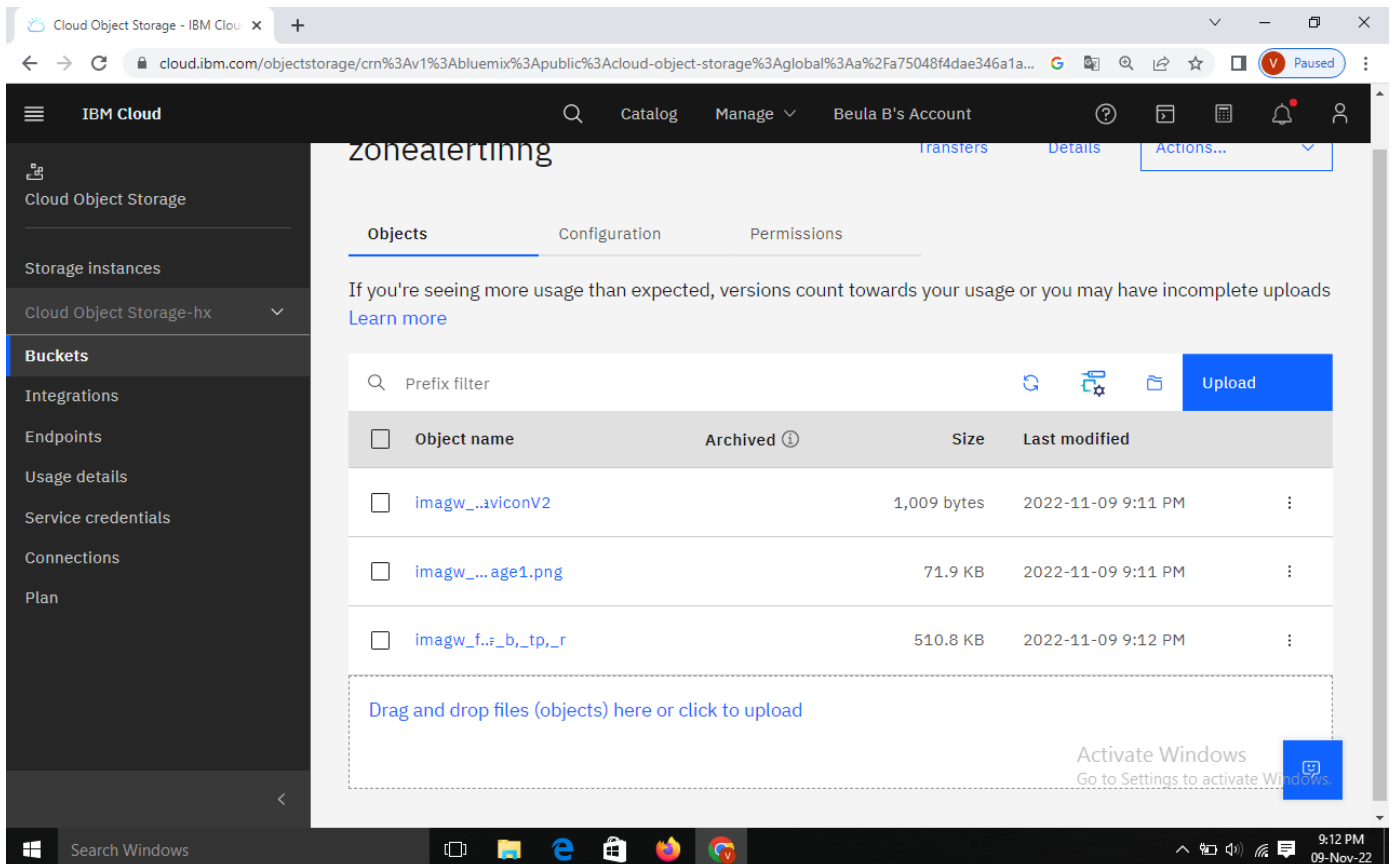


### Assignment-3

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.

Cloud Object Storage - IBM Cloud

cloud.ibm.com/objectstorage/crn%3Av1%3Abluemix%3Apublic%3Acloud-object-storage%3Aglobal%3Aa%2Fa75048f4dae346a1a...

IBM Cloud

Cloud Object Storage

Storage instances

Cloud Object Storage-hx

**Buckets**

Integrations

Endpoints

Usage details

Service credentials

Connections

Plan

### Bucket access policies

Manage access to this bucket by creating IAM policies for users and service IDs. Users and service IDs must also have an instance level viewer role (or higher) to use the console or to list buckets using the REST API.

**Access policies**

Access policy update

**Access group policy created**

A new access policy for this bucket was created for the group:  
Public Access  
To delete/edit go to the [IAM console](#).

As a Content Reader, one can read and list objects in the bucket.

Create access policy

**Context-based restrictions**

Create rules to restrict access to your bucket based on resource attributes and contexts.

Activate Windows  
Go to Settings to activate Windows.

Search Windows

9:13 PM  
09-Nov-22

Cloud Object Storage - IBM Cloud

cloud.ibm.com/objectstorage/crn%3Av1%3Abluemix%3Apublic%3Acloud-object-storage%3Aglobal%3Aa%2Fa75048f4dae346a1a...

IBM Cloud

Cloud Object Storage

Storage instances

Cloud Object Storage-hx

**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 +

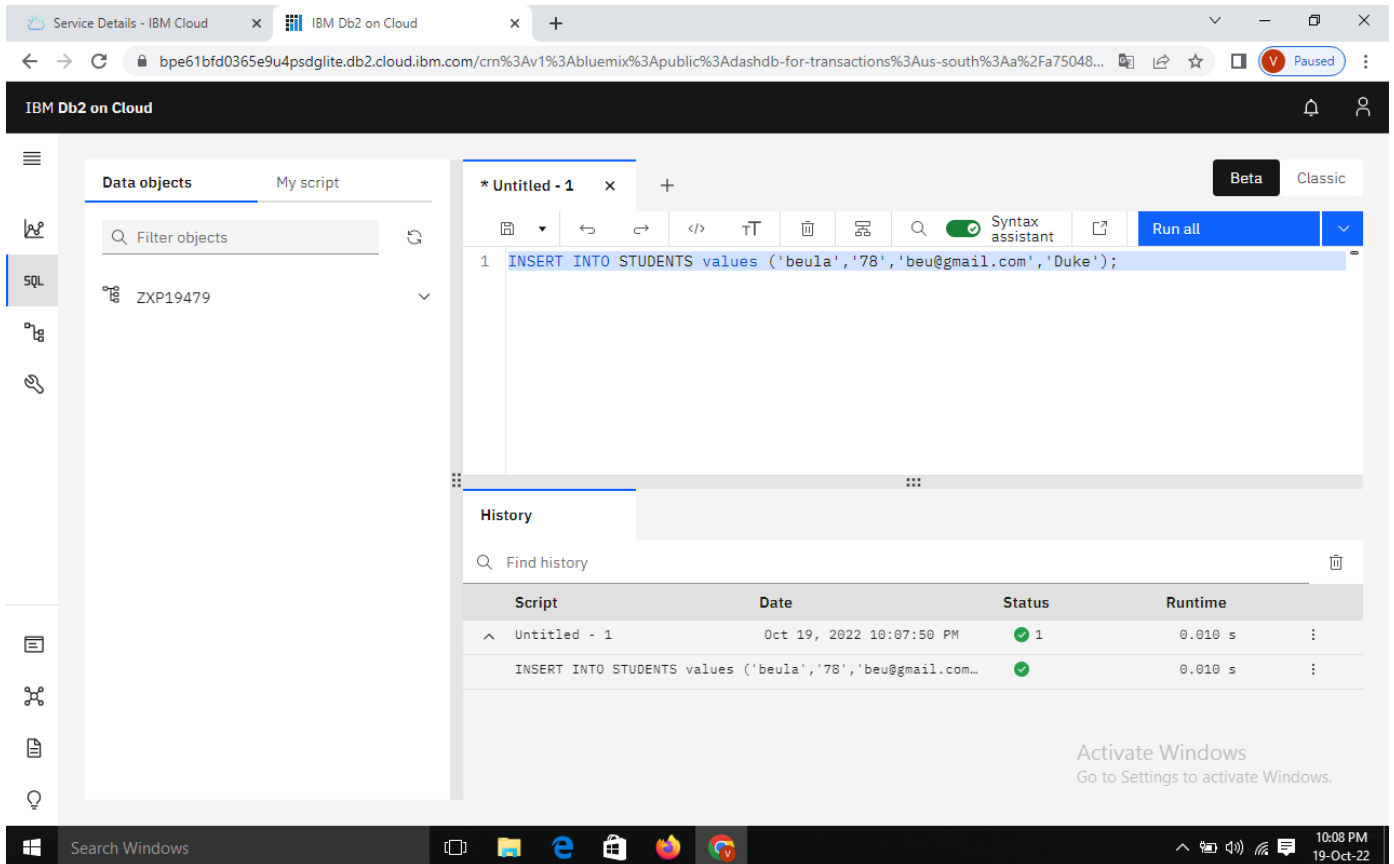
Name	Public access ⓘ	Location ⓘ	Storage class	Created
zonealertinng	Yes	jp-tok	Smart Tier	2022-11-09 9:08 PM

Activate Windows  
Go to Settings to activate Windows.

Search Windows

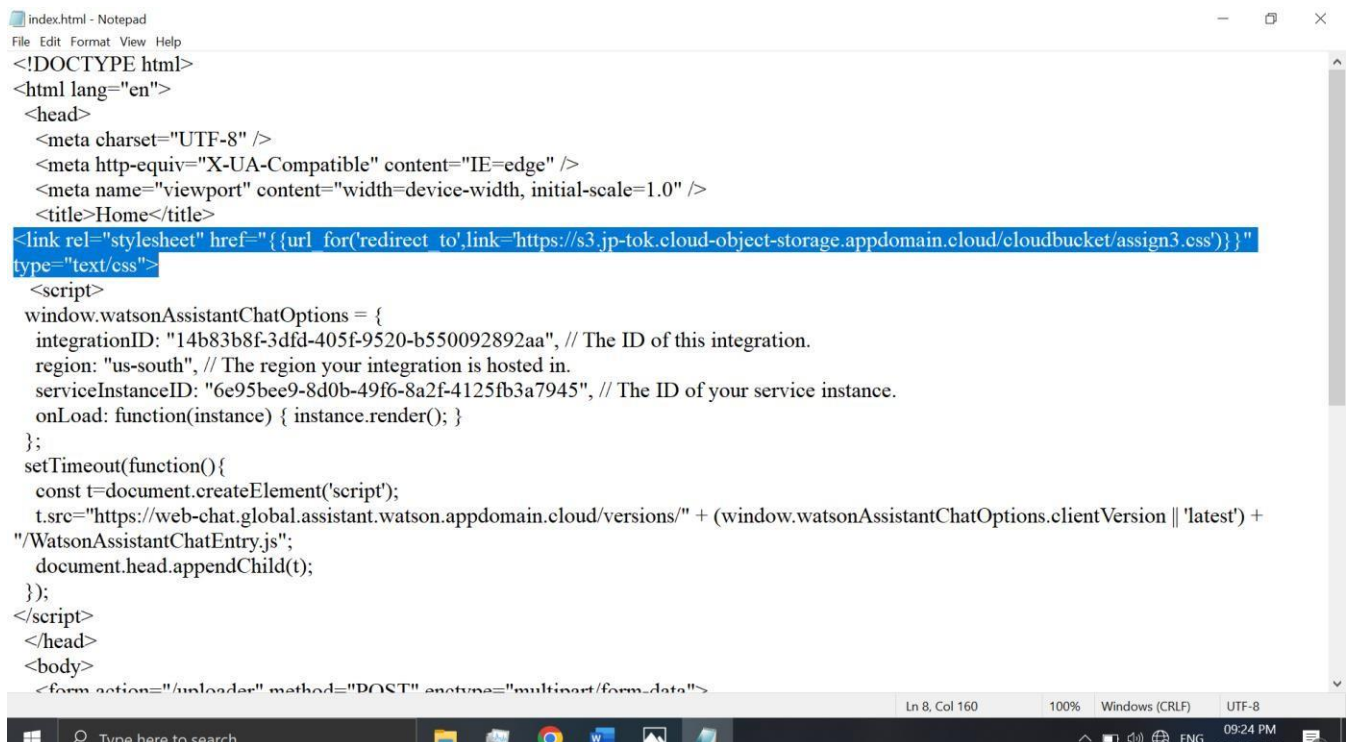
9:13 PM  
09-Nov-22

## 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');`. Below the editor, a 'History' table shows the execution of the script. The table has columns: Script, Date, Status, and Runtime. The first row shows 'Untitled - 1' executed on 'Oct 19, 2022 10:07:50 PM' with a status of '1' and a runtime of '0.010 s'. The second row shows the same script executed with a status of '✓' and a runtime of '0.010 s'. The bottom of the console shows a Windows taskbar with the time '10:08 PM 19-Oct-22'.

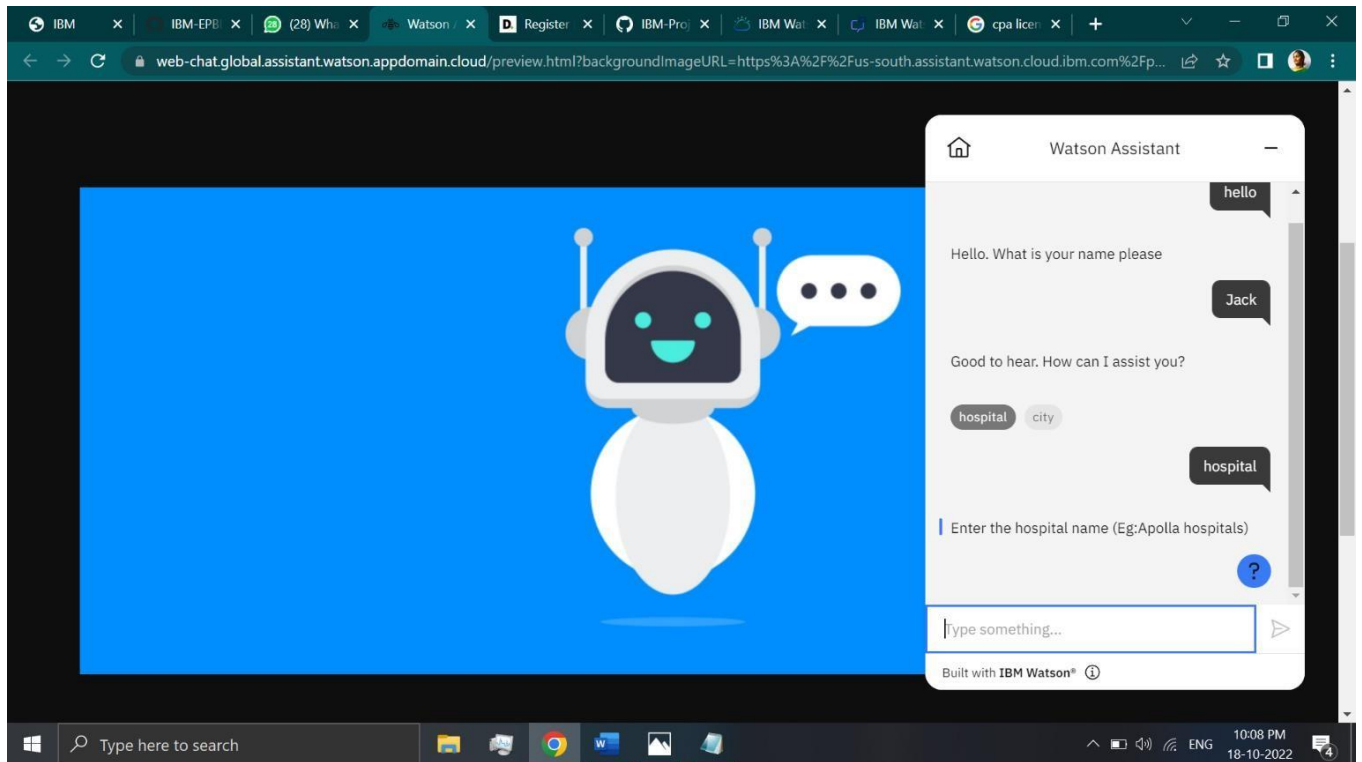
Script	Date	Status	Runtime
Untitled - 1	Oct 19, 2022 10:07:50 PM	1	0.010 s
INSERT INTO STUDENTS values ('beula','78','beu@gmail.com...		✓	0.010 s



The screenshot shows a Notepad++ window with the file 'index.html'. The HTML code includes a DOCTYPE declaration, a head section with meta tags for charset, http-equiv, viewport, and title, and a link to a CSS file. The link href is: `{url_for('redirect to', link='https://s3.jp-tok.cloud-object-storage.appdomain.cloud/cloudbucket/assign3.css')}`. The body section contains a script that initializes the Watson Assistant chat options and sets a timeout for loading the chat entry script. The script src is: `https://web-chat.global.assistant.watson.appdomain.cloud/versions/' + (window.watsonAssistantChatOptions.clientVersion || 'latest') + '/WatsonAssistantChatEntry.js'`. The bottom of the window shows a Windows taskbar with the time '09:24 PM 10-10-2022'.

```
<!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">
</head>
<body>
  <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>
</body>
<form action="/uploader" method="POST" enctype="multipart/form-data">
```

