Final Sprint Flask + Website Code

```
Text to speech
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale
<title>Embeddable Text-to-Speech</title>
<style>
    /* CSS to make the input field and button bigger */
    #tts-input {
        font-size: 18px; /* Increase font size */
        padding: 10px; /* Increase padding */
        width: 300px; /* Increase width */
    }

    #tts-button {
        font-size: 18px; /* Increase font size */
        padding: 12px 24px; /* Increase padding */
    }
</style>
</head>
<body>
    <!-- Embeddable Text-to-Speech Interface -->
    <form id="tts-form">
        <input type="text" id="tts-input" placeholder="Type what
        <input type="submit" id="tts-button" value="Speak" hidde
    </form>

    <!-- Script for handling the Text-to-Speech -->
    <script>
```

```
    document.getElementById("tts-form").addEventListener("submi
        event.preventDefault(); // Prevent the default form subm
        var text = document.getElementById("tts-input").value;
        if(text) {
            var xhr = new XMLHttpRequest();
            xhr.open("POST", "http://10.211.55.7:5000/speak", tr
            xhr.setRequestHeader("Content-Type", "application/js
            xhr.onload = function () {
                if (xhr.status === 200) {
                    console.log("Text sent to the robot:", text)
                    document.getElementById("tts-input").value =
                    // Speak "Response sent" after submitting
                    speakResponse("Response sent");
                } else {
                    console.error("Error from server:", xhr.stat
                }
            };
            xhr.onerror = function () {
                console.error("Request failed");
            };
            var data = JSON.stringify({"text": text});
            xhr.send(data);
        } else {
            alert("Please enter some text to speak.");
        }
    });

    // Function to speak a given text
    function speakResponse(text) {
        var msg = new SpeechSynthesisUtterance();
        msg.text = text;
        window.speechSynthesis.speak(msg);
    }

    // Clear input field and focus after speaking
    document.getElementById("tts-input").addEventListener("input
```

```
        // If Enter key is pressed, focus on input field
        if (event.keyCode === 13) {
            event.preventDefault();
            document.getElementById("tts-input").value = '';
            document.getElementById("tts-input").focus();
        }
    });
    </script>
</body>
</html>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale
<title>Robot Remote Controller</title>
<style>
    body {
        font-family: Arial, sans-serif;
        text-align: center;
        overflow: hidden;
    }
    .container {
        margin-top: 50px;
    }
    .button-group {
        display: flex;
        justify-content: center;
        margin-bottom: 20px;
    }
    button {
        padding: 10px 20px;
        font-size: 16px;
        margin: 5px;
```

```
        cursor: pointer;
    }
    #forwardBtn {
        width: calc((100% - 20px) / 2); /* Half of the containe
        height: auto;
    }
</style>
</head>
<body>
<div class="container">
    <h1>Robot Remote Controller</h1>
    <div class="button-group">
        <button id="turnLeftBtn">Turn Left</button>
        <button id="turnRightBtn">Turn Right</button>
    </div>
    <div class="button-group">
        <button id="turnHeadLeftBtn">Turn Head Left</button>
        <button id="turnHeadRightBtn">Turn Head Right</button>
    </div>
    <button id="forwardBtn">Walk Forward</button><br>
</div>
<script>
    document.getElementById("turnLeftBtn").addEventListener("cl
        var xhr = new XMLHttpRequest();
        xhr.open('POST', 'http://10.211.55.7:5000/turn_left', t
        xhr.setRequestHeader('Content-Type', 'application/json'
        xhr.onreadystatechange = function () {
            if (xhr.readyState === 4 && xhr.status === 200) {
                console.log(xhr.responseText);
            }
        };
        xhr.send();
    });

    document.getElementById("turnRightBtn").addEventListener("c
        var xhr = new XMLHttpRequest();
```

```
        xhr.open('POST', 'http://10.211.55.7:5000/turn_right', t
        xhr.setRequestHeader('Content-Type', 'application/json')
        xhr.onreadystatechange = function () {
            if (xhr.readyState === 4 && xhr.status === 200) {
                console.log(xhr.responseText);
