

IMAGE RECOGNITION WITH IBM CLOUD VISUAL RECOGNITION

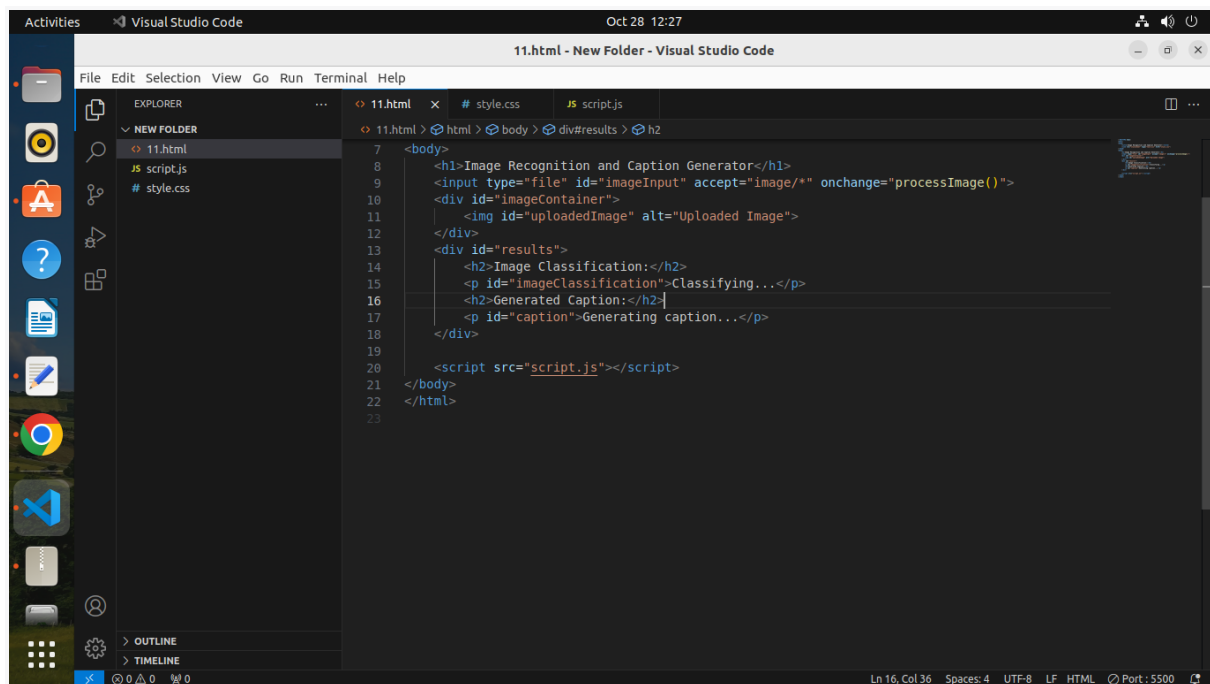
Problem statement :

Continue building the image recognition system by integrating IBM Cloud Visual Recognition and AI-generated captions. Implement the image classification process using the IBM Cloud Visual Recognition API. Use natural language generation to create captions for the recognized images.

Problem solution :

