Documentation for Using Talking Head API Endpoints on Port 5100

1 Overview

The Talking Head API provides several endpoints to manage and interact with an animated talking head. This includes loading characters, starting and stopping mouth animations, setting emotions, and streaming video frames. Additionally, there is a classification endpoint to analyze text and update the talking head's emotion accordingly.

2 API Endpoints

2.1 Load Talking Head Character

Endpoint: `/api/talkinghead/load`
Method: `POST`
Description: Loads the talking head sprite posted in the request. Resumes animation if the talking head module was paused.
Request Body: Form data containing the image file to load.
Example:

```
curl -X POST -F "file=@character.png" http://localhost:5100/api/
   talkinghead/load
```

Usage:

- **Localhost:** `http://localhost:5100/api/talkinghead/load`
- **Azure/AWS:** Replace `localhost` with your server's address and include the API key in headers if required.

2.2 Load Custom Emotion Templates

Endpoint: `/api/talkinghead/load_emotion_templates`
Method: `POST`
Description: Loads custom emotion templates for the talking head or resets to defaults.
Request Body: JSON object with emotion templates. Send a blank JSON to reset to defaults.
Example:

```
curl -X POST -H "Content-Type:␣application/json" -d '{}' http://localhost
   :5100/api/talkinghead/load_emotion_templates
```

Usage:

- **Localhost:** `http://localhost:5100/api/talkinghead/load_emotion_templates`
- **Azure/AWS:** Replace `localhost` with your server's address and include the API key in headers if required.

2.3 Load Custom Animator Settings

Endpoint: `/api/talkinghead/load_animator_settings`
Method: `POST`
Description: Loads custom settings for the talking head animator and postprocessor or resets to defaults.
Request Body: JSON object with animator settings. Send a blank JSON to reset to defaults.
Example:

```
curl -X POST -H "Content-Type:␣application/json" -d '{}' http://localhost
   :5100/api/talkinghead/load_animator_settings
```

Usage:

- **Localhost:** `http://localhost:5100/api/talkinghead/load_animator_settings`
- **Azure/AWS:** Replace `localhost` with your server's address and include the API key in headers if required.

2.4 Unload Talking Head

Endpoint: `/api/talkinghead/unload`
Method: `GET`
Description: Pauses the talking head module. To resume, load a character via `/api/talkinghead/loa`
Example:

```
curl -X GET http://localhost:5100/api/talkinghead/unload
```

Usage:

- **Localhost:** `http://localhost:5100/api/talkinghead/unload`
- **Azure/AWS:** Replace `localhost` with your server's address and include the API key in headers if required.

2.5 Start Talking Animation

Endpoint: `/api/talkinghead/start_talking`
Method: `GET`
Description: Starts the mouth animation for talking.
Example:

```
curl -X GET http://localhost:5100/api/talkinghead/start_talking
```

Usage:

- **Localhost:** `http://localhost:5100/api/talkinghead/start_talking`
- **Azure/AWS:** Replace `localhost` with your server's address and include the API key in headers if required.

2.6 Stop Talking Animation

Endpoint: `/api/talkinghead/stop_talking`
Method: `GET`
Description: Stops the mouth animation for talking.
Example:

```
curl -X GET http://localhost:5100/api/talkinghead/stop_talking
```

Usage:

- **Localhost:** `http://localhost:5100/api/talkinghead/stop_talking`
- **Azure/AWS:** Replace `localhost` with your server's address and include the API key in headers if required.

2.7 Set Emotion

Endpoint: `/api/talkinghead/set_emotion`
Method: `POST`
Description: Sets the talking head character emotion.
Request Body: JSON object with the emotion name.

```
{
  "emotion_name": "joy"
}
```

Example:

```
curl -X POST -H "Content-Type:␣application/json" -d '{"emotion_name":␣"
   joy"}' http://localhost:5100/api/talkinghead/set_emotion
```

Usage:

- **Localhost:** `http://localhost:5100/api/talkinghead/set_emotion`
- **Azure/AWS:** Replace `localhost` with your server's address and include the API key in headers if required.

