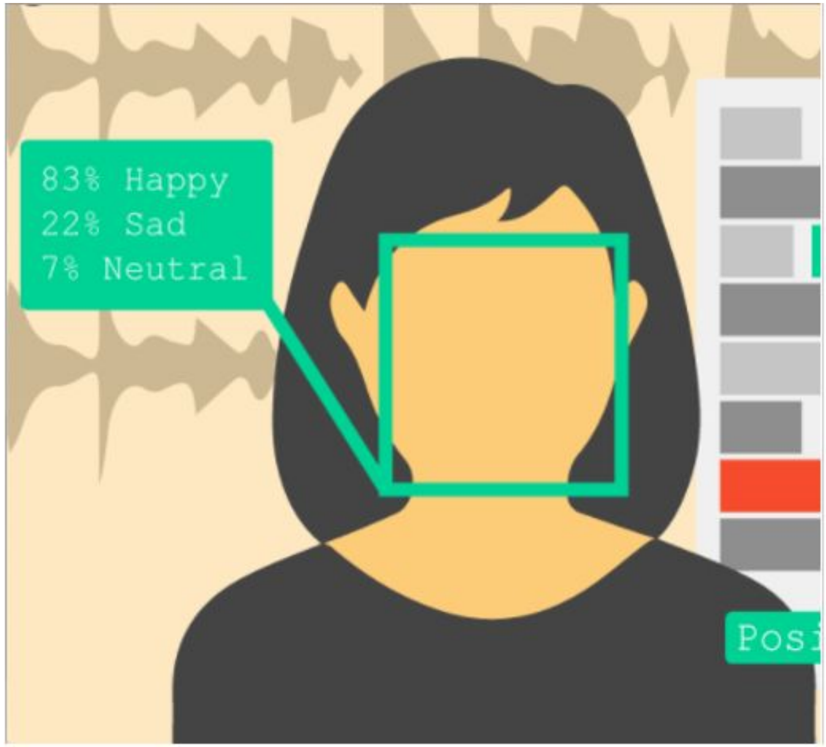


P4 Project -

Elevate - Emotion Recognition & play a song to uplift the mood.

Course

CSCE 5214 (Fall 2020)

Participants

Son Chau | sonchau@my.unt.edu
Naga Sumanth | nagasumanthvankadari@my.unt.edu
Jongwook Yoon | jongwookyoon@my.unt.edu
Miguel Quintana | quintana.miguel@gmail.com

Project Name, Participants, & Workflow

- Project name
  - Elevate - Emotion Recognition & Mood Prediction of Song
- Participants
  - Son Chau, Naga Sumanth, Miguel Quintana, Jongwook Yoon
- Workflow
  - Weekly meeting on Discord every Saturday and/or Sunday morning.
  - Team members are splitted into two groups.
    - Son and Miguel to work on Emotion Recognition Part.
    - Naga and Jong to work on mood classification of song.
- GitHub
  - https://github.com/UNT-5214-P4/Elevate

