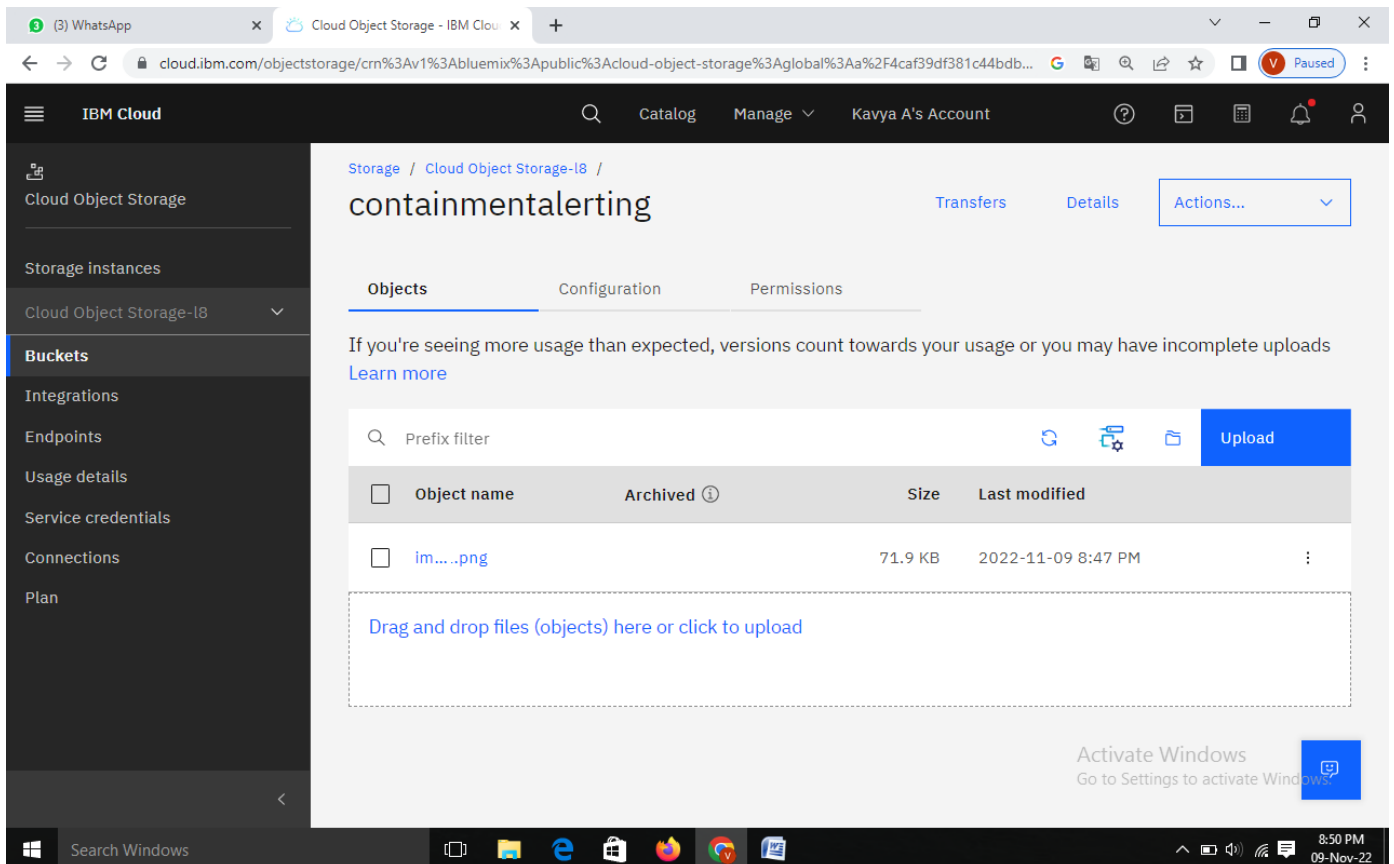


Assignment-3

Date	10 October 2022
Team ID	PNT2022TMID30533
Project Name	CONTAINMENT ZONE ALERTING APPLICATION