2.8 Stream Video Frames

Endpoint: `/api/talkinghead/result_feed`
Method: `GET`
Description: Streams live character output as a series of PNG encoded images.
Example:

```
curl -X GET http://localhost:5100/api/talkinghead/result_feed
```

Usage:

- **Localhost:** `http://localhost:5100/api/talkinghead/result_feed`
- **Azure/AWS:** Replace `localhost` with your server's address and include the API key in headers if required.

2.9 Classify Text

Endpoint: `/api/classify`
Method: `POST`
Description: Performs sentiment analysis on the text and returns the classification result. If the talking head module is enabled, it automatically updates its emotion based on the classification result.
Request Body: JSON object with the text to classify.

```
{
  "text": "I am very happy today!"
}
```

Example:

```
curl -X POST -H "Content-Type:␣application/json" -d '{"text":␣"I␣am␣very␣
   happy␣today!"}' http://localhost:5100/api/classify
```

Usage:

- **Localhost:** `http://localhost:5100/api/classify`
- **Azure/AWS:** Replace `localhost` with your server's address and include the API key in headers if required.

2.10 1Get Classification Labels

Endpoint: `/api/classify/labels`
Method: `GET`
Description: Returns the available classifier labels for text sentiment (character emotion).
Example:

```
curl -X GET http://localhost:5100/api/classify/labels
```

Usage:

- **Localhost:** `http://localhost:5100/api/classify/labels`
- **Azure/AWS:** Replace `localhost` with your server's address and include the API key in headers if required.

3 Added Endpoints

3.1 Generate Audio

Endpoint: `/api/generate_audio`
Method: `POST`
Description: Generates audio from the provided text and voice using the AllTalk TTS module.
Request Body: JSON object containing the text, voice, and language.

```
{
  "text": "Hello, world!",
  "voice": "voice1",
  "language": "en"
}
```

Example:

```
curl -X POST -H "Content-Type:␣application/json" -d '{"text":␣"Hello,␣
   world!",␣"voice":␣"voice1",␣"language":␣"en"}' http://localhost:5100/
   api/generate_audio
```

Usage:

- **Localhost:** `http://localhost:5100/api/generate_audio`
- **Azure/AWS:** Replace `localhost` with your server's address and include the API key in headers if required.

3.2 Chat

Endpoint: `/api/chat`
Method: `POST`
Description: Sends a chat prompt to the Groq API and returns the response from the language model.
Request Body: JSON object containing the prompt.

```
{
  "prompt": "What is the weather today?"
}
```

Example:

```
curl -X POST -H "Content-Type:␣application/json" -d '{"prompt":␣"What␣is␣
   the␣weather␣today?"}' http://localhost:5100/api/chat
```

Usage:

- **Localhost:** `http://localhost:5100/api/chat`
- **Azure/AWS:** Replace `localhost` with your server's address and include the API key in headers if required.

4 Example Scenario

4.1 Scenario 1: Streaming Live Video with Expression Changes

1. **Load Character:**

   ```
   curl -X POST -F "file=@character.png" http://localhost:5100/api/
      talkinghead/load
   ```

2. **Start Talking:**

   ```
   curl -X GET http://localhost:5100/api/talkinghead/start_talking
   ```

3. **Change Expression:**

   ```
   curl -X POST -H "Content-Type:␣application/json" -d '{"
      emotion_name":␣"joy"}' http://localhost:5100/api/talkinghead/
      set_emotion
   ```

4. **Stream Video Frames:**

   ```
   curl -X GET http://localhost:5100/api/talkinghead/result_feed
   ```

5. **Toggle Talking and Change Expression:** Set up a script or use the provided JavaScript to toggle talking and change expressions every 5 seconds.

4.2 Scenario 2: Classify Text and Update Emotion

1. **Classify Text:**

```
curl -X POST -H "Content-Type:␣application/json" -d ’{"text":␣"I␣
   am␣very␣happy␣today!"}’ http://localhost:5100/api/classify
```

2. **Set Emotion Based on Classification:** The emotion will be automatically set if the talking head module is enabled.

5 Considerations for Azure or AWS Deployment

- **API Endpoint URL:** Replace `localhost` with your server’s address.
- **API Keys:** Include API keys in headers if required.

```
curl -X POST -H "Content-Type:␣application/json" -H "Authorization
   :␣Bearer␣YOUR_API_KEY" -d ’{"text":␣"I␣am␣very␣happy␣today!"}’
   http://your-server-address/api/classify
```

6 Conclusion

This documentation covers all the essential endpoints related to the Talking Head functionality, providing detailed usage instructions and scenarios.