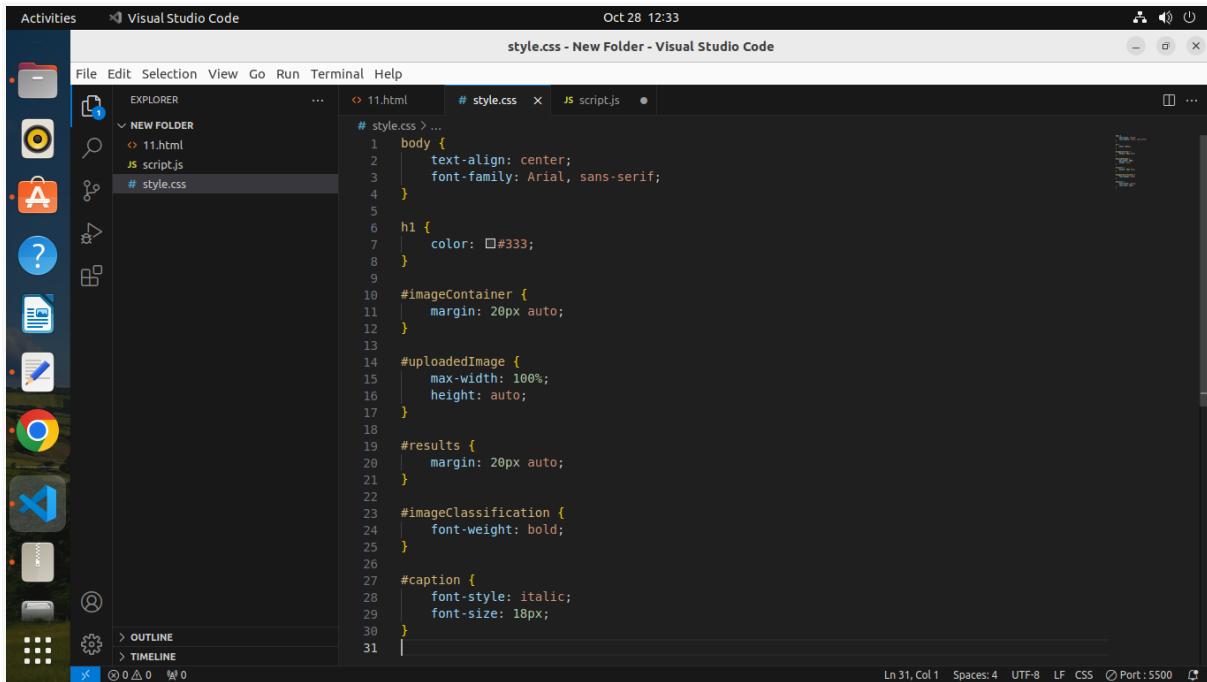
To integrate IBM Cloud Visual Recognition and AI-generated captions into a web application, you can use HTML,CSS, and Java Script

HTML code :

A screenshot of the Visual Studio Code editor interface. The top bar shows 'Visual Studio Code' and the date 'Oct 28 12:27'. The main window title is '11.html - New Folder - Visual Studio Code'. The Explorer sidebar on the left shows a 'NEW FOLDER' with files '11.html', 'script.js', and 'style.css'. The main editor area displays the HTML code for '11.html'. The code includes a file input, an image container, and a results section with placeholders for image classification and caption generation. The status bar at the bottom indicates 'Ln 16, Col 36', 'Spaces: 4', 'UTF-8', 'LF', 'HTML', and 'Port: 5500'.

```
7 <body>
8 <h1>Image Recognition and Caption Generator</h1>
9 <input type="file" id="imageInput" accept="image/*" onchange="processImage()">
10 <div id="imageContainer">
11 <img id="uploadedImage" alt="Uploaded Image">
12 </div>
13 <div id="results">
14 <h2>Image Classification:</h2>
15 <p id="imageClassification">Classifying...</p>
16 <h2>Generated Caption:</h2>
17 <p id="caption">Generating caption...</p>
18 </div>
19
20 <script src="script.js"></script>
21 </body>
22 </html>
23
```

