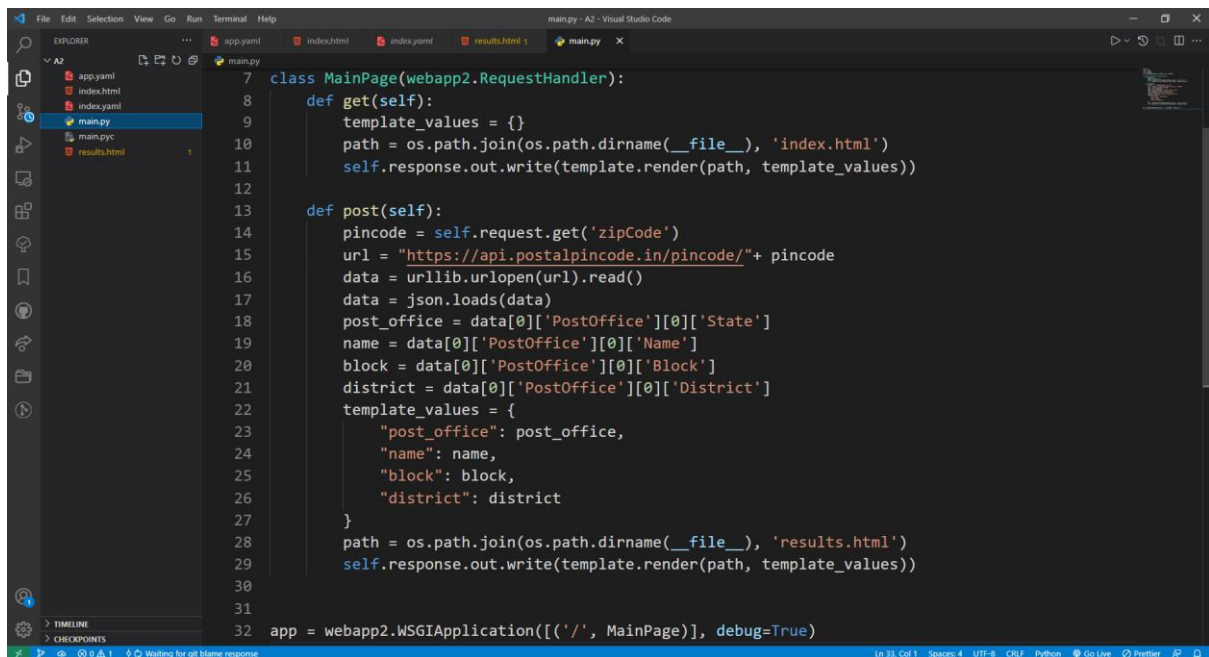


The screenshot shows the Visual Studio Code editor with the 'app.yaml' file open. The Explorer sidebar on the left shows a project structure with files: app.yaml, index.html, index.yaml, main.py, main.pyc, and results.html. The main editor area displays the content of 'app.yaml' with the following code:

```
1 runtime: python27
2 threadsafe: true
3
4 handlers:
5 - url: /
6   script: main.app
```

The status bar at the bottom indicates the file is at Line 6, Column 19, using UTF-8 encoding, CRLF line endings, and the YAML language mode.

d) main.py :



The screenshot shows the Visual Studio Code editor with the 'main.py' file open. The Explorer sidebar on the left shows the same project structure as in the previous image. The main editor area displays the content of 'main.py' with the following code:

```
7 class MainPage(webapp2.RequestHandler):
8     def get(self):
9         template_values = {}
10        path = os.path.join(os.path.dirname(__file__), 'index.html')
11        self.response.out.write(template.render(path, template_values))
12
13    def post(self):
14        pincode = self.request.get('zipCode')
15        url = "https://api.postalpincode.in/pincode/" + pincode
16        data = urllib.urlopen(url).read()
17        data = json.loads(data)
18        post_office = data[0]['PostOffice'][0]['State']
19        name = data[0]['PostOffice'][0]['Name']
20        block = data[0]['PostOffice'][0]['Block']
21        district = data[0]['PostOffice'][0]['District']
22        template_values = {
23            "post_office": post_office,
24            "name": name,
25            "block": block,
26            "district": district
27        }
28        path = os.path.join(os.path.dirname(__file__), 'results.html')
29        self.response.out.write(template.render(path, template_values))
30
31
32 app = webapp2.WSGIApplication([('/', MainPage)], debug=True)
```