7 Example HTML and JavaScript

You can look at the following HTML and JavaScript codes for better understanding.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,␣initial-scale=1.0"
        >
    <title>Talking Head</title>
    <style>
        body {
            display: flex;
            flex-direction: row;
            height: 100vh;
            margin: 0;
            font-family: Arial, sans-serif;
        }
        #left-panel {
            flex: 1;
            display: flex;
            flex-direction: column;
            align-items: center;
            justify-content: center;
            background-color: #f0f0f0;
            padding: 20px;
        }
        #right-panel {
            flex: 1;
            display: flex;
            flex-direction: column;
            justify-content: space-between;
            background-color: #ffffff;
            border-left: 1px solid #ccc;
            padding: 20px;
        }
        #fileInput, #generateAudioButton, #languageSelector {
            margin: 10px 0;
        }
        #expression-holder {
            margin-top: 20px;
        }
        #chatBox {
            flex: 1;
            display: flex;
            flex-direction: column;
            border: 1px solid #ccc;
            border-radius: 5px;
            padding: 10px;
            overflow-y: auto;
            background-color: #fafafa;
        }
        .message {
            margin: 5px 0;
            padding: 10px;
            border-radius: 5px;
        }
        .user-message {
            align-self: flex-end;
            background-color: #dcf8c6;
        }
        .bot-message {
            align-self: flex-start;
            background-color: #e6e6e6;
        }
        #chatInputContainer {
            display: flex;
            margin-top: 10px;
        }
        #chatInput {
            flex: 1;
            padding: 10px;
            border: 1px solid #ccc;
            border-radius: 5px;
        }
        #sendButton {
            margin-left: 10px;
            padding: 10px;
            background-color: #007bff;
            color: #fff;
            border: none;
            border-radius: 5px;
            cursor: pointer;
        }
        #sendButton:hover {
            background-color: #0056b3;
        }
    </style>
</head>
<body>
    <div id="left-panel">
        <input type="file" id="fileInput">
        <button id="generateAudioButton">Generate Audio</button>
        <select id="languageSelector">
            <option value="en">English</option>
            <option value="ja">Japanese</option>
        </select>
        <div id="expression-holder" style="display:␣none;">
            <img id="expression-image" class="expression" src="" alt="
               Character␣Expression">
        </div>
    </div>
    <div id="right-panel">
        <div id="chatBox"></div>
        <div id="chatInputContainer">
            <textarea id="chatInput" rows="2" placeholder="Type␣your␣
               message␣here..."></textarea>
            <button id="sendButton">Send</button>
        </div>
    </div>
    <script>
        document.addEventListener('DOMContentLoaded', () => {
            const serverAddress = 'http://localhost:5100';
            const jsonFilePath = 'http://0.0.0.0:8000/morph_indices.json';
            const audioFilePath = 'http://0.0.0.0:8000/myoutputfile.wav';
            const imgElement = document.getElementById('expression-image')
               ;
            const expressionHolder = document.getElementById('expression-
            holder');
        const chatBox = document.getElementById('chatBox');
        const chatInput = document.getElementById('chatInput');
        const sendButton = document.getElementById('sendButton');
        const languageSelector = document.getElementById('
            languageSelector');
        const expressions = ["anger", "joy", "grief", "fear", "
            surprise"];
        let currentExpressionIndex = 0;

        function loadCharacter() {
            const formData = new FormData();
            const imageFile = document.getElementById('fileInput').
                files[0];
            formData.append('file', imageFile);

            fetch(`${serverAddress}/api/talkinghead/load`, {
                method: 'POST',
                body: formData,
            })
            .then(response => {
                if (!response.ok) {
                    throw new Error('Failed to load character');
                }
                displayTalkingHead();
            })
            .catch(error => console.error('Error loading character:',
                error));
        }

        function generateAudio(text) {
            const voice = "male_01.wav";
            const language = languageSelector.value;

            fetch(`${serverAddress}/api/generate_audio`, {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/json',
                },
                body: JSON.stringify({ text, voice, language }),
            })
            .then(response => response.json())
            .then(data => {
                if (data.status === "success") {
                    console.log("Audio␣generated␣successfully,␣checking
                        ␣for␣JSON␣file␣update...");
                    checkJsonFileUpdate();
                } else {
```