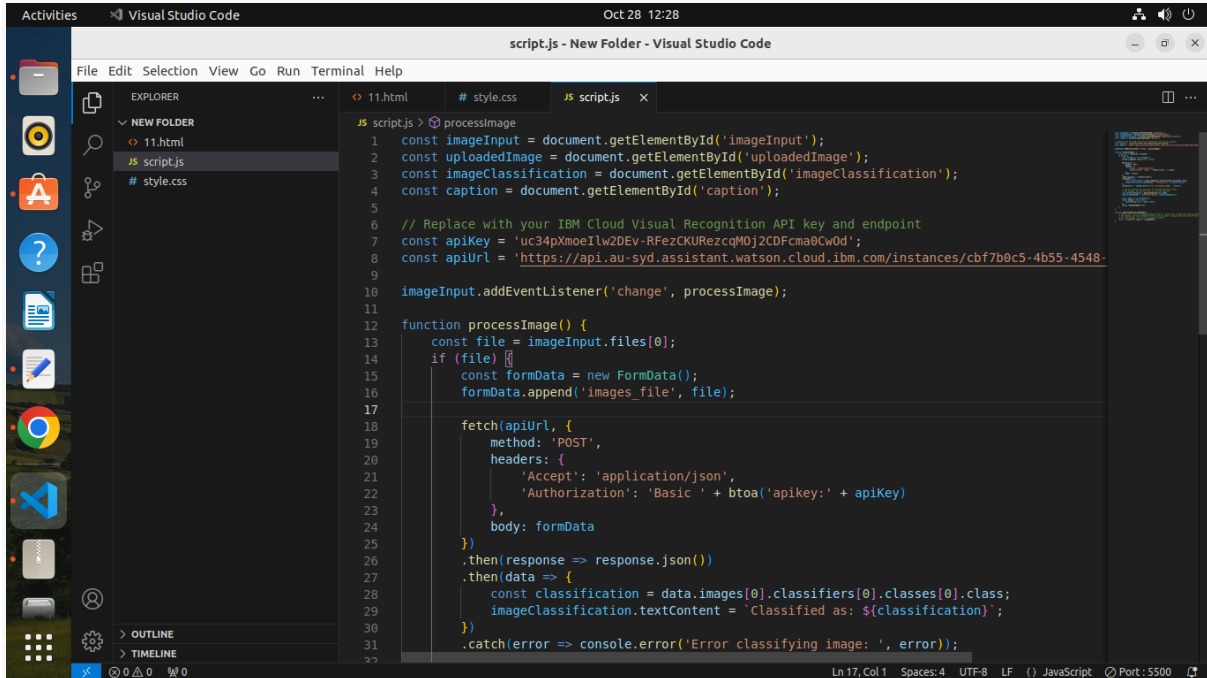
CSS code :



A screenshot of the Visual Studio Code editor interface. The title bar shows 'style.css - New Folder - Visual Studio Code' and the date 'Oct 28 12:33'. The Explorer sidebar on the left shows a file structure with '11.html', 'script.js', and 'style.css'. The main editor area displays the content of 'style.css', which includes styles for the body, h1, imageContainer, uploadedImage, results, imageClassification, and caption. The status bar at the bottom indicates 'Ln 31, Col 1', 'Spaces: 4', 'UTF-8', 'LF', 'CSS', and 'Port: 5500'.