Project Overview

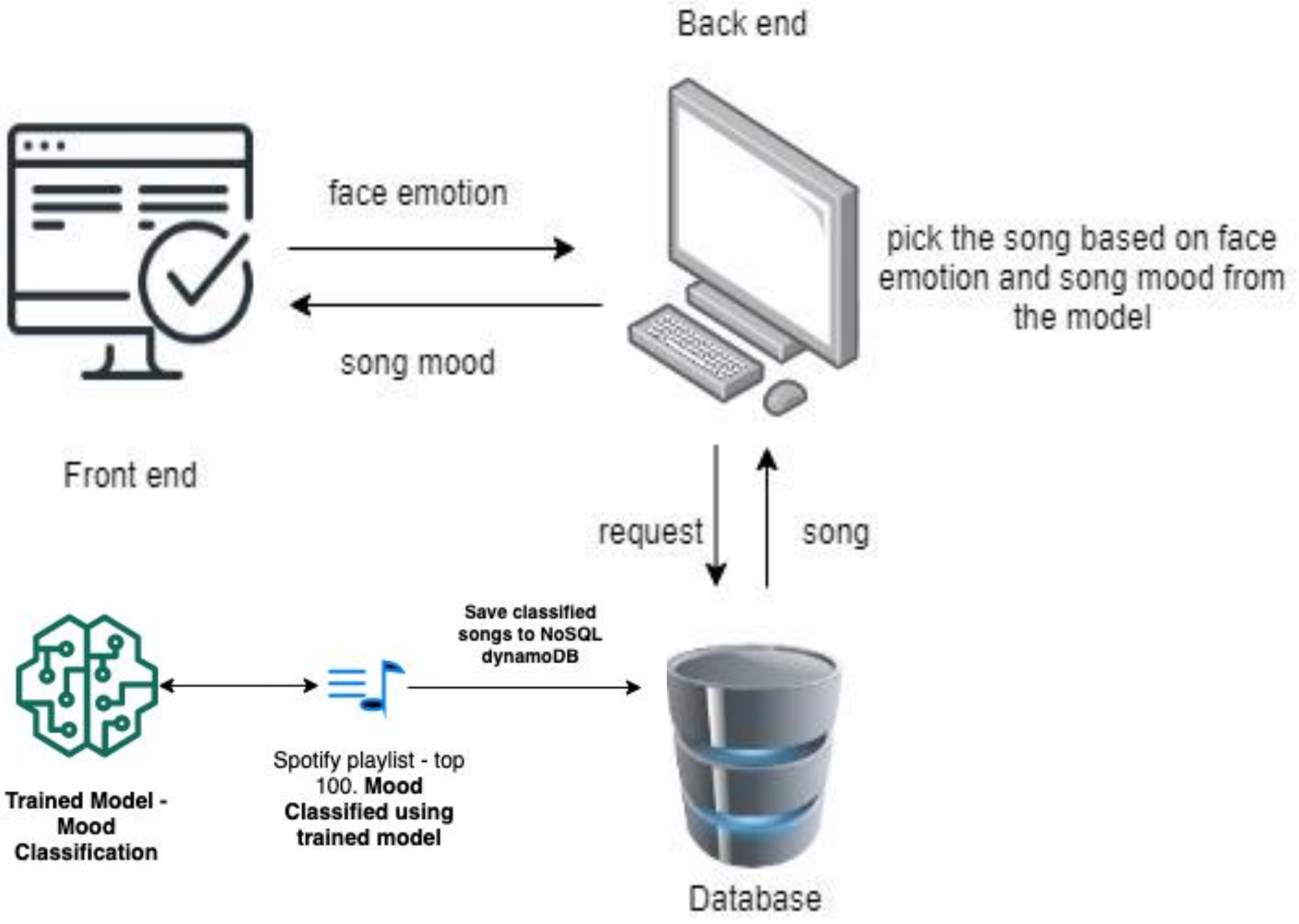
Our Project goal is to determine the emotion recognition and play a song from playlist to uplift the mood.

This involves:

1. Analyzing Video Stream and predicting an emotion. We will need to detect face from video stream and use amazon emotion rekognition API to determine emotion.
2. Build a model that analyzes an audio file and determines the mood of a song. Classify the sample playlist of spotify songs into different moods. Store the mood of song into dynamoDB.
3. Use emotion recognition response and play a song from classified playlist using predefined mapping between emotion <-> mood of the song.

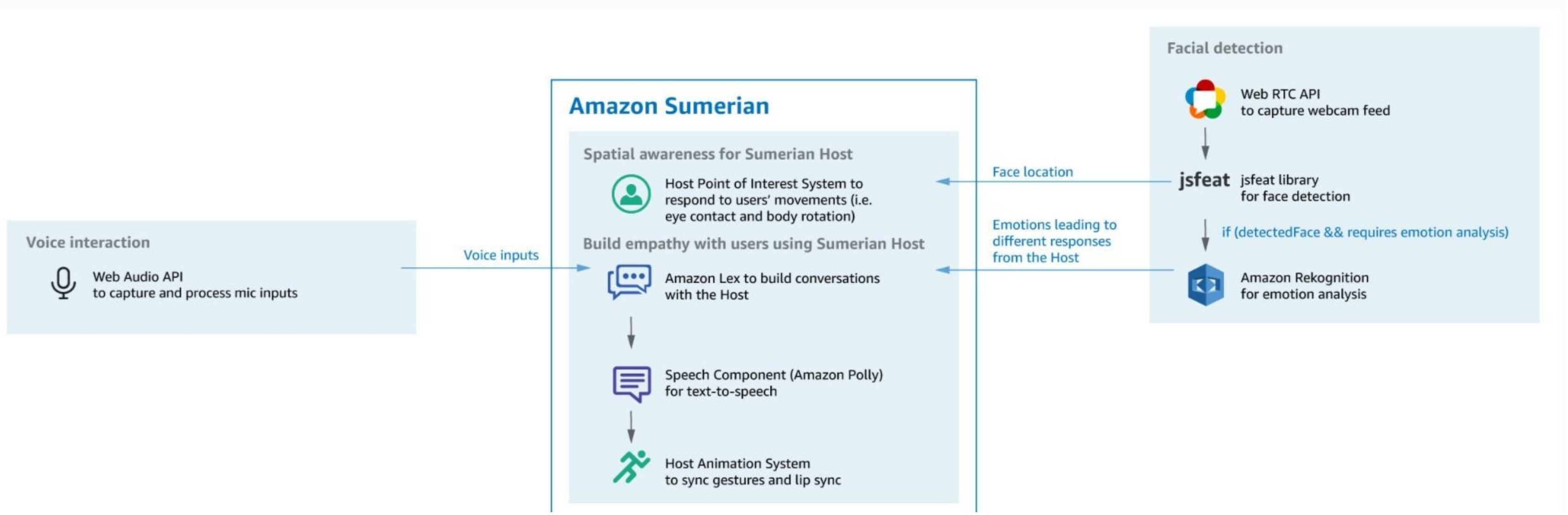
Architecture

- Face detection with jsfeat
- Face emotion with AWS Rekognition
- Interaction using Sumerian, Lex, Polly technologies.
- Song selection based on mood from dynamodb