The status bar at the bottom indicates the file is at Line 33, Column 1, using UTF-8 encoding, CRLF line endings, and the Python language mode.

e) Google Cloud SDK Console:

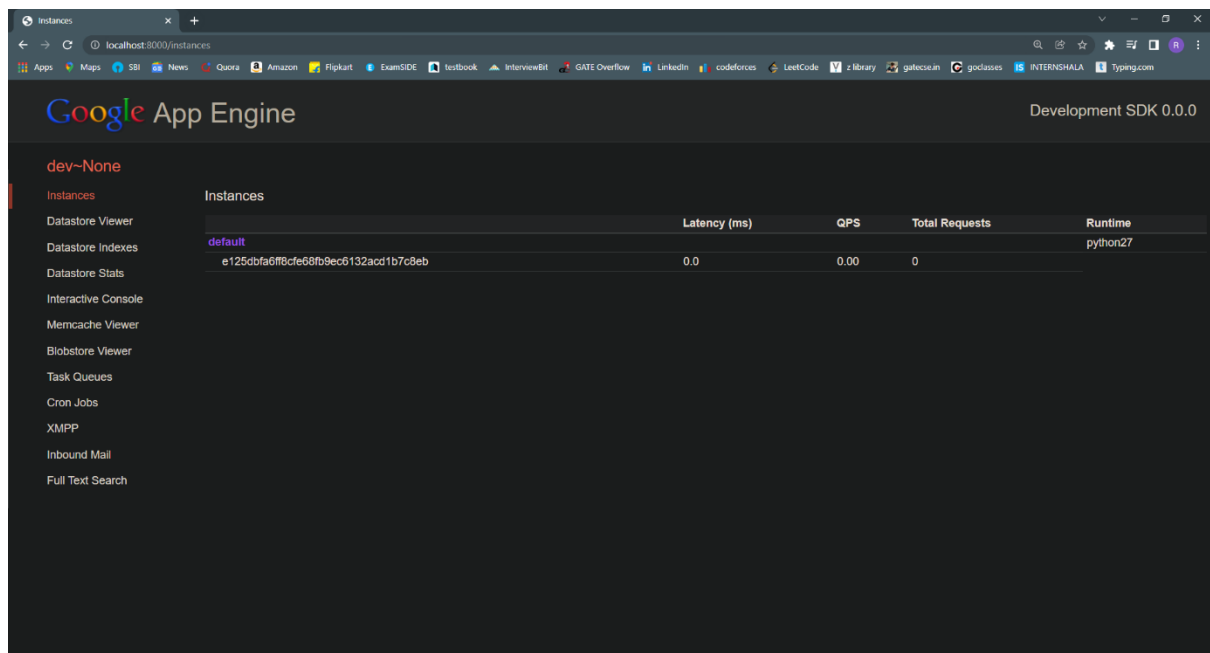
```
Google Cloud SDK Shell - py google-cloud-sdk\bin\dev_appserver.py D:\ROHIT\TE_Assignment_SEM_II\CCL\A2
Welcome to the Google Cloud SDK! Run "gcloud -h" to get the list of available commands.
---
C:\Users\rapien\AppData\Local\Google\Cloud SDK>py google-cloud-sdk\bin\dev_appserver.py D:\ROHIT\TE_Assignment_SEM_II\CCL\A2

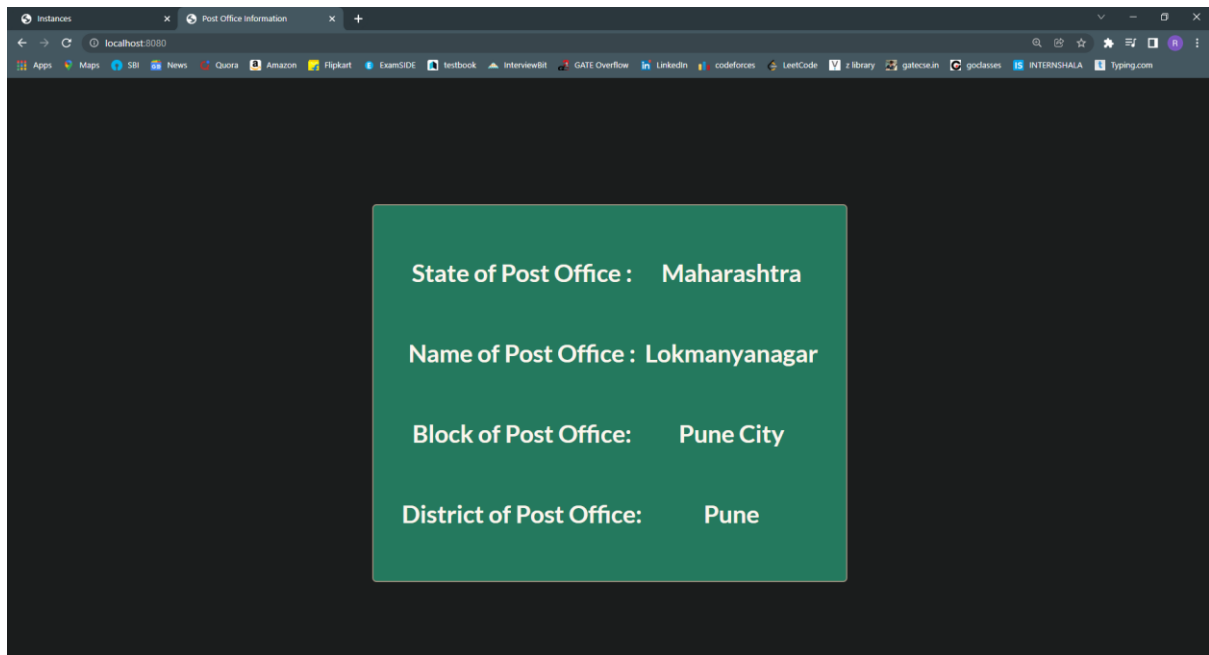
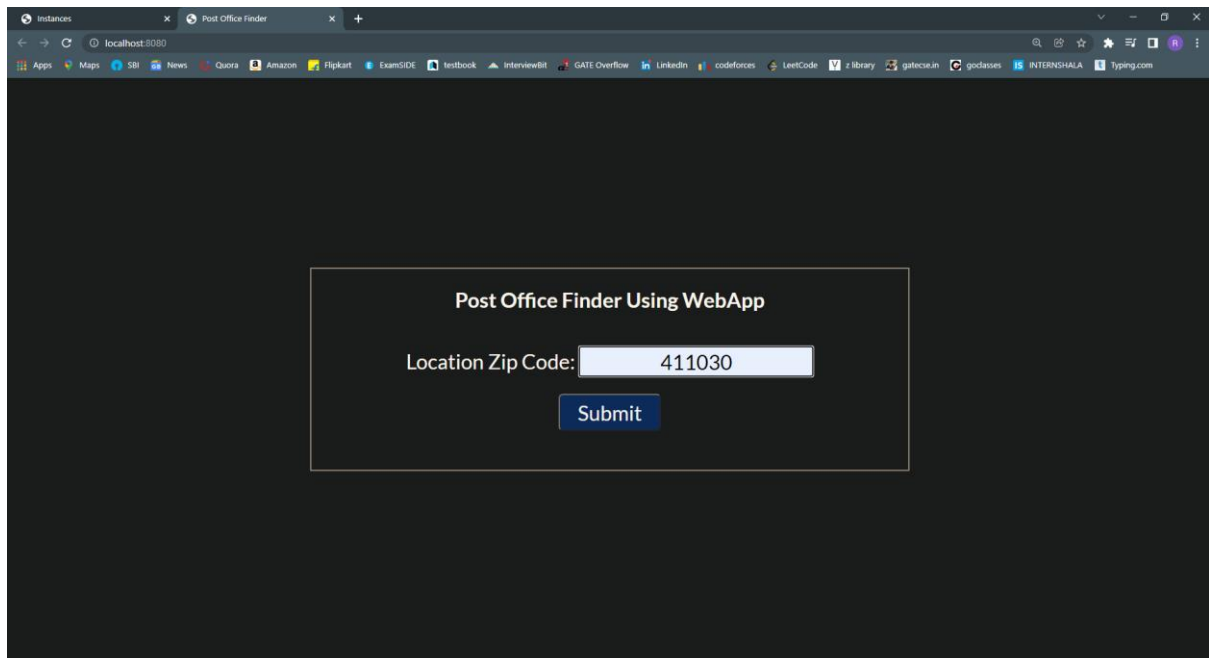
Updates are available for some Cloud SDK components. To install them,
please run:
$ gcloud components update

INFO 2022-01-20 10:52:47,226 devappserver2.py:239] Using cloud Datastore Emulator.
We are gradually rolling out the emulator as the default datastore implementation of dev_appserver.
If broken, you can temporarily disable it by --support_datastore_emulator=False
Read the documentation: https://cloud.google.com/appengine/docs/standard/python/tools/migrate-cloud-datastore-emulator
Help us validate that the feature is ready by taking this survey: https://goo.gl/forms/UArIcs8K9CUsCm733