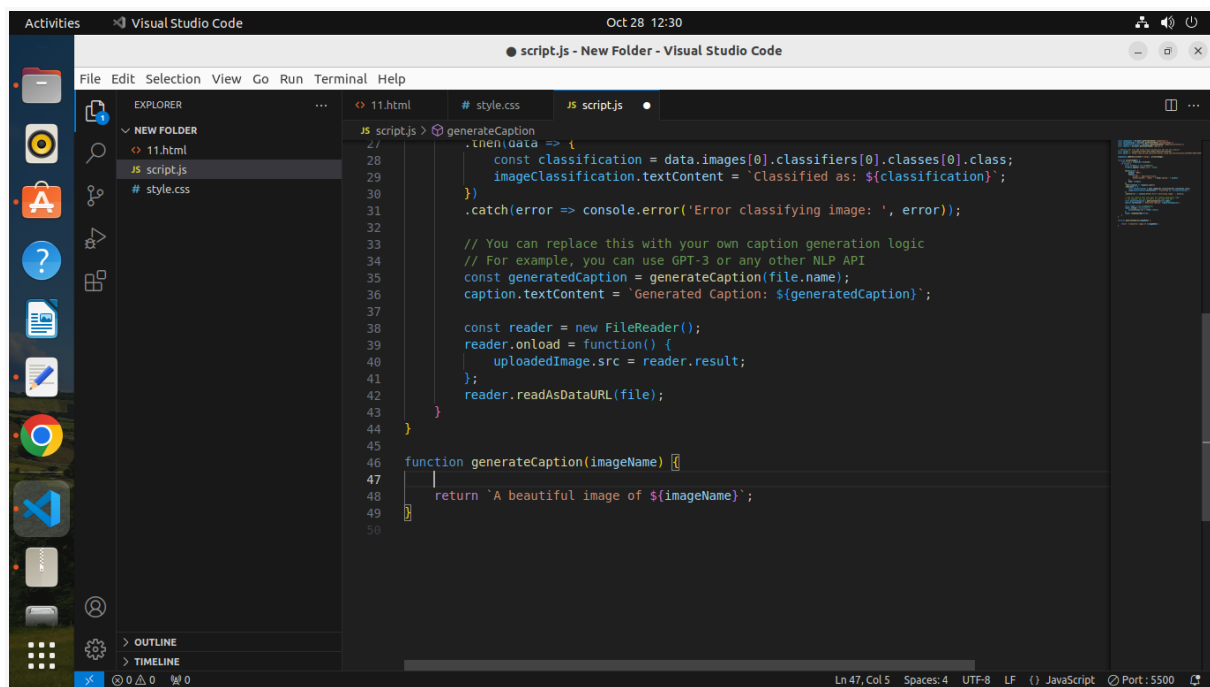
```
# style.css > ...
1  body {
2      text-align: center;
3      font-family: Arial, sans-serif;
4  }
5
6  h1 {
7      color: #333;
8  }
9
10 #imageContainer {
11     margin: 20px auto;
12 }
13
14 #uploadedImage {
15     max-width: 100%;
16     height: auto;
17 }
18
19 #results {
20     margin: 20px auto;
21 }
22
23 #imageClassification {
24     font-weight: bold;
25 }
26
27 #caption {
28     font-style: italic;
29     font-size: 18px;
30 }
31
```

Script.js :