```
                console.error('Error generating audio:', data.
                    message);
            }
        })
        .catch(error => console.error('Error generating audio:',
            error));
    }

    function checkJsonFileUpdate() {
        let lastModifiedTime = null;
        const checkUpdate = () => {
            fetch(jsonFilePath, { method: 'HEAD' })
                .then(response => {
                    const newModifiedTime = new Date(response.
                        headers.get('Last-Modified')).getTime();
                    if (lastModifiedTime === null || newModifiedTime
                         > lastModifiedTime) {
                        console.log("JSON␣file␣updated.");
                        lastModifiedTime = newModifiedTime;
                        startTalking();
                        playAudio();
                    } else {
                        console.log("JSON␣file␣not␣yet␣updated,␣
                            checking␣again...");
                        setTimeout(checkUpdate, 1000); // Check
                            again after 1 second
                    }
                })
                .catch(error => console.error('Error checking JSON
                    file update:', error));
        };

        checkUpdate();
    }

    function startTalking() {
        console.log("Calling␣start_talking␣endpoint...");
        fetch(`${serverAddress}/api/talkinghead/start_talking`, {
            method: 'GET' })
            .then(response => {
                if (!response.ok) {
                    throw new Error('Failed to start talking');
                }
                console.log("Talking␣started.");
            })
            .catch(error => console.error('Error starting talking
                :', error));
    }
```

```
        function playAudio() {
            console.log("Playing␣audio␣from:", audioFilePath);
            const audio = new Audio(audioFilePath);
            audio.play()
                .then(() => {
                    console.log("Audio␣playback␣started.");
                })
                .catch(error => {
                    console.error('Error playing audio:', error);
                });
        }

        function changeExpressionPeriodically() {
            setInterval(() => {
                changeExpression();
            }, 5000);
        }

        function changeExpression() {
            const currentExpression = expressions[
               currentExpressionIndex];
            currentExpressionIndex = (currentExpressionIndex + 1) %
               expressions.length;

            fetch(`${serverAddress}/api/talkinghead/set_emotion`, {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/json',
                },
                body: JSON.stringify({ emotion_name: currentExpression
                   }),
            })
            .then(response => {
                if (!response.ok) {
                    throw new Error('Failed to set emotion');
                }
                console.log(`Expression set to: ${currentExpression}`);
            })
            .catch(error => console.error('Error setting emotion:',
               error));
        }

        function displayTalkingHead() {
            const talkingheadResultFeedSrc = `${serverAddress}/api/
               talkinghead/result_feed`;

            expressionHolder.style.display = 'block';
            if (imgElement.src !== talkingheadResultFeedSrc) {
                imgElement.src = talkingheadResultFeedSrc;
            }
        }

        function getLLMResponse(prompt) {
            fetch(`${serverAddress}/api/chat`, {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/json',
                },
                body: JSON.stringify({ prompt }),
            })
            .then(response => response.json())
            .then(data => {
                const llmResponse = data.response;
                console.log("LLM␣Response:", llmResponse);
                addMessageToChatBox('user', prompt);
                addMessageToChatBox('bot', llmResponse);
                generateAudio(llmResponse);
            })
            .catch(error => console.error('Error fetching LLM response
                :', error));
        }

        function addMessageToChatBox(sender, message) {
            const messageElement = document.createElement('div');
            messageElement.classList.add('message');
            messageElement.classList.add(sender === 'user' ? 'user-
                message' : 'bot-message');
            messageElement.textContent = message;
            chatBox.appendChild(messageElement);
            chatBox.scrollTop = chatBox.scrollHeight;
        }

        sendButton.addEventListener('click', () => {
            const userMessage = chatInput.value;
            if (userMessage.trim()) {
                getLLMResponse(userMessage);
                chatInput.value = '';
            }
        });

        document.getElementById('fileInput').addEventListener('change
            ', loadCharacter);
        document.getElementById('generateAudioButton').
            addEventListener('click', () => generateAudio("Okay,␣Okay!␣
            I␣understand.␣Just␣two␣more␣slides.␣And␣we␣are␣done␣with␣
```

```
                this."));
        });
    </script>
</body>
</html>
```