Back end
face emotion
song mood
pick the song based on face emotion and song mood from the model
Front end
request
song
Save classified songs to NoSQL dynamoDB
Trained Model - Mood Classification
Spotify playlist - top 100. Mood Classified using trained model
Database

Front end

- AWS Sumerian for 3D host
- AWS Lex chatbot
- AWS Polly speech
- AWS Rekognition emotion



Voice interaction

Web Audio API to capture and process mic inputs

Voice inputs

Amazon Sumerian

Spatial awareness for Sumerian Host

Host Point of Interest System to respond to users' movements (i.e. eye contact and body rotation)

Build empathy with users using Sumerian Host

Amazon Lex to build conversations with the Host

Speech Component (Amazon Polly) for text-to-speech

Host Animation System to sync gestures and lip sync

Face location

Emotions leading to different responses from the Host

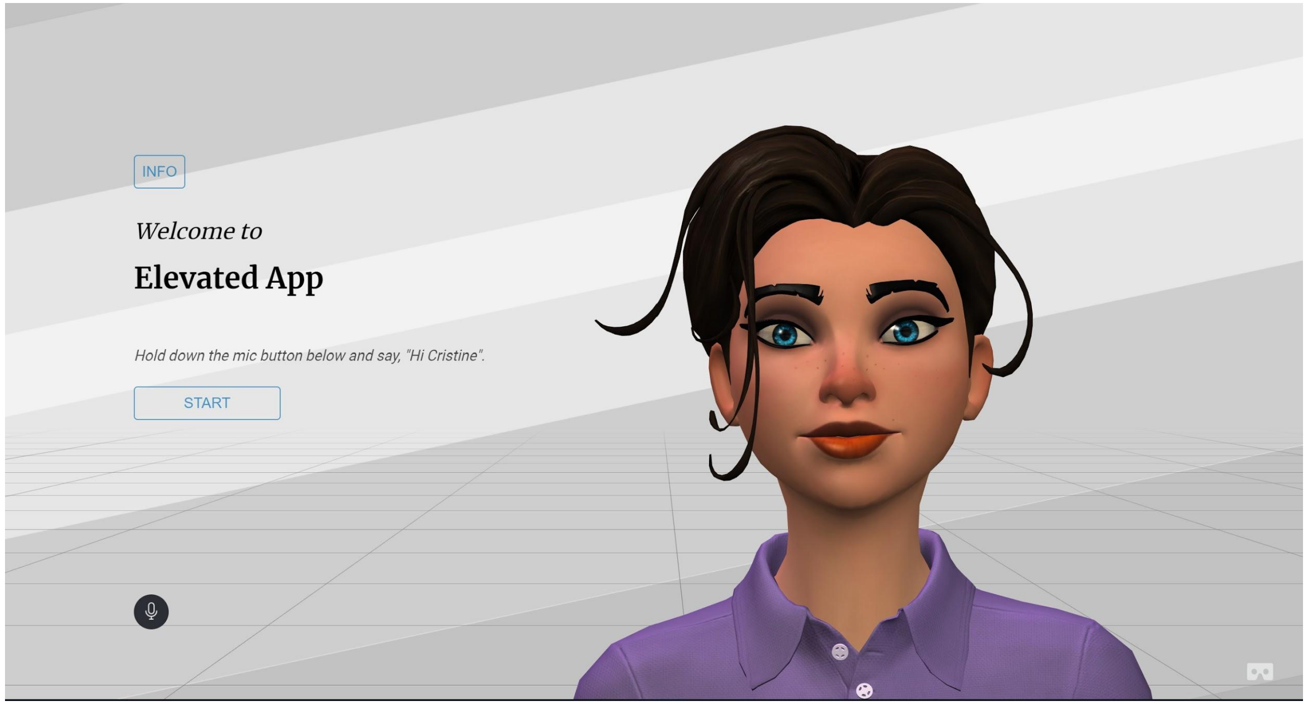
Facial detection

Web RTC API to capture webcam feed

jsfeat jsfeat library for face detection

if (detectedFace && requires emotion analysis)