            }
        };
        xhr.send();
    });

    document.getElementById("turnHeadLeftBtn").addEventListener(
        var xhr = new XMLHttpRequest();
        xhr.open('POST', 'http://10.211.55.7:5000/look_left', tr
        xhr.setRequestHeader('Content-Type', 'application/json')
        xhr.onreadystatechange = function () {
            if (xhr.readyState === 4 && xhr.status === 200) {
                console.log(xhr.responseText);
            }
        };
        xhr.send();
    });

   document.getElementById("turnHeadRightBtn").addEventListener(
    var xhr = new XMLHttpRequest();
    xhr.open('POST', 'http://10.211.55.7:5000/look_right', true)
    xhr.setRequestHeader('Content-Type', 'application/json');
    xhr.onreadystatechange = function () {
        if (xhr.readyState === 4 && xhr.status === 200) {
            console.log(xhr.responseText);
        }
    };
    xhr.send();
});



    document.getElementById("forwardBtn").addEventListener("clic
```

```
        var xhr = new XMLHttpRequest();
        xhr.open('POST', 'http://10.211.55.7:5000/walk', true);
        xhr.setRequestHeader('Content-Type', 'application/json')
        xhr.onreadystatechange = function () {
            if (xhr.readyState === 4 && xhr.status === 200) {
                console.log(xhr.responseText);
            }
        };
        xhr.send();
    });
</script>
</body>
</html>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>NAO Robot Live Feed</title>
    <style>
        #placeholder {
            display: none; /* Initially hidden */
            text-align: center;
            background-color: black; /* Set background to black
            color: white; /* Set text color to white */
            height: 100vh; /* Make the placeholder take full vie
            width: 100%; /* Make the placeholder take full width
            position: fixed; /* Fixed position to cover the whol
            top: 0; /* Position at the top */
            left: 0; /* Position at the left */
            z-index: 1000; /* Make sure it's on top of other ele
            display: flex;
            justify-content: center; /* Center content horizonta
            align-items: center; /* Center content vertically */
        }
```

```
        #videoFeed {
            width: 100%; /* Make the video feed take full width
        }
    </style>
</head>
<body>
    <h1>NAO Robot Live Stream</h1>
    <img id="videoFeed" src="" alt="Live Stream" onerror="showPl
    <div id="placeholder">
        <p>Waiting for the NAO Robot to go live...</p>
    </div>

    <script>
        function showPlaceholder() {
            document.getElementById('placeholder').style.display
            document.getElementById('videoFeed').style.display =
        }

        window.onload = function() {
            var img = document.getElementById('videoFeed');
            img.onerror = showPlaceholder;
            img.onload = function() {
                img.style.display = 'block'; // Show the video f
                document.getElementById('placeholder').style.dis
            };

            // Append a timestamp to the video feed URL to force
            img.src = "http://10.211.55.7:5000/video_feed?" + ne
        };
    </script>
</body>
</html>
```

```
Flask server
from flask import Flask, Response, request
```

```
import cv2
import numpy as np
from naoqi import ALProxy
import vision_definitions
import sys




app = Flask(__name__)

# NAO robot IP and Port
NAO_IP = "192.168.86.152"  # Replace 'your_nao_ip' with your NAO
NAO_PORT = 9559  # Default port

#head angle
head_yaw_angle = 0.0

# Create a proxy to ALVideoDevice on the robot
try:
    video_service = ALProxy("ALVideoDevice", NAO_IP, NAO_PORT)
except Exception as e:
    print("Could not create proxy to ALVideoDevice: {}".format(e
    exit(1)

# Subscribe to the camera feed
resolution = vision_definitions.kVGA  # 640x480
colorSpace = vision_definitions.kRGBColorSpace
fps = 20

camera_index = 0  # 0 for the top camera, 1 for the bottom camer

try:
    video_client = video_service.subscribeCamera(
        "python_client", camera_index, resolution, colorSpace, f
except Exception as e:
```

```
    print("Could not subscribe to camera: {}".format(e))
    exit(1)


def gen_frames():
    while True:
        # Obtain an image from the robot's camera
        nao_image = video_service.getImageRemote(video_client)

        if nao_image is None:
            continue