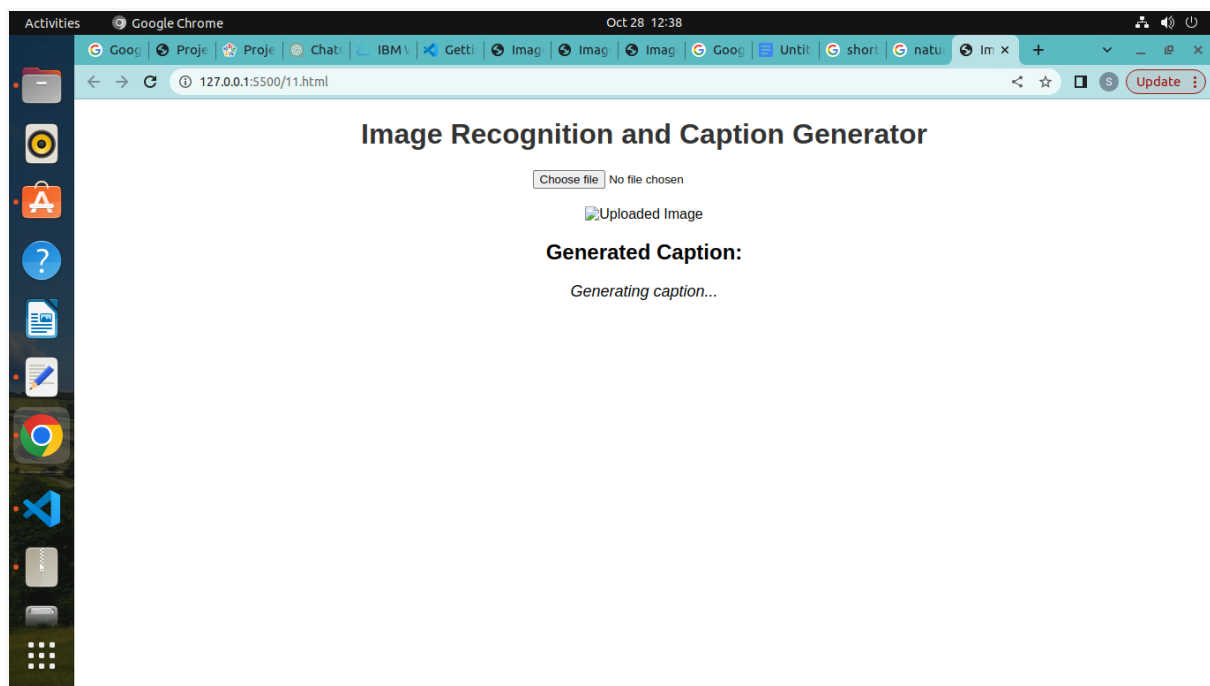
A screenshot of the Visual Studio Code editor interface. The title bar shows 'script.js - New Folder - Visual Studio Code' and the date 'Oct 28 12:28'. The Explorer sidebar on the left shows a file structure with '11.html', 'script.js', and 'style.css'. The main editor area displays the content of 'script.js', which includes a processImage function that uses the IBM Cloud Visual Recognition API to classify an image. The status bar at the bottom indicates 'Ln 17, Col 1', 'Spaces: 4', 'UTF-8', 'LF', 'JavaScript', and 'Port: 5500'.

```
script.js > processImage
1  const imageInput = document.getElementById('imageInput');
2  const uploadedImage = document.getElementById('uploadedImage');
3  const imageClassification = document.getElementById('imageClassification');
4  const caption = document.getElementById('caption');
5
6  // Replace with your IBM Cloud Visual Recognition API key and endpoint
7  const apiKey = 'uc34pXmoeIlw2DEv-RFczCKURzCqMOj2CDFcma0Cw0d';
8  const apiUrl = 'https://api.au-syd.assistant.watson.cloud.ibm.com/instances/cbf7b0c5-4b55-4548-';
9
10 imageInput.addEventListener('change', processImage);
11
12 function processImage() {
13     const file = imageInput.files[0];
14     if (file) {
15         const formData = new FormData();
16         formData.append('images_file', file);
17
18         fetch(apiUrl, {
19             method: 'POST',
20             headers: {
21                 'Accept': 'application/json',
22                 'Authorization': 'Basic ' + btoa('apikey:' + apiKey)
23             },
24             body: formData
25         })
26         .then(response => response.json())
27         .then(data => {
28             const classification = data.images[0].classifiers[0].classes[0].class;
29             imageClassification.textContent = `Classified as: ${classification}`;
30         })
31         .catch(error => console.error('Error classifying image: ', error));
32     }
33 }
```

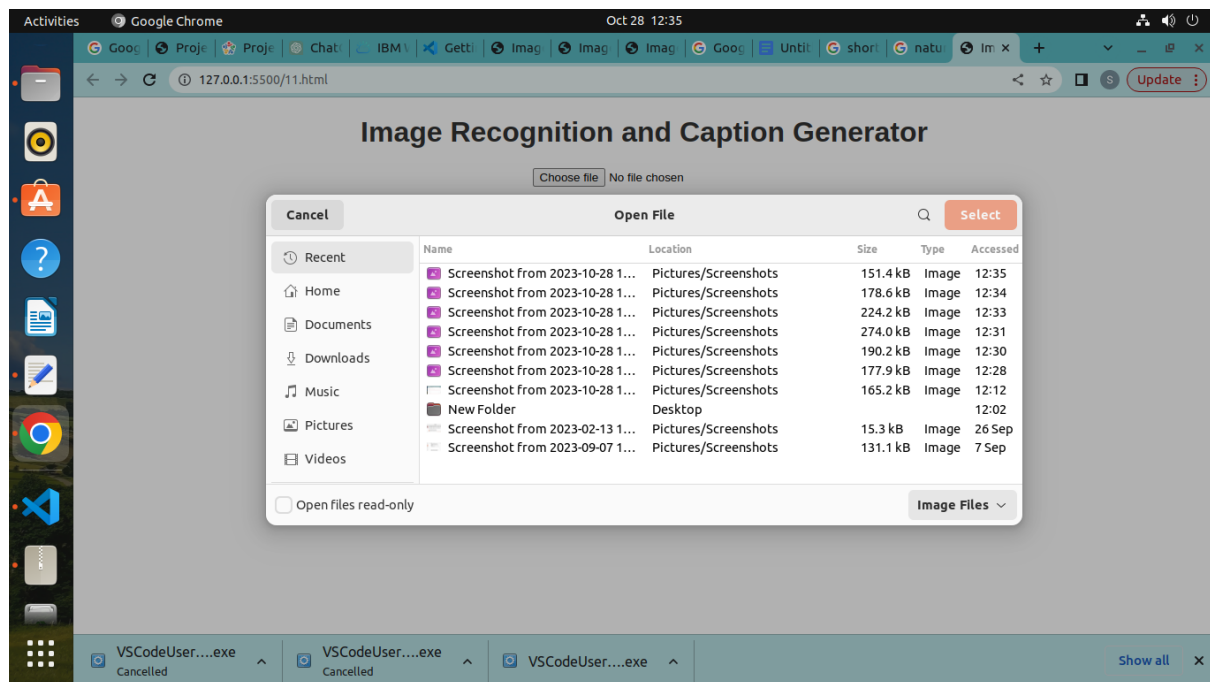


```
27 // generateCaption
28 .then(data => {
29     const classification = data.images[0].classifiers[0].classes[0].class;
30     imageClassification.textContent = `Classified as: ${classification}`;
31 })
32 .catch(error => console.error('Error classifying image: ', error));
33
34 // You can replace this with your own caption generation logic
35 // For example, you can use GPT-3 or any other NLP API
36 const generatedCaption = generateCaption(file.name);
37 caption.textContent = `Generated Caption: ${generatedCaption}`;
38
39 const reader = new FileReader();
40 reader.onload = function() {
41     uploadedImage.src = reader.result;
42 };
43 reader.readAsDataURL(file);
44
45
46 function generateCaption(imageName) {
47     return `A beautiful image of ${imageName}`;
48 }
49
50
```

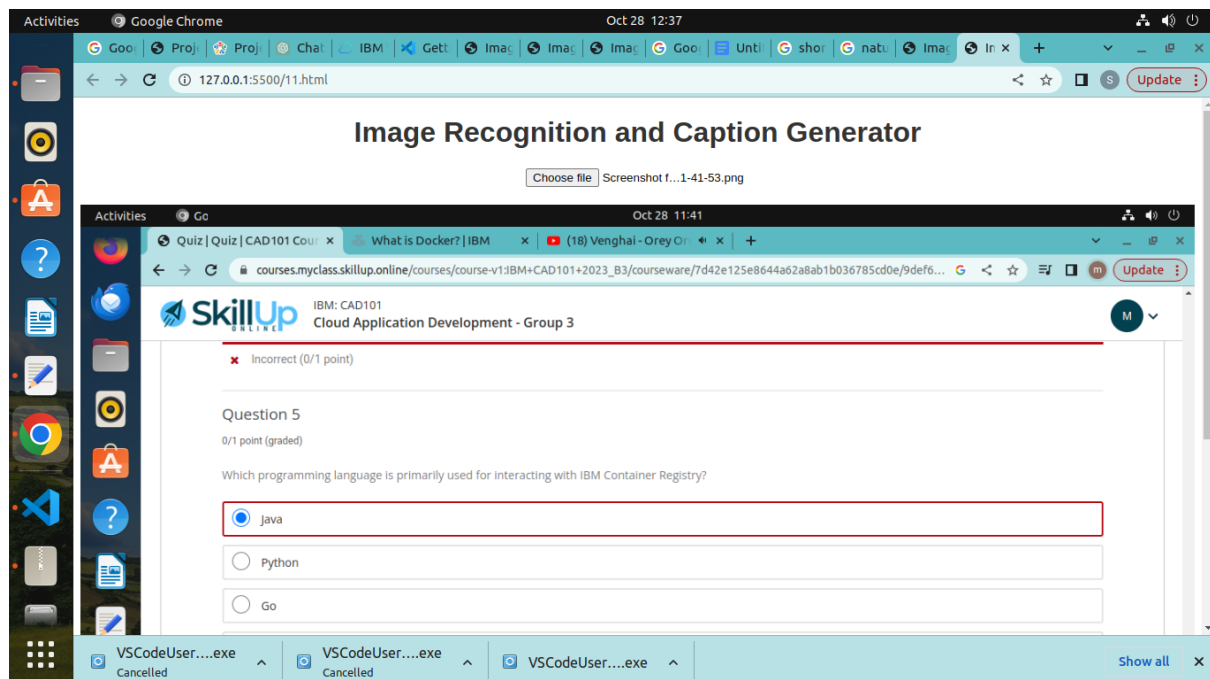
Output :

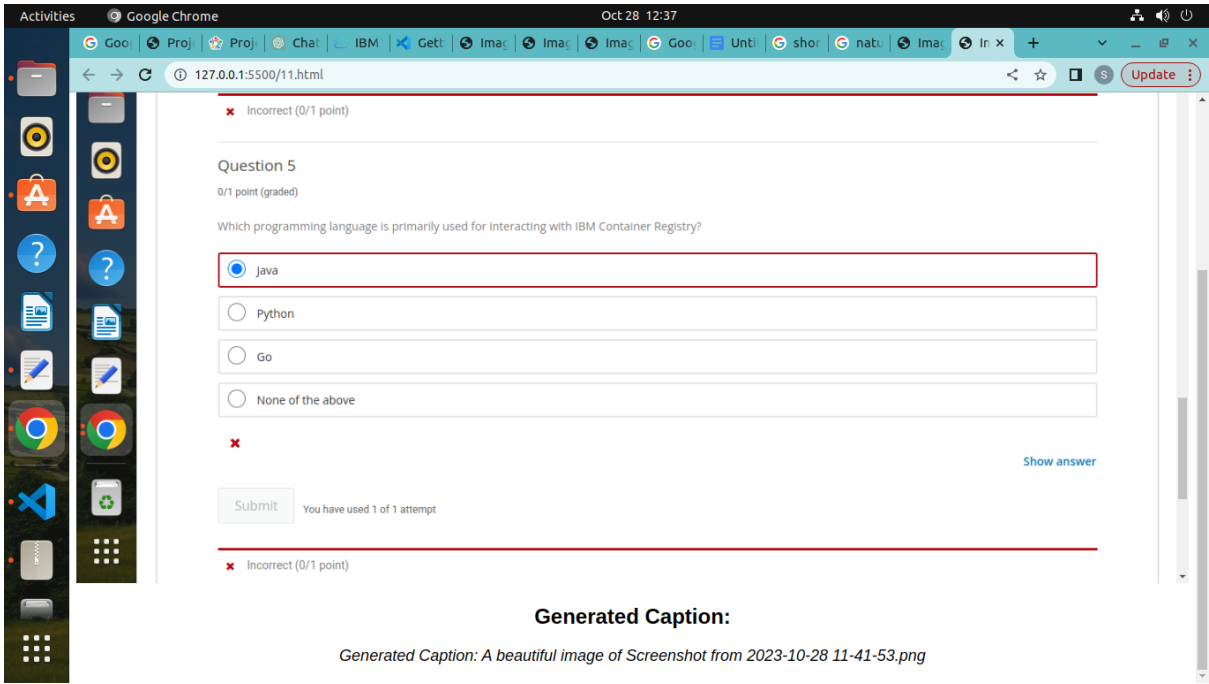


Choosing an image :



Output :





Generated Caption:

Generated Caption: A beautiful image of Screenshot from 2023-10-28 11-41-53.png