1. CREATE A BUCKET IN IBM OBJECT STORAGE.



2. Upload an 5 images to ibm object storage and make it public.
write html code todisplaying all the 5 images.

IBM Cloud

Cloud Object Storage

Storage instances

Cloud Object Storage-l8

Buckets

Integrations

Endpoints

Usage details

Service credentials

Connections

Plan

Access policy type

☒ User ☐ Service ID ☐ Access Group

Select a user:

Kavya A (kavyakannan446@gmail.com) [Add users](#)

Access policy update

User policy created

A new access policy for this bucket was created for the user:
Kavya A (kavyakannan446@gmail.com)
To delete/edit go to the [IAM console](#).

Public access

Context-based restrictions

Firewall (legacy)

Activate Windows
Go to Settings to activate Windows.

Search Windows

8:53 PM
09-Nov-22

IBM Cloud

Cloud Object Storage

Storage instances

Cloud Object Storage-l8

Buckets

Integrations

Endpoints

Usage details

Service credentials

Connections

Plan

Buckets

Buckets serve as containers for objects, and can be individually configured in terms of their location, resiliency, billing rates, security, and object lifecycle rules.

Search

Create bucket +

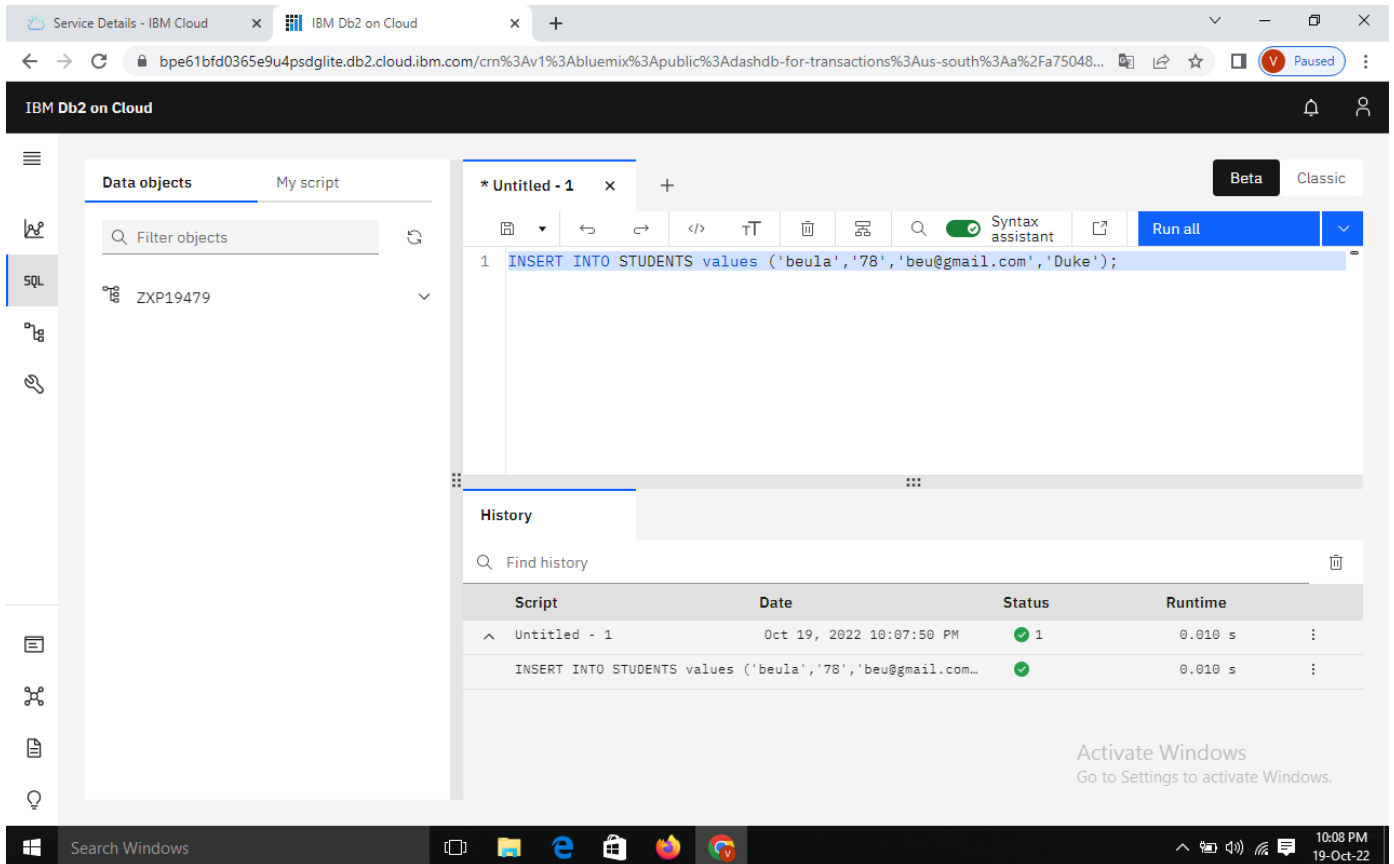
Name	Public access ⓘ	Location ⓘ	Storage class	Created
containmentalerting	Yes	jp-tok	Smart Tier	2022-11-09 8:45 PM

Activate Windows
Go to Settings to activate Windows.

Search Windows

8:56 PM
09-Nov-22

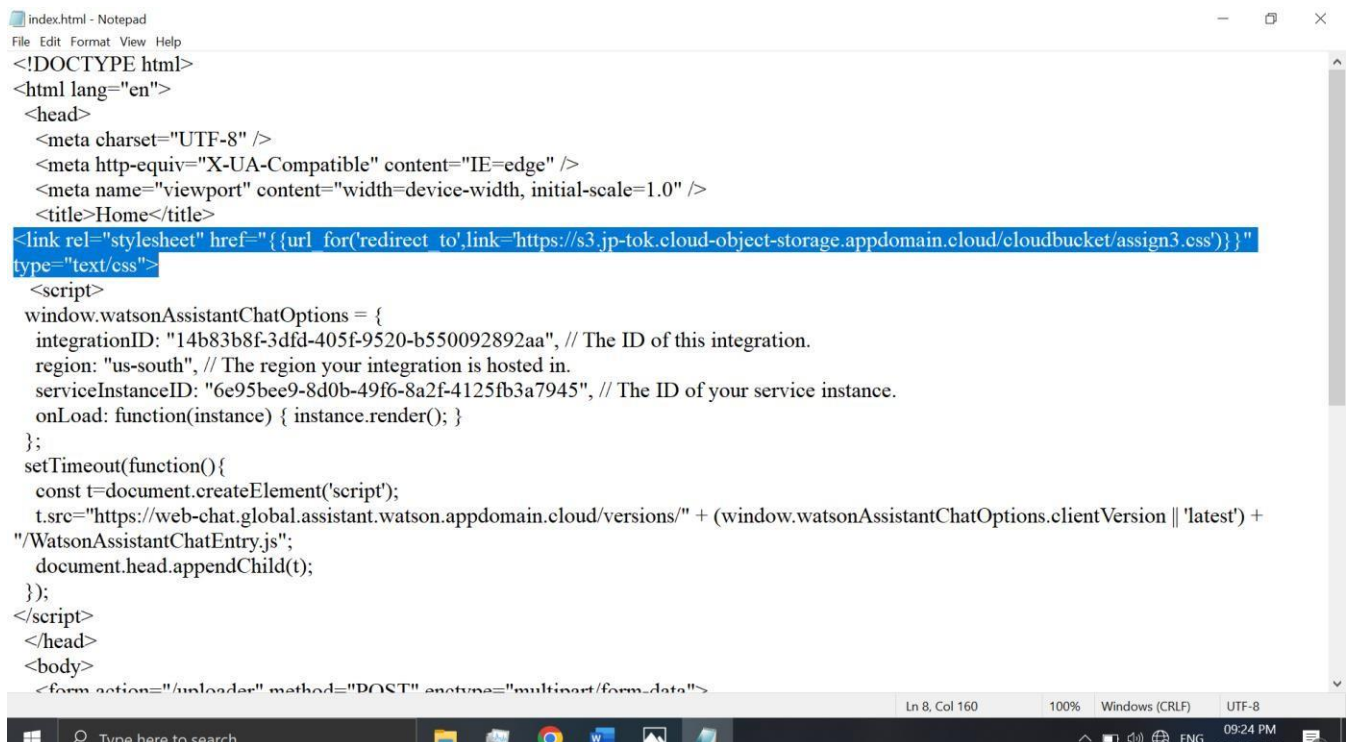
2. Upload a css page to the object storage and use the same page in your HTML code.



The screenshot shows the IBM Db2 on Cloud console. On the left, the 'Data objects' tab is selected, showing a search bar and a list of objects including 'ZXP19479'. The main area displays a SQL script in a text editor: `INSERT INTO STUDENTS values ('beula','78','beu@gmail.com','Duke');`. The script is highlighted in blue. Below the editor, a 'History' table shows the execution details:

Script	Date	Status	Runtime
Untitled - 1	Oct 19, 2022 10:07:50 PM	✓ 1	0.010 s
INSERT INTO STUDENTS values ('beula','78','beu@gmail.com...		✓	0.010 s

The bottom of the console shows a Windows taskbar with the search bar and various application icons.

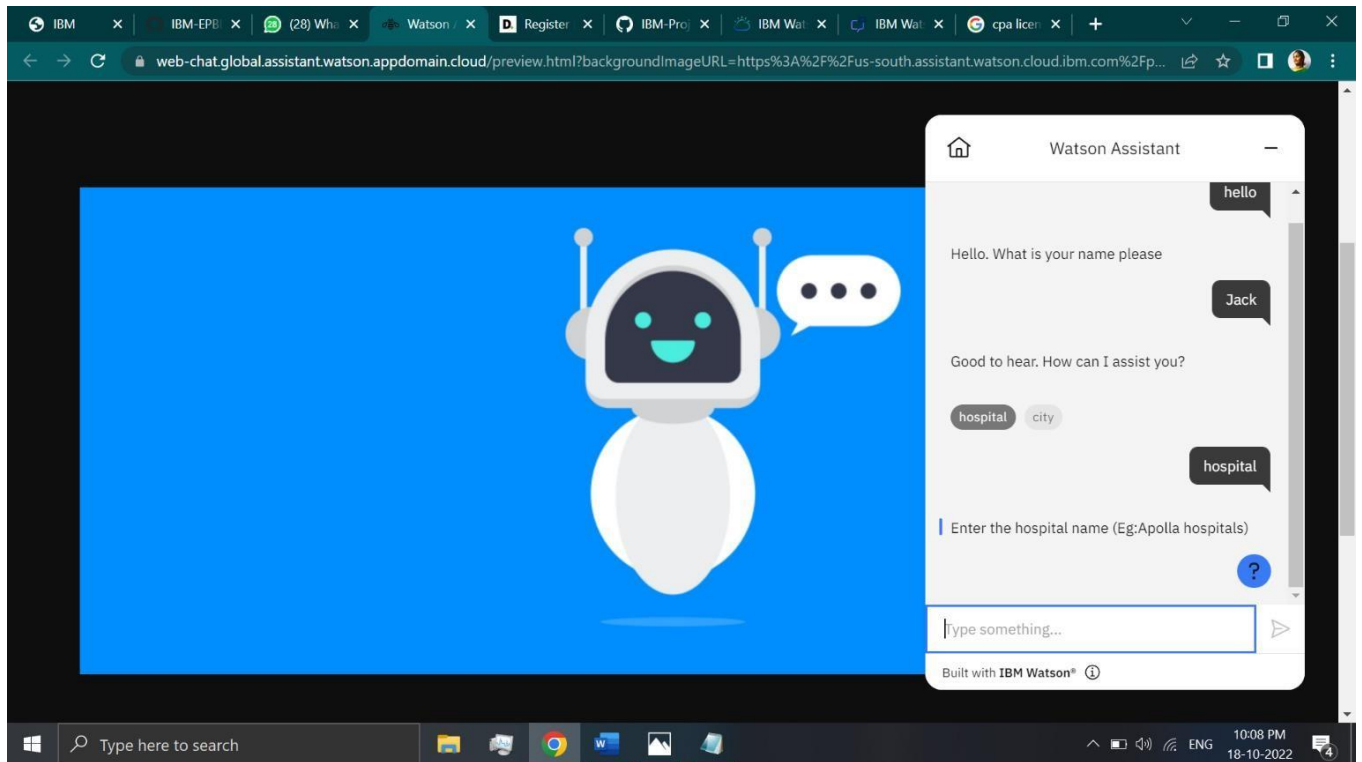


The screenshot shows a Notepad++ window editing a file named 'index.html'. The code is as follows:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta http-equiv="X-UA-Compatible" content="IE=edge" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Home</title>
  <link rel="stylesheet" href="{url_for('redirect to',link='https://s3.jp-tok.cloud-object-storage.appdomain.cloud/cloudbucket/assign3.css')}" type="text/css">
  <script>
    window.watsonAssistantChatOptions = {
      integrationID: "14b83b8f-3dfd-405f-9520-b550092892aa", // The ID of this integration.
      region: "us-south", // The region your integration is hosted in.
      serviceInstanceID: "6e95bee9-8d0b-49f6-8a2f-4125fb3a7945", // The ID of your service instance.
      onLoad: function(instance) { instance.render(); }
    };
    setTimeout(function() {
      const t=document.createElement('script');
      t.src="https://web-chat.global.assistant.watson.appdomain.cloud/versions/" + (window.watsonAssistantChatOptions.clientVersion || 'latest') +
        "/WatsonAssistantChatEntry.js";
      document.head.appendChild(t);
    });
  </script>
</head>
<body>
  <form action="/uploader" method="POST" enctype="multipart/form-data">
```

The bottom of the window shows a Windows taskbar with the search bar and various application icons.

3. Design a chatbot using IBM Watson assistant for hospital.



Web URL for Assistant:

<https://web-chat.global.assistant.watson.appdomain.cloud/preview.html?backgroundImageURL=https%3A%2F%2Fus-south.assistant.watson.cloud.ibm.com%2Fpublic%2Fimages%2Fupx-6e95bee9-8d0b-49f6-8a2f-4125fb3a7945%3A%3Ab437fcda-6135-46e5-9e62-faa901cdce2d&integrationID=14b83b8f-3dfd-405f-9520-b550092892aa®ion=us-south&serviceInstanceID=6e95bee9-8d0b-49f6-8a2f-4125fb3a7945>

4. Create Watson assistant service with 10 steps and use 3 conditions in it. Load thatscript in HTML page.

The screenshot displays the IBM Watson Assistant Lite web interface. The top navigation bar includes 'IBM Watson Assistant Lite', 'Upgrade', 'Fashion Bot', and a 'Learning center' link. The main content area is divided into two panels. The left panel shows a list of conversation steps, with step 10, 'Bed Availability', highlighted. This step includes the text 'Today 50 beds available. Thank you' and the action 'Action complete'. The right panel shows the 'Assistant says' section with a text input field containing 'Today 50 beds available. Thank you'. Below this is the 'And then' section with a dropdown menu set to 'End the action'. A 'Preview' button is located at the bottom right of the interface. The bottom of the screen shows a Windows taskbar with various application icons and a system tray displaying the temperature (29°C), weather (Cloudy), and time (11:34, 19-10-2022).

Figure 1. 10 steps of conversation

Included 3 conditions in steps:

The screenshot displays the IBM Watson Assistant Lite interface. On the left, a conversation flow diagram shows four steps. Step 3 is highlighted with a blue border and contains the text "Enter the hospital name (Eg: Apollo hospitals)" and a "Free text" input field. On the right, the configuration panel for Step 3 is shown. It indicates that Step 3 is taken "with conditions". Under the "Conditions" section, there are two conditions listed: "2. Good to he... is hospital" and "2. Good to he... is city". The "Assistant says" section shows the output text "Enter the hospital name (Eg: Apollo hospitals)".

The screenshot displays the IBM Watson Assistant Lite interface. On the left, a conversation flow diagram shows five steps. Step 5 is highlighted with a blue border and contains the text "Hospital is 30km away from your location". On the right, the configuration panel for Step 5 is shown. It indicates that Step 5 is taken "with conditions". Under the "Conditions" section, there is one condition listed: "4. Here are th... is Distance". The "Assistant says" section shows the output text "Hospital is 30km away from your location".

 |

Step 8 is taken with conditions

 f_x

If $\frac{1}{2} \leq \frac{1}{2} \leq 1$ of this is true:

4. Here are th... is Parking facility

and Add condition +

Newvcondition group +

B *I*      

</>

Yes, this hospital has parking facility

Index.html

```
<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="UTF-8" />

    <meta http-equiv="X-UA-Compatible" content="IE=edge" />

    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

    <title>Home</title>

    <link rel="stylesheet" href="{ { url_for('redirect_to',link='https://s3.jp-tok.cloud-object-storage.appdomain.cloud/cloudbucket/assign3.css') } }" type="text/css">

    <script>

      window.watsonAssistantChatOptions = {

        integrationID: "14b83b8f-3dfd-405f-9520-b550092892aa", // The ID of this integration.

        region: "us-south", // The region your integration is hosted in.

        serviceInstanceID: "6e95bee9-8d0b-49f6-8a2f-4125fb3a7945", // The ID of your service instance.

        onLoad: function(instance) { instance.render(); }

      };

      setTimeout(function(){

        const t=document.createElement('script');

        t.src="https://web-chat.global.assistant.watson.appdomain.cloud/versions/" +
(window.watsonAssistantChatOptions.clientVersion || 'latest') + "/WatsonAssistantChatEntry.js";

        document.head.appendChild(t);

      });

    </script> </head>

    <body>

      <form action="/uploader" method="POST" enctype="multipart/form-data">

        <input type="text" placeholder="Enter file name" name="filename" />

        <br />

        <br />

        <input type="file" name="file" />

        <br />

        <br />

        <input type="submit" />

      </form>

    </body>

  </html>
```



```

</form>

<br/>

<br/>

<br/>

{% for row in files %}

    <div style="border: 1px solid #EFEFEF;margin:10px;">

        <h3>Filename : { {row}} </h3>

        </td>

    </div>

{% endfor %}

</body>

</html>

```

App.py

```

import io

from flask import Flask,redirect,url_for,render_template,request

import ibm_boto3

from ibm_botocore.client import Config, ClientError

COS_ENDPOINT="https://s3.jp-tok.cloud-object-storage.appdomain.cloud"

COS_API_KEY_ID=""

COS_INSTANCE_CRN=""


cos = ibm_boto3.resource("s3",
    ibm_api_key_id=COS_API_KEY_ID,
    ibm_service_instance_id=COS_INSTANCE_CRN,
    config=Config(signature_version="oauth"),
    endpoint_url=COS_ENDPOINT
)

```

