**Assignment -3**

**IBM Object Storage and IBM Watson Assistant**

|  |  |
| --- | --- |
| Assignment Date | 17 October 2022 |
| Student Name | Sowmiya Priya R S |
| Student Roll Number | 715519104051 |
| Maximum Marks | 2 Marks |

**Question 1:**

Create a Bucket in IBM object storage.

**Solution:**

**ibmCloudTest.py**

import ibm\_boto3

from ibm\_botocore.client import Config, ClientError

from ibm\_s3transfer.aspera.manager import AsperaTransferManager

from ibm\_s3transfer.aspera.manager import AsperaConfig

from flask import Flask, render\_template, url\_for, request, redirect

COS\_ENDPOINT="https://s3.jp-tok.cloud-object-storage.appdomain.cloud"

COS\_API\_KEY\_ID="wprEMAxyjHj5sPI959wL\_3HJczOWRbYn52XUuLrDSJON"

COS\_INSTANCE\_CRN="crn:v1:bluemix:public:cloud-object-storage:global:a/602bcedcf9224f7b8c2e1aed60258292:b846251f-3216-44c8-b123-4e13e3571cda::"

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

)

ms\_transfer\_config = AsperaConfig(multi\_session=2, multi\_session\_threshold\_mb=100)

transfer\_manager = AsperaTransferManager(client=client, transfer\_config=ms\_transfer\_config)

app=Flask(\_\_name\_\_)

bucket\_name = "flask-application"

download\_filename = "E:\IMS\static\css\Styles.css"

object\_name = "Styles.css"

with AsperaTransferManager(client) as transfer\_manager:

future = transfer\_manager.download(bucket\_name, object\_name, download\_filename)

future.result()

def get\_item(bucket\_name, item\_name):

print("Retrieving item from bucket: {0}, key: {1}".format(bucket\_name, item\_name))

try:

file = cos.Object(bucket\_name, item\_name).get()

print("File Contents: {0}".format(file["Body"].read()))

except ClientError as be:

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

except Exception as e:

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

def get\_bucket\_contents(bucket\_name):

print("Retrieving bucket contents from: {0}".format(bucket\_name))

try:

files = cos.Bucket(bucket\_name).objects.all()

files\_names = []

print(files)

for file in files:

files\_names.append(file.key)

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

return files\_names

except ClientError as be:

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

except Exception as e:

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

@app.route('/')

def index():

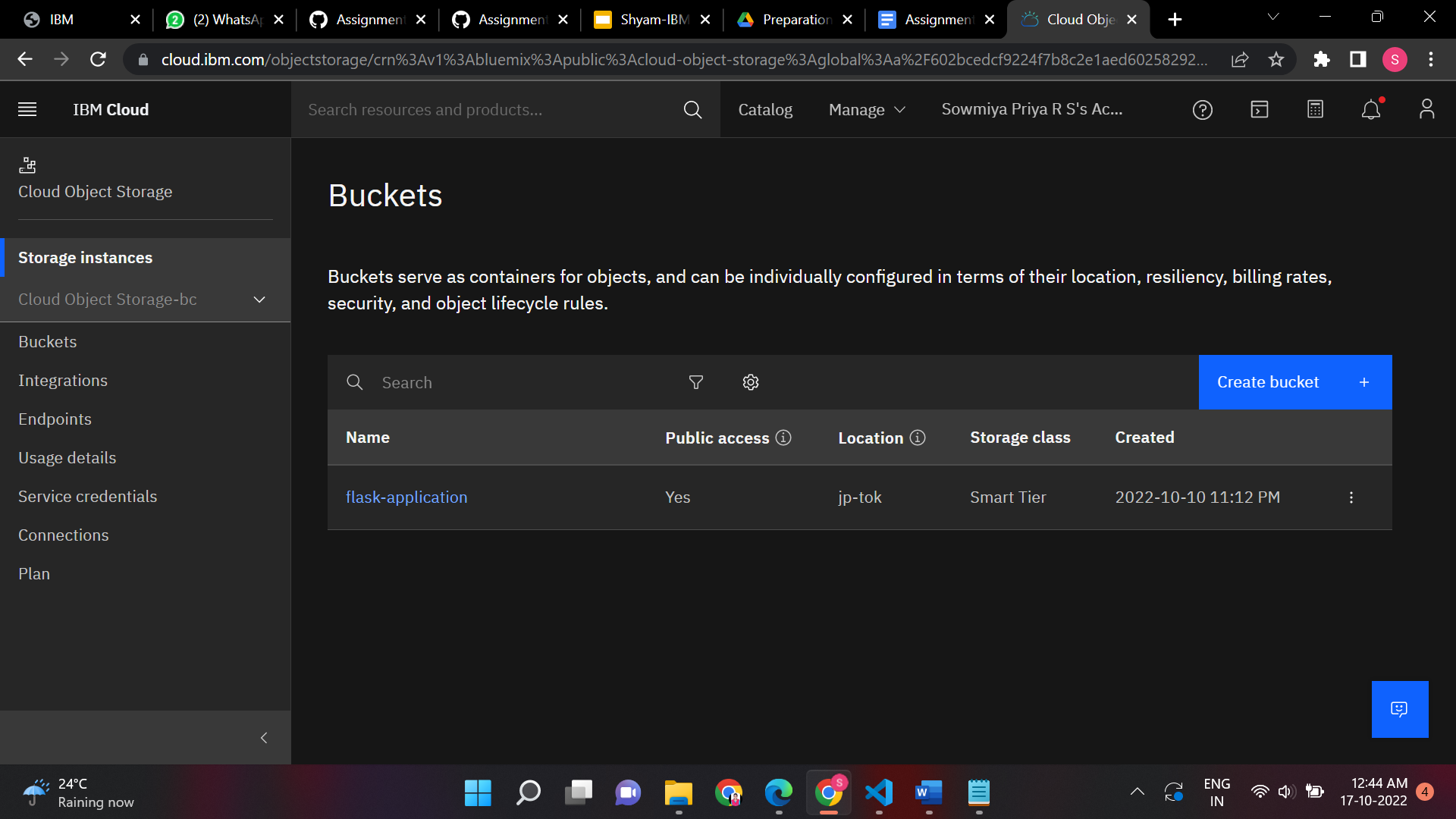
files = get\_bucket\_contents('flask-application')

return render\_template('index.html', files = files)

if \_\_name\_\_=='\_\_main\_\_':

app.run(debug=True)