        # Get the image size and pixel array.
        image_width = nao_image[0]
        image_height = nao_image[1]
        array = nao_image[6]
        image_string = bytes(bytearray(array))

        # Convert the string to an image
        img = np.frombuffer(image_string, dtype=np.uint8)
        img = img.reshape((image_height, image_width, 3))

        # Encode the frame in JPEG format
        (flag, encodedImage) = cv2.imencode(".jpg", img)
        if not flag:
            continue

        # Yield the encoded image in byte format
        yield (b'--frame\r\n'
               b'Content-Type: image/jpeg\r\n\r\n' +
               encodedImage.tobytes() + b'\r\n')


def right_head_turn():
    try:
        # Create proxies for necessary modules
```

```
        motion_proxy = ALProxy("ALMotion", NAO_IP, NAO_PORT)
    except Exception as e:
        print("Could not create a proxy to ALMotionDevice: {}".f
        exit(1)

    # Set stiffness for head motors
    motion_proxy.setStiffnesses("Head", 1.0)

    # Define the desired angles for head movement (in radians)
    # The head joints are HeadYaw and HeadPitch
    # HeadYaw controls left-right movement and HeadPitch control
    # Make sure the values are within the range supported by the
    head_yaw_angle = head_yaw_angle - .1  # Example angle for le
    head_pitch_angle = 0.0  # Example angle for up-down movement

    # Set the angles for head movement
    motion_proxy.setAngles(["HeadYaw", "HeadPitch"], [head_yaw_a

    # Wait for the movement to complete (you can adjust this tim
    motion_proxy.waitUntilMoveIsFinished()

def left_head_turn():
    try:
        # Create proxies for necessary modules
        motion_proxy = ALProxy("ALMotion", NAO_IP, NAO_PORT)
    except Exception as e:
        print("Could not create a proxy to ALMotionDevice: {}".f
        exit(1)

    # Set stiffness for head motors
    motion_proxy.setStiffnesses("Head", 1.0)

    # Define the desired angles for head movement (in radians)
    # The head joints are HeadYaw and HeadPitch
    # HeadYaw controls left-right movement and HeadPitch control
    # Make sure the values are within the range supported by the
```

```
    head_yaw_angle = head_yaw_angle + .1  # Example angle for le
    head_pitch_angle = 0.0  # Example angle for up-down movement

    # Set the angles for head movement
    motion_proxy.setAngles(["HeadYaw", "HeadPitch"], [head_yaw_a

    # Wait for the movement to complete (you can adjust this tir
    motion_proxy.waitUntilMoveIsFinished()

@app.route('/video_feed')
def video_feed():
    # Return the response generated along with the specific medi
    # type (mime type).
    return Response(gen_frames(), mimetype='multipart/x-mixed-re

@app.route('/turn_left', methods=['POST'])
def handle_turn_left():
    # Call the function to turn the robot left
    return 'Turned left'

@app.route('/turn_right', methods=['POST'])
def handle_turn_right():
    # Call the function to turn the robot right
    return 'Turned right'

@app.route('/look_left', methods=['POST'])
def handle_look_left():
    # Call the function to turn the robot's head left
    return 'Looked left'

@app.route('/look_right', methods=['POST'])
def handle_look_right():
    # Call the function to turn the robot's head right
    return 'Looked right'

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

```
def walk_forward():
    try:
        motion_proxy = ALProxy("ALMotion", NAO_IP, NAO_PORT)
        motion_proxy.walkTo(0.5, 0, 0)  # Move 0.5 meters forwar
    except Exception as e:
        print("Could not make the robot walk: {}".format(e))
    return 'Walked forward'

@app.route('/speak', methods=['POST'])
def robot_speak():
    data = request.json.get('text', '')
    if data:
        try:
            tts = ALProxy("ALTextToSpeech", IP, PORT)
            tts.say(data)
            return "Success", 200
        except Exception as e:
            return "An error occurred: {}".format(str(e)), 500
    return "No text provided", 400



if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```