Amazon Rekognition for emotion analysis



INFO

Welcome to

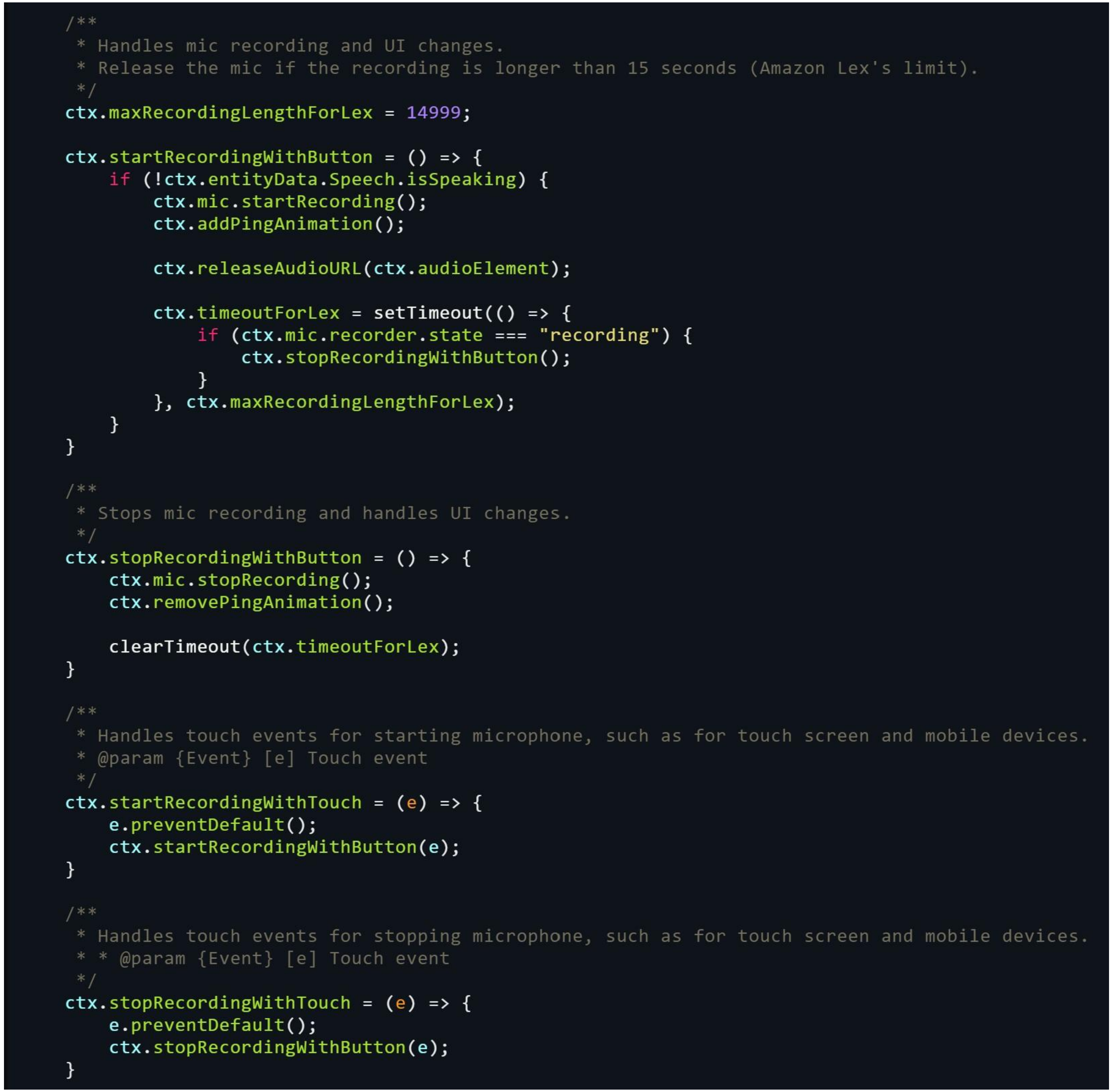
Elevated App

Hold down the mic button below and say, "Hi Cristine".

START

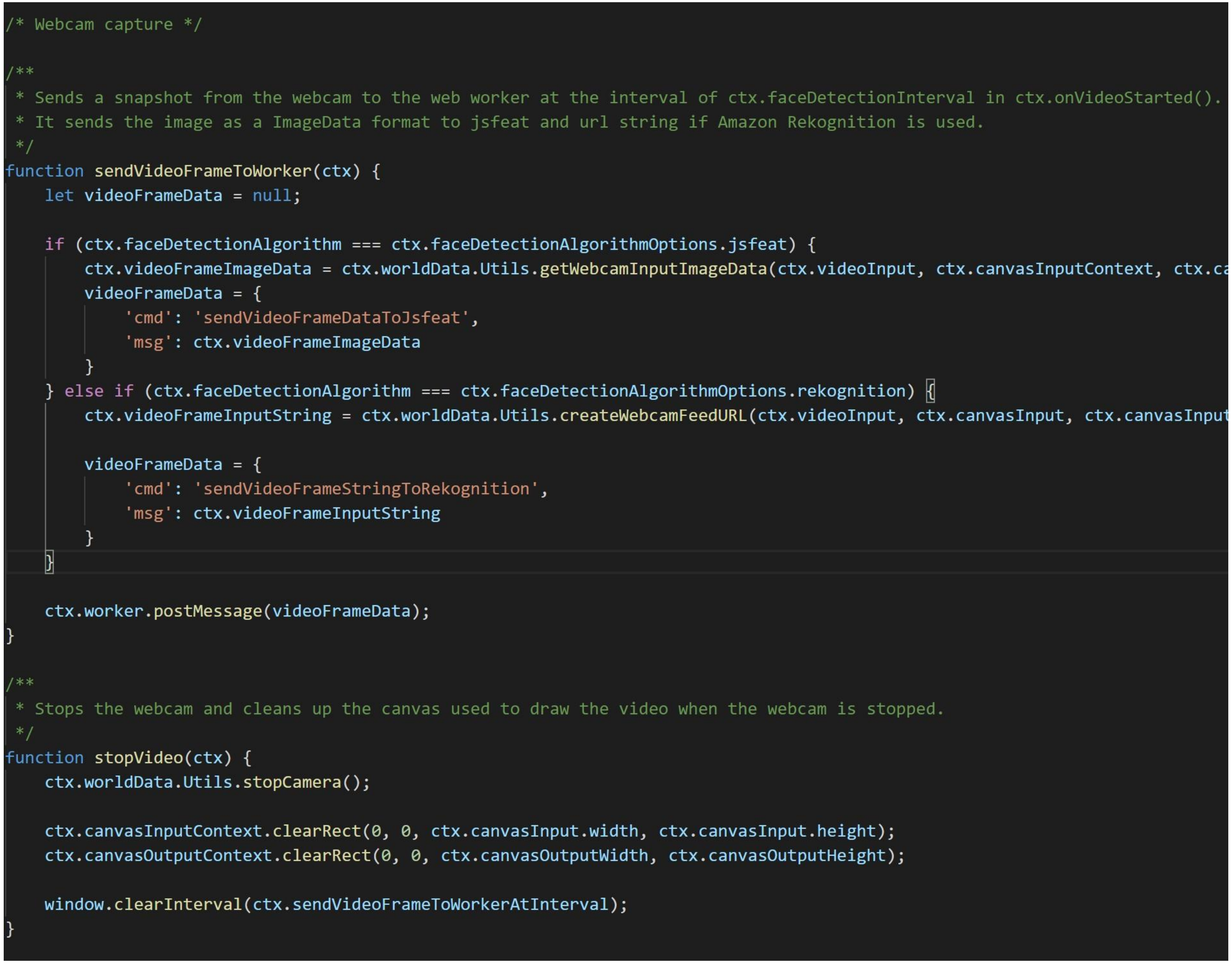
Code:
https://github.com/UNT-5214-P4/Elevate/tree/sumerian

Get mic input



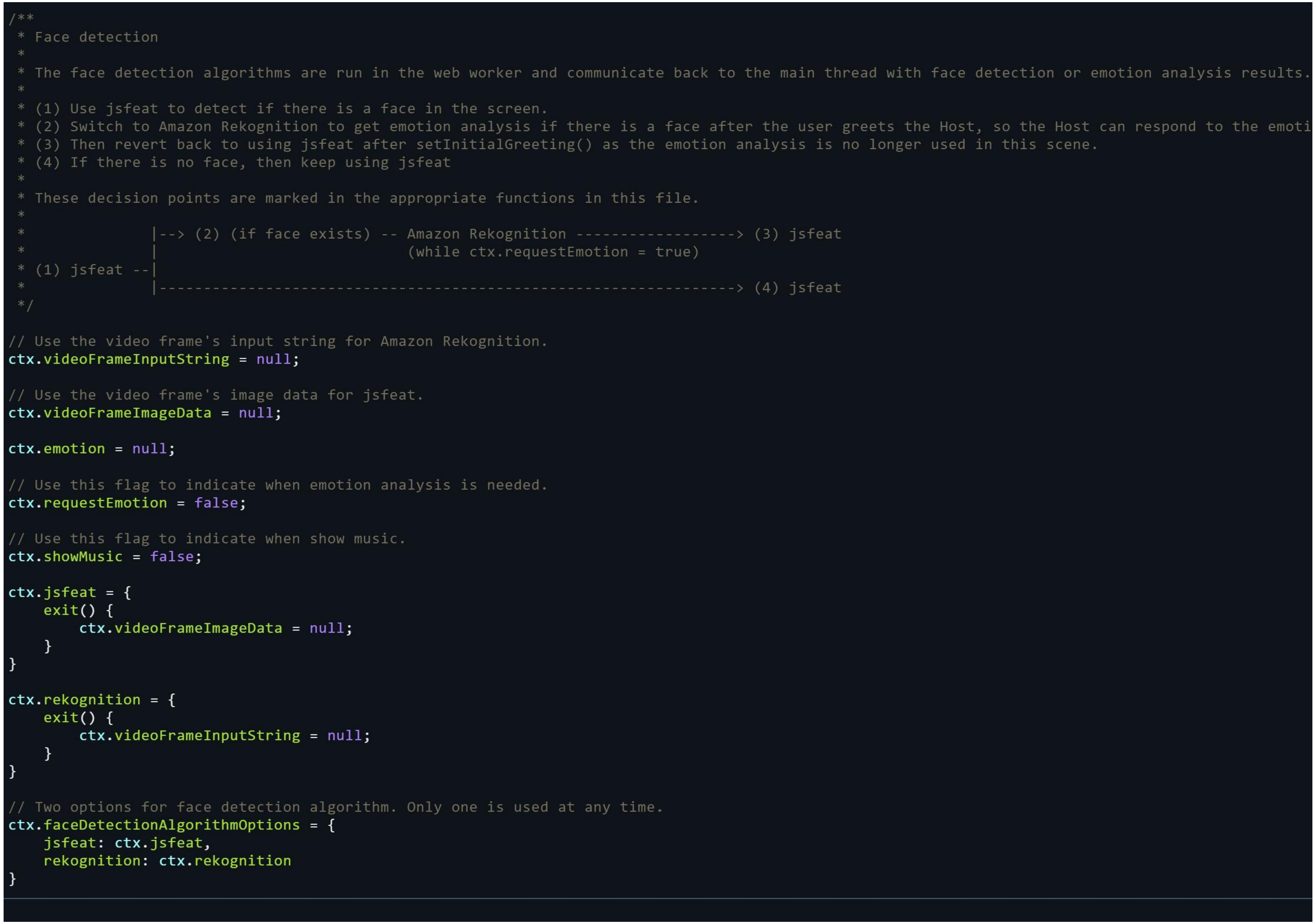
```
/**
 * Handles mic recording and UI changes.
 * Release the mic if the recording is longer than 15 seconds (Amazon Lex's limit).
 */
ctx.maxRecordingLengthForLex = 14999;

ctx.startRecordingWithButton = () => {
    if (!ctx.entityData.Speech.isSpeaking) {
        ctx.mic.startRecording();
        ctx.addPingAnimation();

        ctx.releaseAudioURL(ctx.audioElement);

        ctx.timeoutForLex = setTimeout(() => {
            if (ctx.mic.recorder.state === "recording") {
                ctx.stopRecordingWithButton();
            }
        }, ctx.maxRecordingLengthForLex);
    }
}

/**
 * Stops mic recording and handles UI changes.
 */
ctx.stopRecordingWithButton = () => {
    ctx.mic.stopRecording();
    ctx.removePingAnimation();

    clearTimeout(ctx.timeoutForLex);
}

/**
 * Handles touch events for starting microphone, such as for touch screen and mobile devices.
 * @param {Event} [e] Touch event
 */
ctx.startRecordingWithTouch = (e) => {
    e.preventDefault();
    ctx.startRecordingWithButton(e);
}

/**
 * Handles touch events for stopping microphone, such as for touch screen and mobile devices.
 * * @param {Event} [e] Touch event
 */
ctx.stopRecordingWithTouch = (e) => {
    e.preventDefault();
    ctx.stopRecordingWithButton(e);
}
```