### 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&region=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 the IBM logo, a search bar, and several tabs: 'Your Platform', 'WhatsApp', 'Project Plan', 'IBM', 'Resource list', 'IBM Watson', and a plus icon. Below the navigation bar, the main header shows 'IBM Watson Assistant Lite' with an 'Upgrade' button and a 'Fashion Bot' dropdown menu. A 'Learning center' link is also present. The main content area is divided into two panels. The left panel, titled 'Hello', shows a list of steps in a conversation flow. Step 10, 'Bed Availability', is highlighted with a blue border and contains the text 'Today 50 beds available. Thank you' and 'Action complete'. The right panel shows the 'Assistant says' section with the text 'Today 50 beds available. Thank you'. Below this, there is a 'Define customer response' dropdown menu and an 'And then' section with an 'End the action' dropdown menu. A 'Preview' button is located at the bottom right of the main content area. The bottom status bar shows the temperature '29°C Cloudy', system icons, and the 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".

Hello



4 == Appointment details

Step 8 is taken with conditions

f<sub>1</sub>

7 number 04257xxxx

If of this is true:

4. Here are th... is Parking facility

X

and Add condition +

New condition group +

Thank you.

Assistant says

B I @ % & | | | | | </>

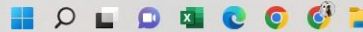
New step +

Yes, this hospital has parking facility

Preview



29°C  
Cloudy



ENG IN 11:16 19-10-2022

## 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']
```

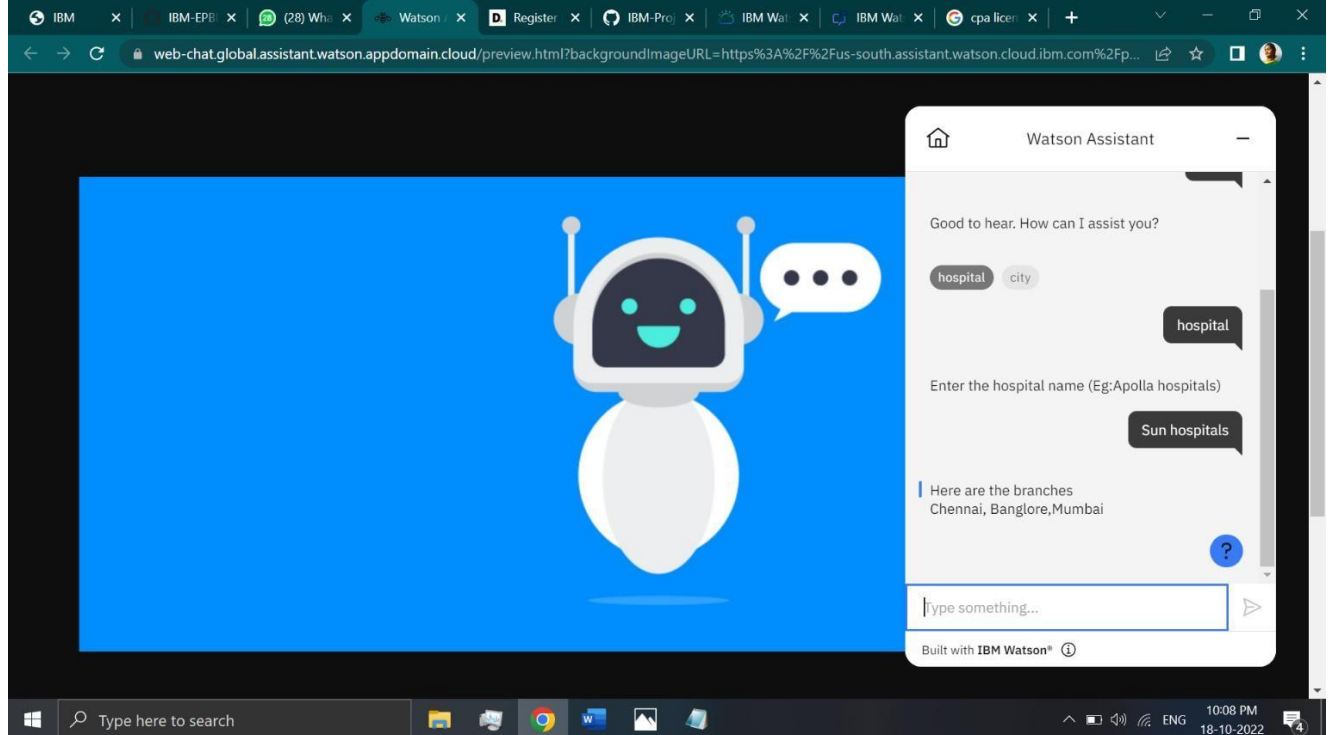
```
try:
```

```
    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)
```