**Output:**



**Question 2:**

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

**Solution:**

**index.html**

<h1>IBM Cloud Storage and IBM Watson Assistant</h1>

<br><hr>

{% block head %}

<!-- CSS -->

<link rel="stylesheet" href="{{ url\_for('static',filename='css/Styles.css') }}" />

{% endblock %}

<script>

window.watsonAssistantChatOptions = {

integrationID: "8abe6aef-f2bb-4376-be93-7fda1857ec36", // The ID of this integration.

region: "jp-tok", // The region your integration is hosted in.

serviceInstanceID: "440aaee1-fbb4-4c2b-8604-cb5d605fa157", // 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>

<!doctype html>

<html>

<body>

{% for row in files %}

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

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

<img src="https://flask-application.s3.jp-tok.cloud-object-storage.appdomain.cloud/{{row}}" width="150px"></td>

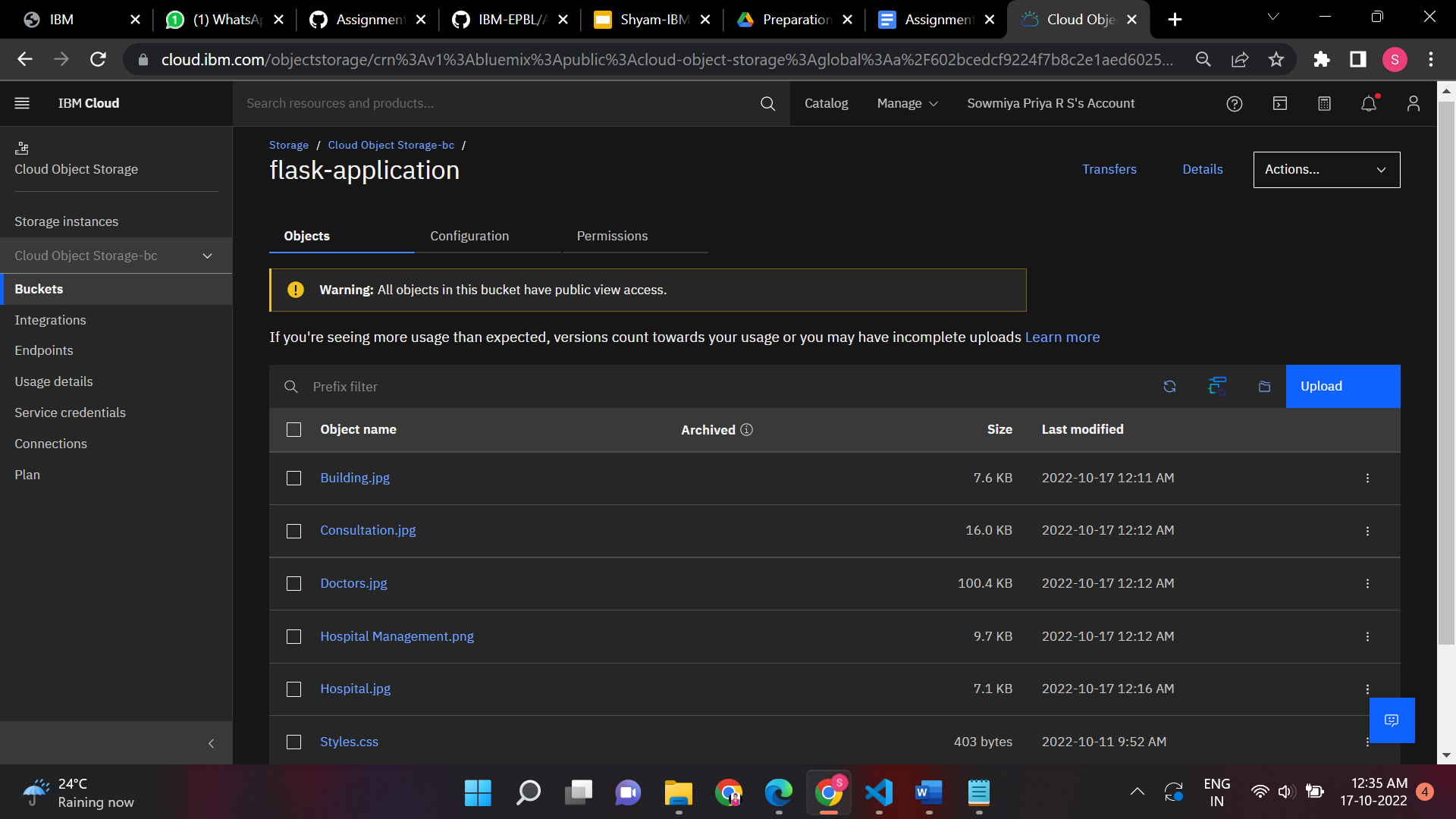
</div>

{% endfor %}

</body>

</html>

**Output:**



**Question 3:**

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

**Solution:**

**Styles.css**

body {

background-color: rgb(190, 229, 247);

text-align: center;

font-style: italic ;

}

.outercont

{

height: 100vh;

width: 100vw;

display: flex;

flex-direction: column;

justify-content: space-around;

align-items: center;

}

p {

font-size: 20px;

}

h1 {

text-align: center;

font-style: normal;

}

h2 {

text-align: center;

}

.cont {

width: 700px;

}

.leftcont {

font-size: 17px;

}

.rightcont {

font-size: 17px;

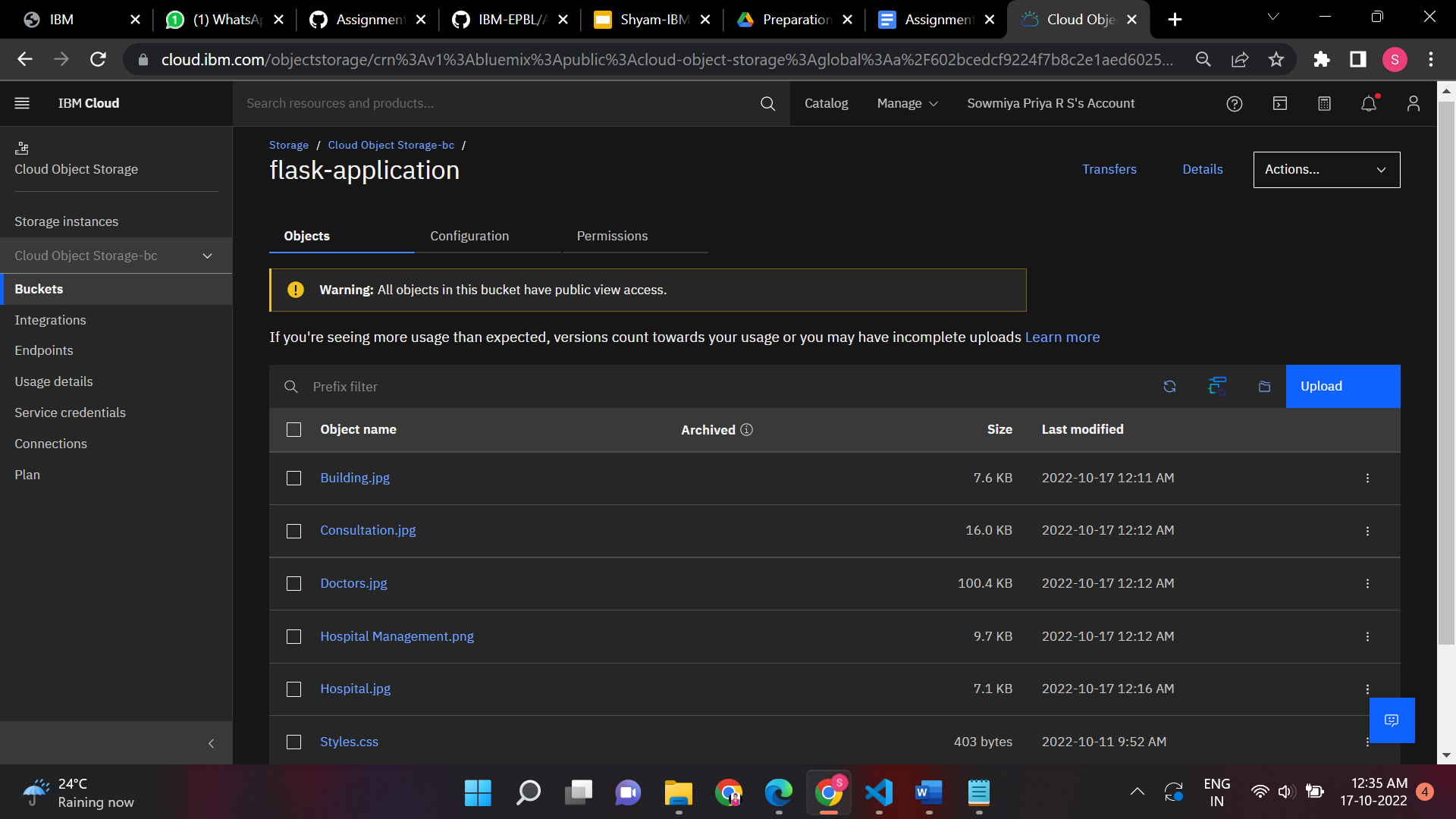
}

.othercont {

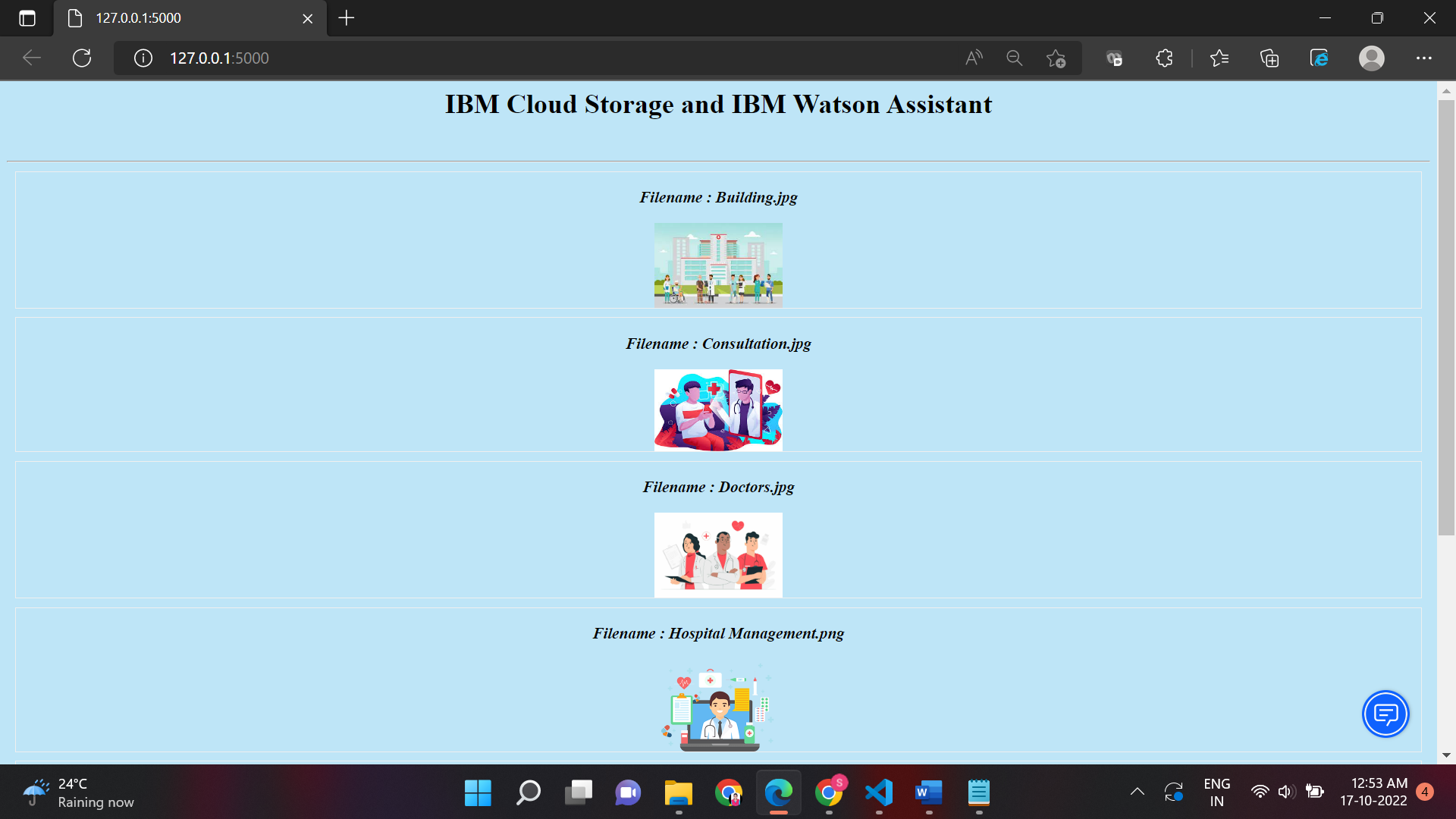
display: flex;

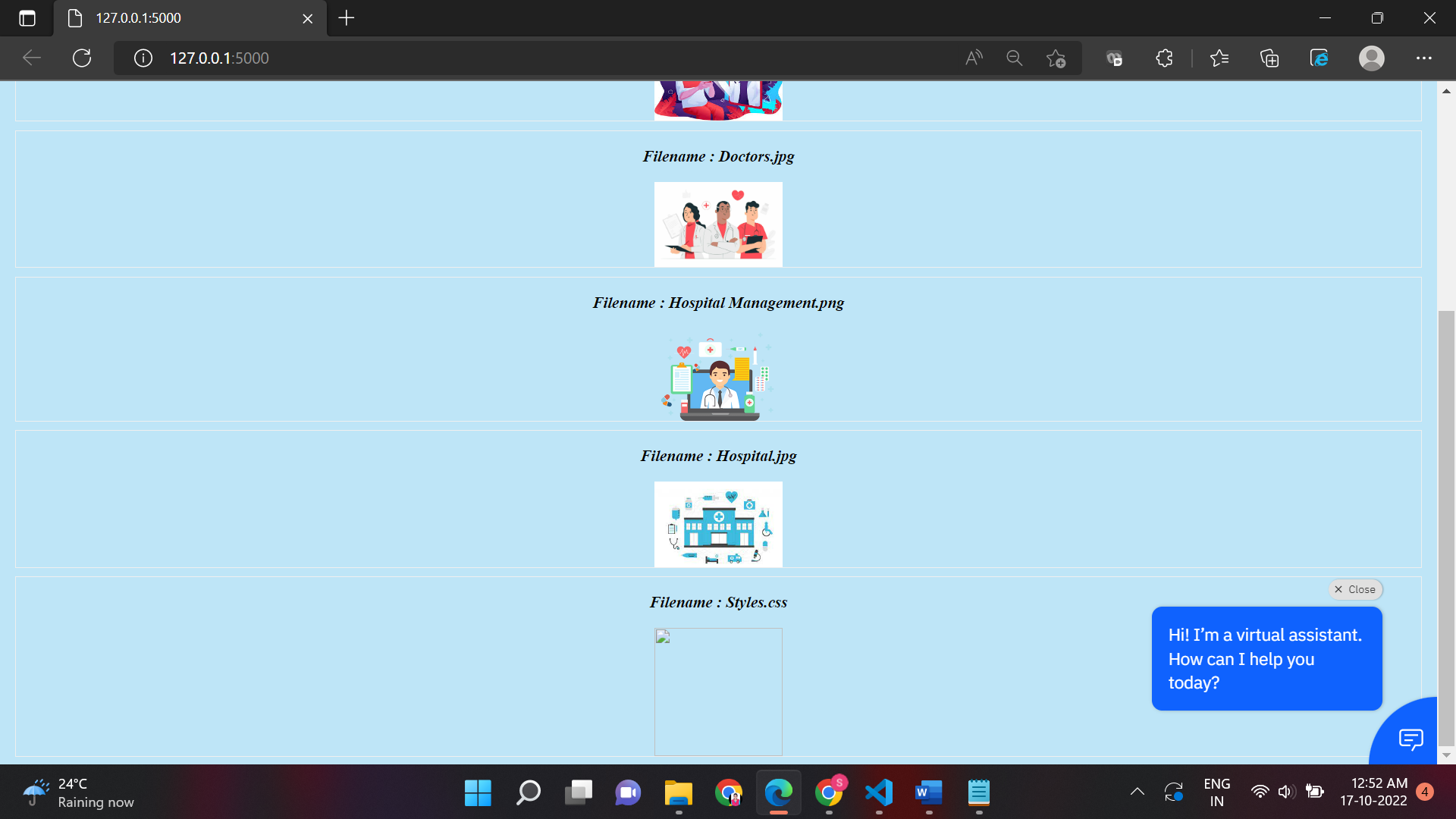
}

**Output:**



**Web Page:**





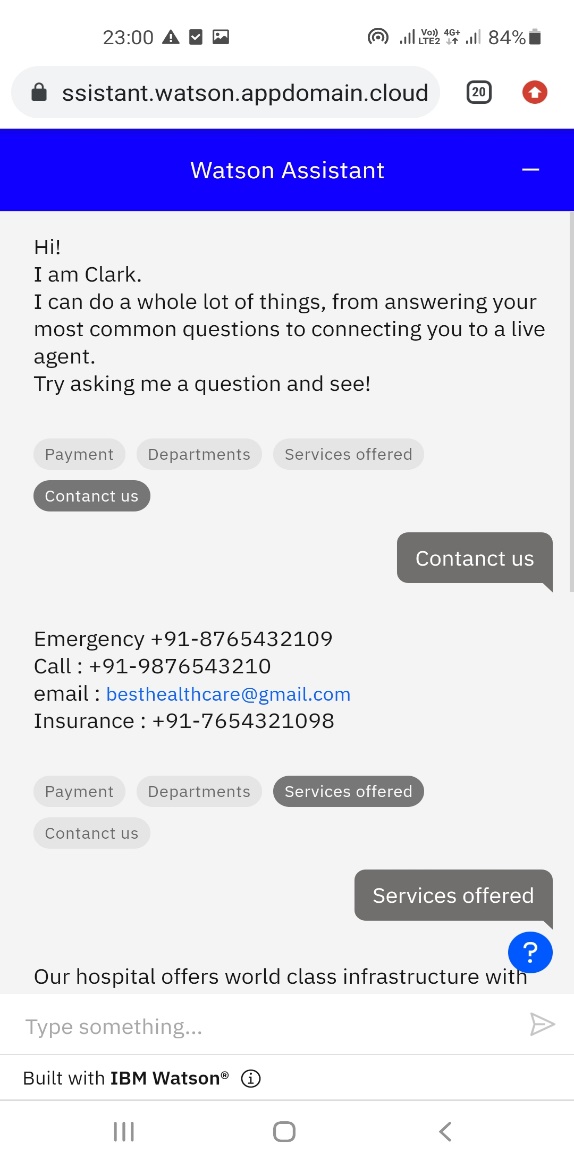
**Question-4:**

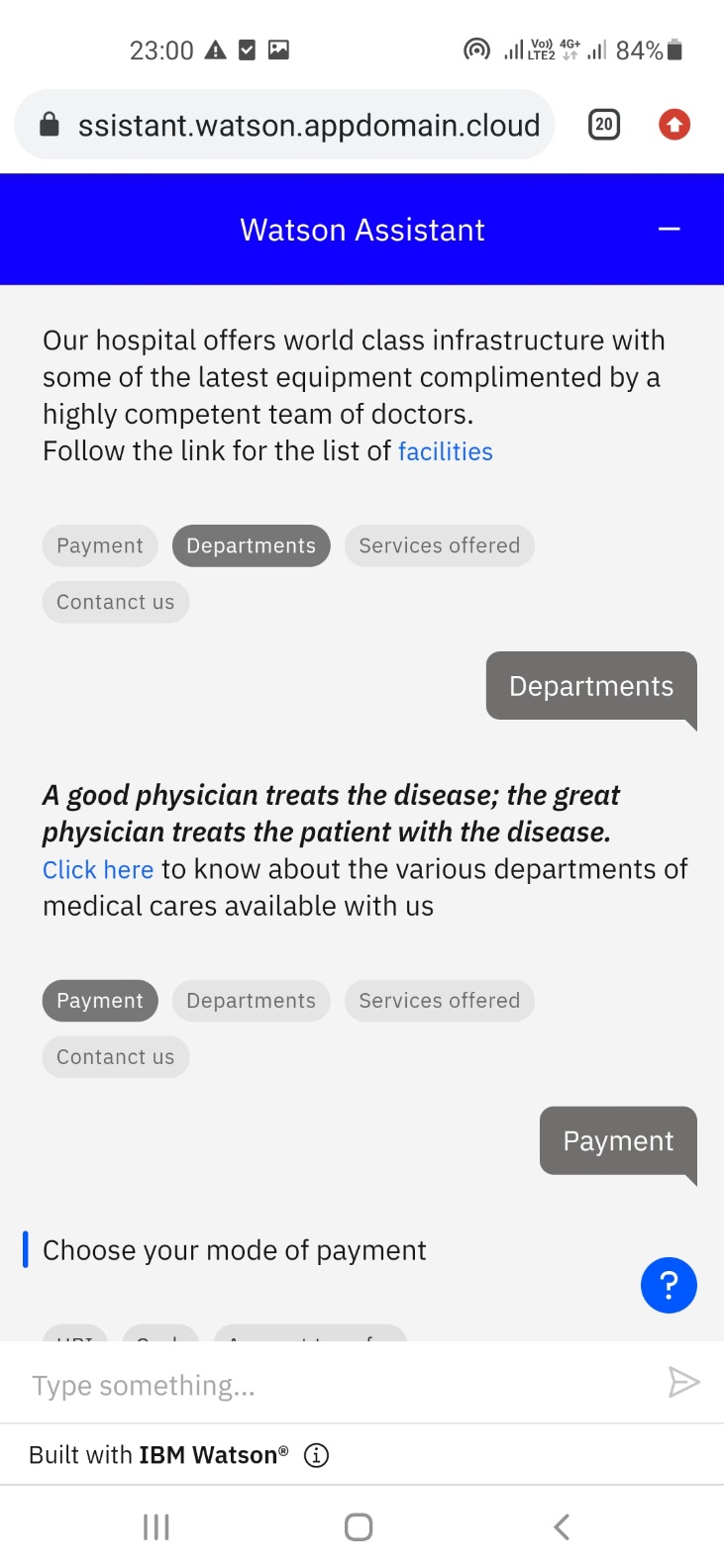
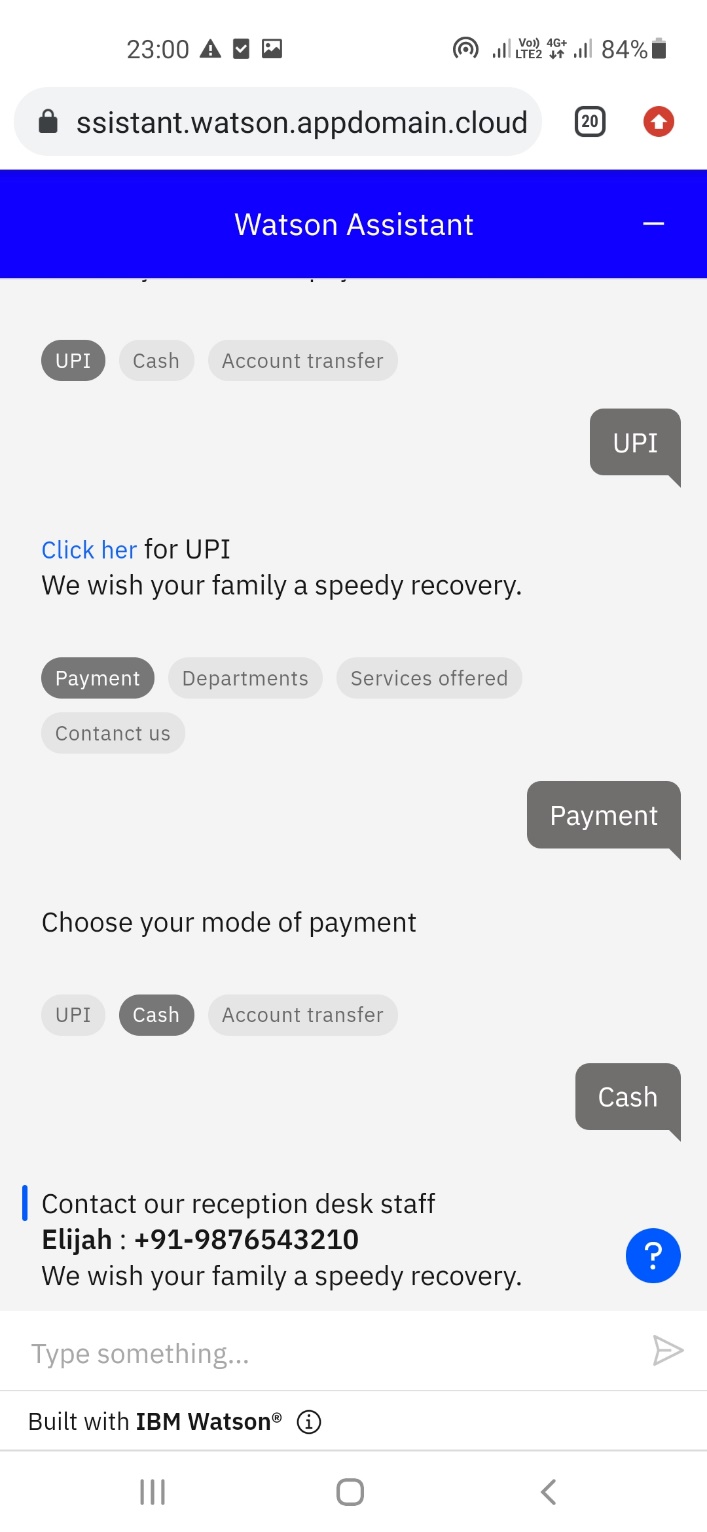
Design a chatbot using IBM Watson assistant for hospital. Ex: User comes with query to know the branches for that hospital in your city. Submit the web URL of that chat bot as a assignment.

**Solution:**

*https://web-chat.global.assistant.watson.appdomain.cloud/preview.html?backgroundImageURL=https%3A%2F%2Fjp-tok.assistant.watson.cloud.ibm.com%2Fpublic%2Fimages%2Fupx-440aaee1-fbb4-4c2b-8604-cb5d605fa157%3A%3Affa26b03-3295-42fd-a098-6723615c4e58&integrationID=a3e3b932-8ee4-43ee-9ccb-f3cfa9687188&region=jp-tok&serviceInstanceID=440aaee1-fbb4-4c2b-8604-cb5d605fa157*

**Output:**





**Question-5:**

5.Create Watson assistant service with 10 steps and use 3 conditions in it. Load that script in HTML page.

**Solution:**

<script>

window.watsonAssistantChatOptions = {

integrationID: "8abe6aef-f2bb-4376-be93-7fda1857ec36", // The ID of this integration.

region: "jp-tok", // The region your integration is hosted in.

serviceInstanceID: "440aaee1-fbb4-4c2b-8604-cb5d605fa157", // 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";

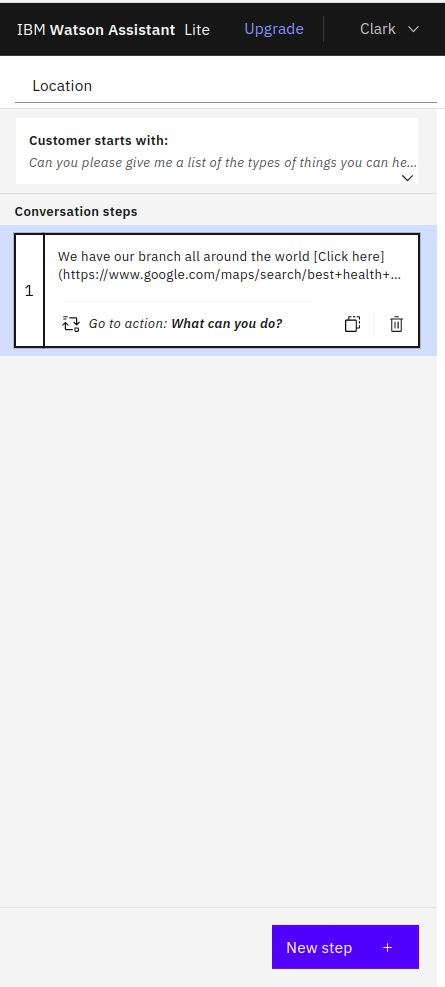
document.head.appendChild(t);

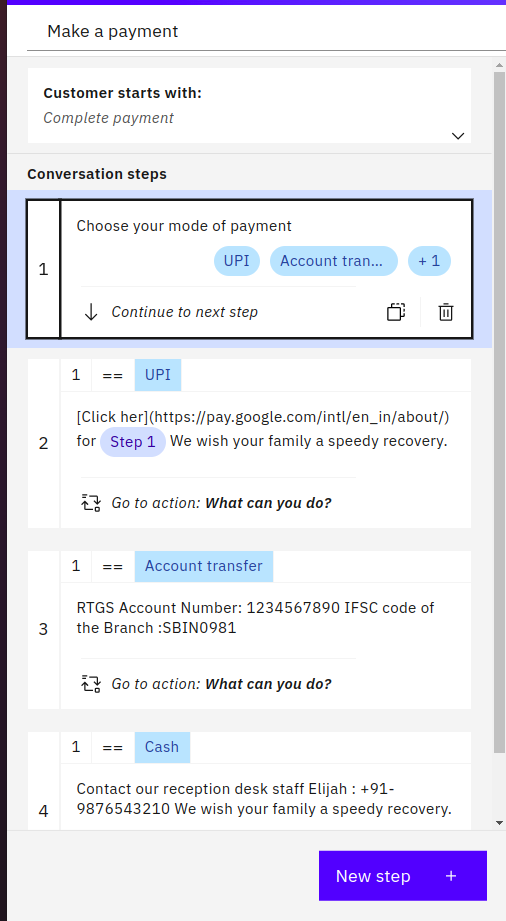
});