Get webcam input



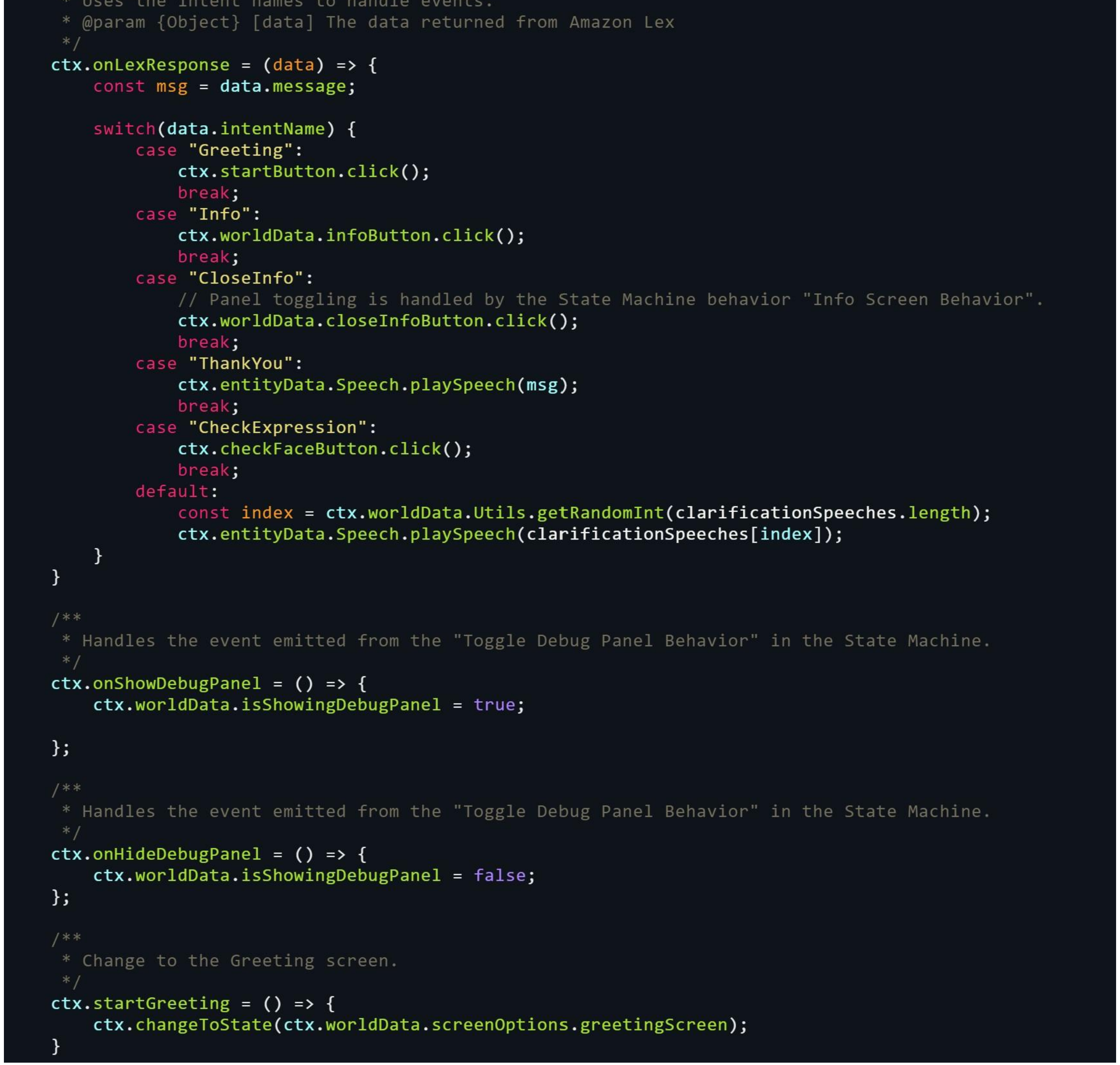
```
/* Webcam capture */

/**
 * Sends a snapshot from the webcam to the web worker at the interval of ctx.faceDetectionInterval in ctx.onVideoStarted().
 * It sends the image as a ImageData format to jsfeat and url string if Amazon Rekognition is used.
 */
function sendVideoFrameToWorker(ctx) {
    let videoFrameData = null;

    if (ctx.faceDetectionAlgorithm === ctx.faceDetectionAlgorithmOptions.jsfeat) {
        ctx.videoFrameImageData = ctx.worldData.Utils.getWebcamInputImageData(ctx.videoInput, ctx.canvasInputContext, ctx.ca
        videoFrameData = {
            'cmd': 'sendVideoFrameDataToJsfeat',
            'msg': ctx.videoFrameImageData
        }
    } else if (ctx.faceDetectionAlgorithm === ctx.faceDetectionAlgorithmOptions.rekognition) {
        ctx.videoFrameInputString = ctx.worldData.Utils.createWebcamFeedURL(ctx.videoInput, ctx.canvasInput, ctx.canvasInput

        videoFrameData = {
            'cmd': 'sendVideoFrameStringToRekognition',
            'msg': ctx.videoFrameInputString
        }
    }

    ctx.worker.postMessage(videoFrameData);
}

/**
 * Stops the webcam and cleans up the canvas used to draw the video when the webcam is stopped.
 */
function stopVideo(ctx) {
    ctx.worldData.Utils.stopCamera();

    ctx.canvasInputContext.clearRect(0, 0, ctx.canvasInput.width, ctx.canvasInput.height);
    ctx.canvasOutputContext.clearRect(0, 0, ctx.canvasOutputWidth, ctx.canvasOutputHeight);

    window.clearInterval(ctx.sendVideoFrameToWorkerAtInterval);
}
```