```
app=Flask(_name_)
```

```
@app.route('/')
```

```
def index():
```

```
try:
```

```
    files = cos.Bucket('cloudbucket').objects.all()
```

```
    files_names = []
```

```
    for file in files:
```

```
        files_names.append(file.key)
```

```
        print(file)
```

```
        print("Item: {0} ({1} bytes)".format(file.key, file.size))
```

```
    return render_template('index.html',files=files_names)
```

```
except ClientError as be:
```

```
    print("CLIENT ERROR: {0}\n".format(be))
```

```
    return render_template('index.html')
```

```
except Exception as e:
```

```
    print("Unable to retrieve bucket contents: {0}".format(e))
```

```
    return render_template('index.html')
```

```
@app.route('/uploader',methods=['POST'])
```

```
def upload():
```

```
name_file=request.form['filename']
```

```
f = request.files['file']
```

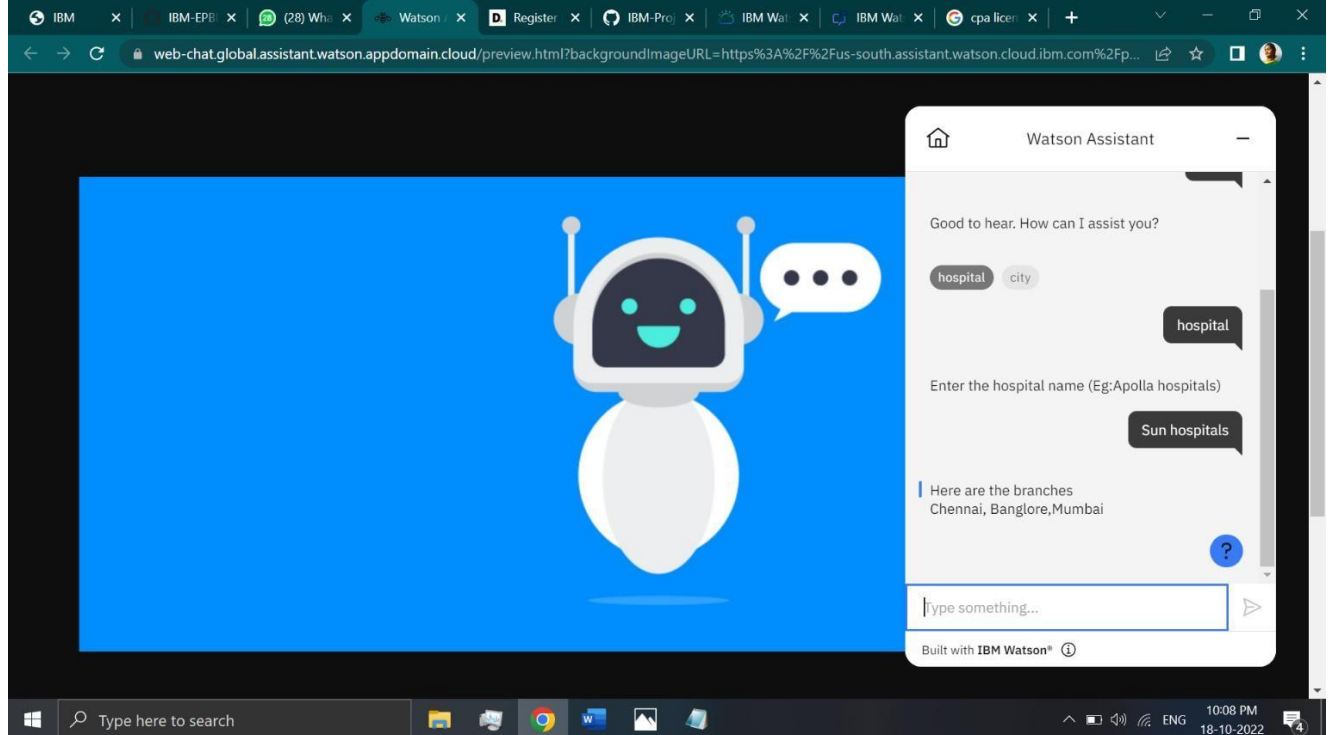
```
try:
```

```
    part_size = 1024 * 1024 * 5
```

```
    file_threshold = 1024 * 1024 * 15
```

```
    transfer_config = ibm_boto3.s3.transfer.TransferConfig(
```

```
        multipart_threshold=file_threshold,
```



```
multipart_chunksize=part_size
```

```
)
```

```
content = f.read()
```

```
cos.Object('cloudbucket', name_file).upload_fileobj(
```

```
    Fileobj=io.BytesIO(content),
```

```
    Config=transfer_config
```

```
)
```

```
return redirect(url_for('index'))
```

```
except ClientError as be:
```

```
    print("CLIENT ERROR: {0}\n".format(be))
```

```
    return redirect(url_for('index'))
```

```
except Exception as e:
```

```
    print("Unable to complete multi-part upload: {0}".format(e))
```

```
    return redirect(url_for('index'))
```

```
if __name__ == '__main__':
```

```
    app.run(host='0.0.0.0', port=8080, debug=True)
```