Report issues at: https://issuetracker.google.com/issues/new?component=187272

INFO 2022-01-20 10:52:47,316 devappserver2.py:316] Skipping SDK update check.
INFO 2022-01-20 10:52:49,410 datastore_emulator.py:156] Starting Cloud Datastore emulator at: http://localhost:24009
WARNING 2022-01-20 10:52:53,614 simple_search_stub.py:1196] Could not read search indexes from c:\users\rapien\appdata\local\temp\appen
gine.None\search_indexes
INFO 2022-01-20 10:52:56,928 datastore_emulator.py:162] Cloud Datastore emulator responded after 7.622000 seconds
INFO 2022-01-20 10:52:56,944 <string>:383] Starting API server at: http://localhost:55516
INFO 2022-01-20 10:52:57,069 <string>:373] Starting gRPC API server at: http://localhost:55517
INFO 2022-01-20 10:52:57,575 dispatcher.py:281] Starting module "default" running at: http://localhost:8080
INFO 2022-01-20 10:52:57,588 admin_server.py:150] Starting admin server at: http://localhost:8000
INFO 2022-01-20 10:53:06,609 instance.py:294] Instance PID: 13080
INFO 2022-01-20 10:53:44,296 module.py:443] [default] Detected file changes:
main.py
INFO 2022-01-20 10:53:45,848 module.py:883] default: "GET / HTTP/1.1" 200 1670
INFO 2022-01-20 10:53:46,354 instance.py:294] Instance PID: 14096
INFO 2022-01-20 10:53:47,111 module.py:883] default: "GET /favicon.ico HTTP/1.1" 404 -
INFO 2022-01-20 10:53:49,388 instance.py:294] Instance PID: 14248
WARNING 2022-01-20 05:24:03,457 sandbox.py:1135] The module _winreg is whitelisted for local dev only. If your application relies on _
winreg, it is likely that it will not function properly in production.
WARNING 2022-01-20 10:54:03,523 urlfetch_stub.py:575] Stripped prohibited headers from URLFetch request: ['Host']
INFO 2022-01-20 10:54:05,206 module.py:883] default: "POST / HTTP/1.1" 200 -
INFO 2022-01-20 10:54:32,466 module.py:883] default: "GET / HTTP/1.1" 200 1670
WARNING 2022-01-20 10:54:51,440 urlfetch_stub.py:575] Stripped prohibited headers from URLFetch request: ['Host']
INFO 2022-01-20 10:54:53,092 module.py:883] default: "POST / HTTP/1.1" 200 -
INFO 2022-01-20 10:55:00,632 module.py:883] default: "GET / HTTP/1.1" 200 1670
WARNING 2022-01-20 10:55:06,009 urlfetch_stub.py:575] Stripped prohibited headers from URLFetch request: ['Host']
```

f) Browser output screenshots:





Conclusion:

