Image Recognition with IBM Cloud Visual Recognition

Phase 3

Development Part 1

Project Title : Facial-emotion-recognition-ai

Introduction:

- In this phase of our image recognition project, we will be focusing on creating the user interface (UI) that allows users to interact with our AI-powered image recognition system.
- We will utilize HTML, CSS, and JavaScript to craft a visually appealing and user-friendly frontend that enables users to upload images and view the captions generated by our AI model.

Project Context:

- Our goal is to build a simple and intuitive web application that makes image recognition accessible to users.

- By leveraging the power of artificial intelligence, our system will provide descriptive captions for images, enhancing accessibility and understanding.

Part 1 Objectives:

1.HTML Structure:

We'll create an HTML structure that defines the layout of our web page. This structure will include components for image upload, caption display, and user interactions.

2.CSS Styling:

CSS will be used to style our interface, ensuring it's visually appealing, responsive, and aligns with our project's branding and usability standards.

3.JavaScript Functionality:

JavaScript will add interactivity to the UI. It will enable us to handle user actions, such as uploading images, and facilitate communication with the backend for image processing.

Development Part 1:

HTML code:

```
<!doctype html>
```

```html
<html lang="fr">
<head>
    <title>Emotion Recognition</title>
    <link rel="stylesheet" href="../staticFiles/index.css" type="text/css">
</head>

<body>
<h1>Emotion Recognition</h1>

<div class="container">


    <div class="left">
        <p id="clock">no faces found</p>
        <script>
            const clock = document.getElementById("clock");

            setInterval(() => {
                fetch("{{ url_for('time_feed') }}")
                    .then(response => {
                        response.text().then(t => {
                            clock.innerHTML = t
                        })
                    });
            }, 10);
        </script>
    </div>
    <div class="right">
        <div class="col-lg-8  offset-lg-2">
            <img src="{{ url_for('video_feed') }}" height="80%" alt="please
wait...">
        </div>
    </div>
</div>
</body>



</html>
```

Description:

<!doctype html>:

This declaration defines the document type as HTML5, indicating that the page follows modern HTML standards.

<html lang="fr">:

This opening tag specifies that the document is written in French (the "fr" language code). This helps search engines and browsers understand the language of the content.

<head>:

Within the head section, metadata and links to external resources are typically placed.

<title>Emotion Recognition</title>:

This sets the title of the web page, which is displayed in the browser's title bar or tab. In this case, the title is "Emotion Recognition."

<link rel="stylesheet" href="../staticFiles/index.css" type="text/css">:

This line links an external CSS stylesheet to the HTML document. The stylesheet is located at **"../staticFiles/index.css**" and is used to style the content of the web page.

<body>:

This is the main content area of the web page.

<h1>Emotion Recognition</h1>:

This is the main heading of the page, displaying "Emotion Recognition" as the title at the top of the web page.

<div class="container">:

This is a container that likely helps structure the content and layout of the page.

<div class="left">:

This div contains content that appears on the left side of the page.

<p id="clock">no faces found</p>:

This paragraph element with the ID "clock" initially displays "no faces found." It is likely intended to display real-time information.

The following JavaScript code updates the content of the "clock" element at regular intervals. It appears to fetch data from a URL (presumably related to time or real-time data) and updates the "clock" element with the fetched data. The fetch request is sent every 10 milliseconds.

<div class="right">:

This div contains content that appears on the right side of the page.

<div class="col-lg-8 offset-lg-2">:

This div is part of a grid or layout system (likely from a CSS framework like Bootstrap). It has a specific width ("col-lg-8") and offset ("offset-lg-2").

<img src="{{ url_for('video_feed') }}" height="80%" alt="please wait...">:

This image element displays an image sourced from the URL generated by the url_for('video_feed') Flask function. The image has a specified height of 80%, and the alt text is "please wait...," which is displayed if the image cannot be loaded or is still loading.

CSS CODE:

```
.container{
    display: inline-flex;
    width: 90%;
    margin-left:5%;
    margin-right:5%;
}
html{
    background-color:#161a1b;
}
```

```

.left{
    float: left;
    background-color: lightgray;
    width:30%;
    padding:4%;
    font-size:220%;
}
.right{
    border:none;
    float: right;
    width:70%;
    text-align: center;
}
.right img {
    width:100%;
}

h1{
    text-align: center;
    color:white;
}
```

Description:

.container:

This CSS class is used to style a container element.

display: inline-flex;:

It sets the display property to "inline-flex," which makes the container an inline-level flex container. This allows its child elements to be displayed in a row, and they will adjust their sizes based on content.

width: 90%;:

The container occupies 90% of the available width of its parent element.

margin-left: 5%; and margin-right: 5%;:

These properties set left and right margins to 5%, which create spacing on the left and right sides of the container.

html:

These styles are applied to the <html> element, affecting the entire web page.

background-color: #161a1b;:

It sets the background color of the HTML element to a dark grayish color (#161a1b).

.left:

This class is used to style the left part of the content.

float: left;:

The "left" content is floated to the left side of its containing element. This allows the content on the right to flow around it.

background-color: lightgray;:

The background color of the left content is set to light gray.

width: 30%;:

The left content takes up 30% of the available width.

padding: 4%;:

It adds a 4% padding on all sides of the left content, creating spacing between the content and its container.

font-size: 220%;:

The font size for the left content is set to 220% of the default font size, making it quite large.

.right:

This class is used to style the right part of the content.

border: none;:

It removes the border from the right content.

float: right;:

The "right" content is floated to the right side of its containing element, allowing it to sit next to the left content.

width: 70%;:

The right content takes up 70% of the available width.

text-align: center;:

Text within the right content is centered horizontally.

.right img:

This selector targets <img> elements within the "right" content.

width: 100%;:

Images within the "right" content are scaled to occupy the full width of their container.

h1:

This selector targets <h1> elements.

text-align: center;:

The text in <h1> elements is centered horizontally.

color: white;:

The text color of <h1> elements is set to white.

JS code:

```
const clock = document.getElementById("clock");


setInterval(() => {
    fetch("{{ url_for('time_feed') }}")
        .then(response => {
            response.text().then(t => {
                clock.innerHTML = t
            })
        });
}, 2000);
```

Description:

const clock = document.getElementById("clock");:

This line retrieves an HTML element with the ID "clock" and stores it in a JavaScript variable called "clock." This element is likely a part of the webpage where you want to display real-time data, such as a clock or timestamp.

setInterval(() => {...}, 2000);:

This is a JavaScript function that repeatedly executes the provided code block at the specified interval in milliseconds. In this case, the code block is executed every 2000 milliseconds (2 seconds).

fetch("{{ url_for('time_feed') }}"):

Inside the interval function, a "fetch" request is made to a URL generated using Flask's url_for function with the endpoint "time_feed." The purpose of this request is to retrieve some real-time data, possibly a timestamp or clock data, from the server.

.then(response => {...}):

This is a promise that handles the response from the fetch request. It specifies what to do when the response is received.

response.text().then(t => { clock.innerHTML = t });:

This part of the code processes the response from the server. It converts the response to text and then updates the content of the HTML element with the ID "clock" (as stored in the "clock" variable) with the text received from the server. This action effectively updates the displayed content with the real-time data obtained from the server.

Conclusion:

By focusing on the frontend development in Part 1, we aim to provide users with an engaging and easy-to-use interface that seamlessly integrates with the backend, where our image recognition model does the heavy lifting. The result will be a user-friendly image recognition application that empowers users to understand and describe images with ease.