</script>

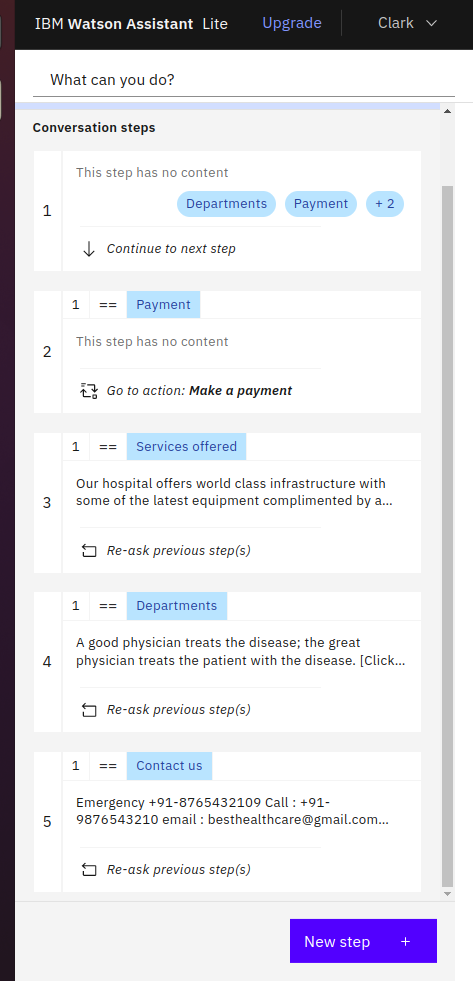
**Steps:**

Steps in payment Steps in location





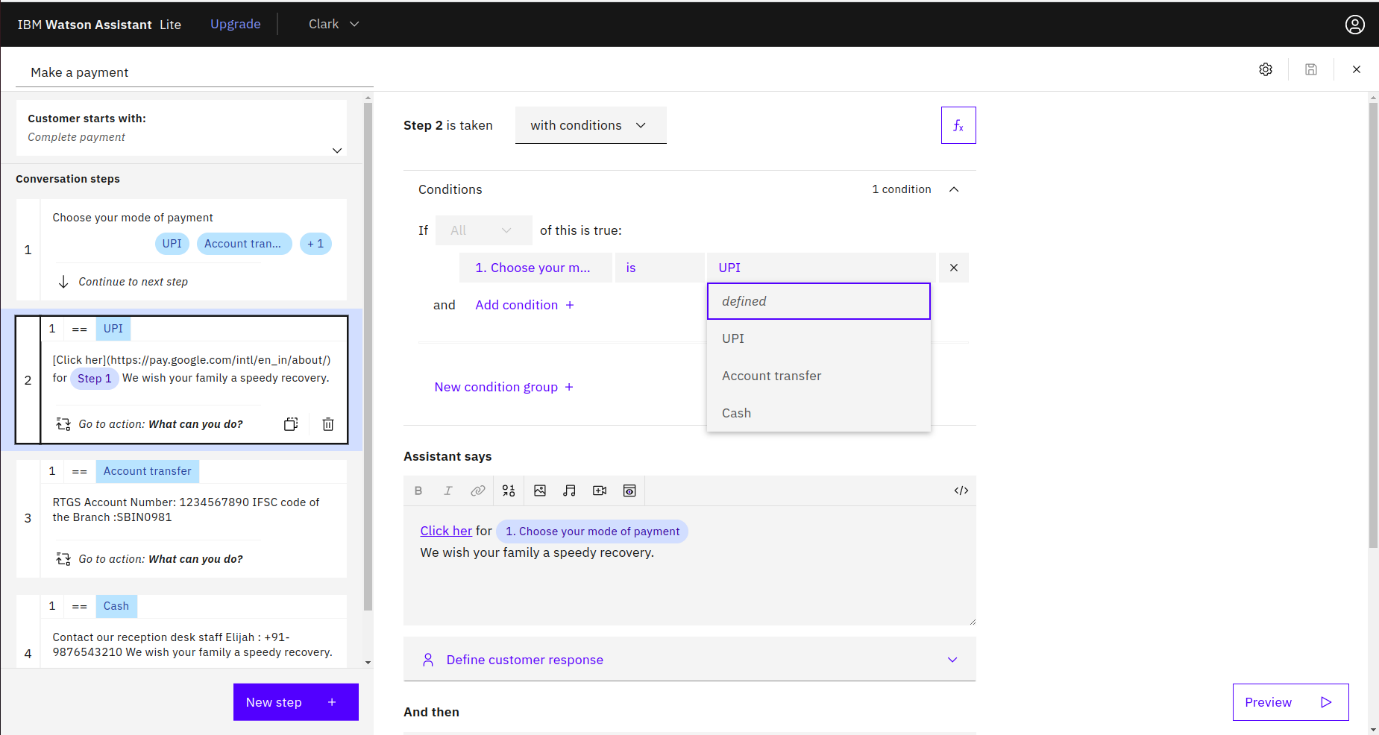
steps in What can you do



**Conditions:**

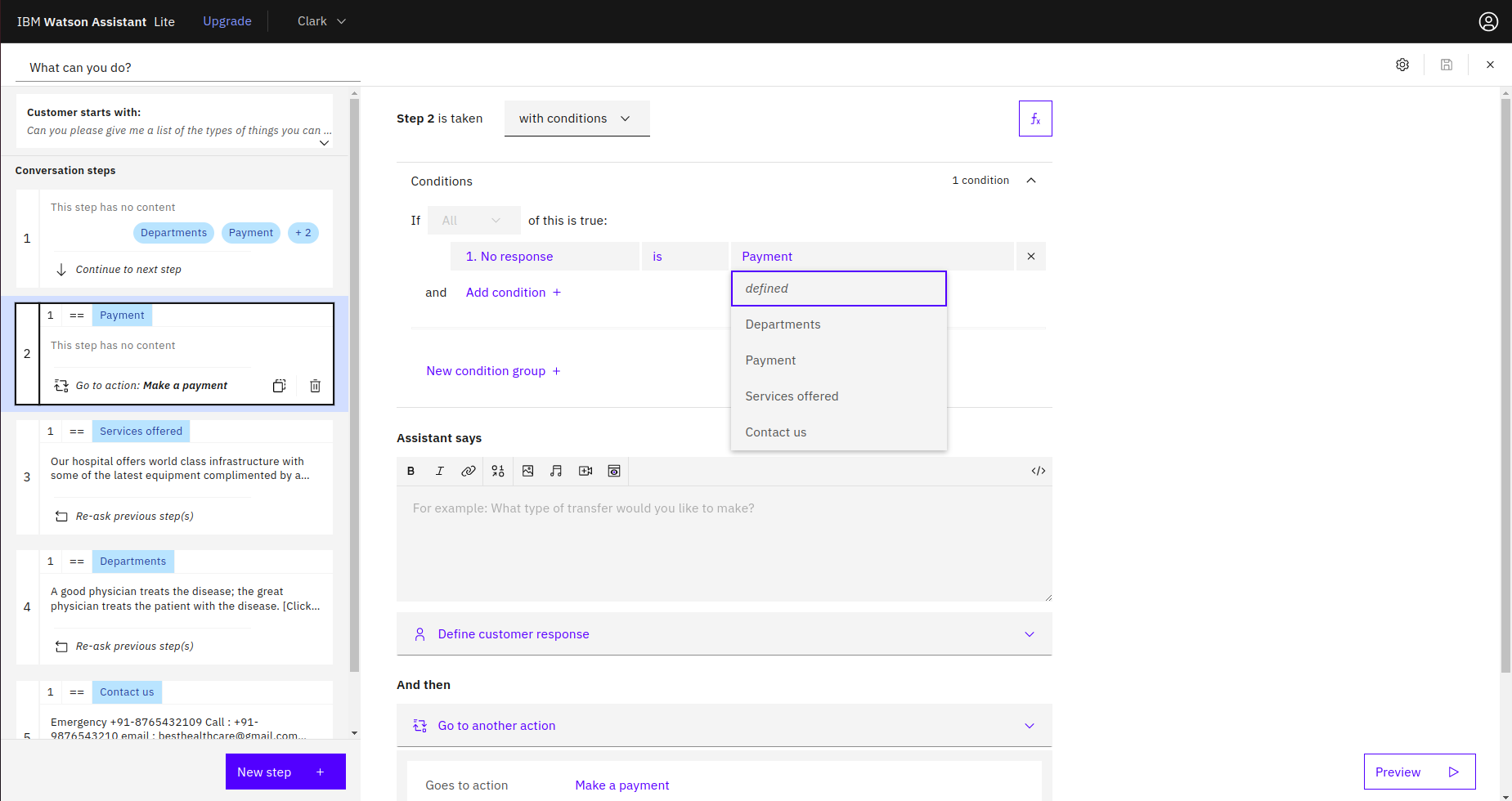
Conditions in payment

* UPI
* Account transfer
* Cash



Conditions in What can you do

* Departments
* Payment
* Services offered
* Contact us



**Output:**