Face detection



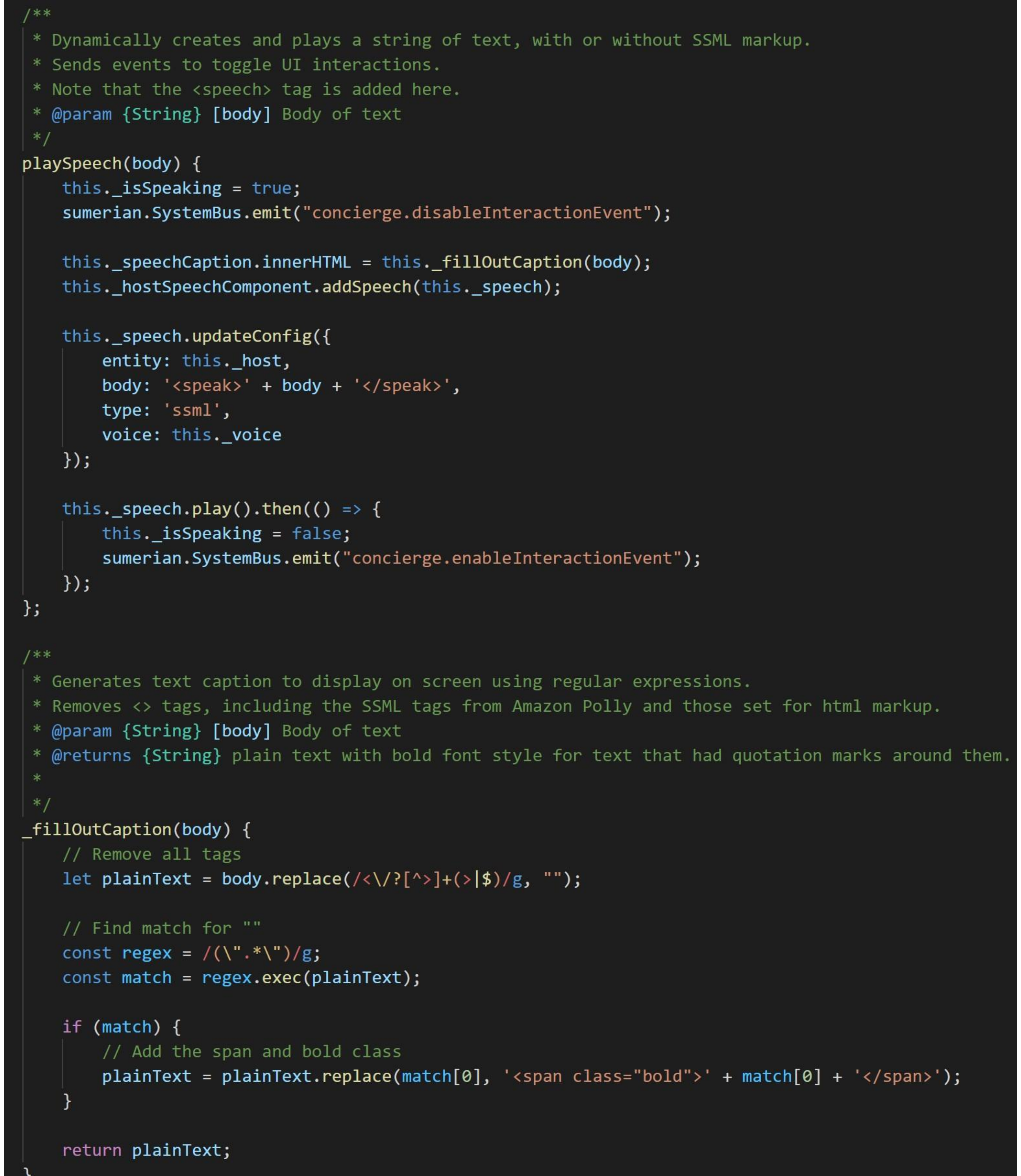
```
/**
 * Face detection
 *
 * The face detection algorithms are run in the web worker and communicate back to the main thread with face detection or emotion analysis results.
 *
 * (1) Use jsfeat to detect if there is a face in the screen.
 * (2) Switch to Amazon Rekognition to get emotion analysis if there is a face after the user greets the Host, so the Host can respond to the emoti
 * (3) Then revert back to using jsfeat after setInitialGreeting() as the emotion analysis is no longer used in this scene.
 * (4) If there is no face, then keep using jsfeat
 *
 * These decision points are marked in the appropriate functions in this file.
 *
 *              |--> (2) (if face exists) -- Amazon Rekognition ------------------> (3) jsfeat
 *              |                            (while ctx.requestEmotion = true)
 * (1) jsfeat --|
 *              |-----------------------------------------------------------------> (4) jsfeat
 */

// Use the video frame's input string for Amazon Rekognition.
ctx.videoFrameInputString = null;

// Use the video frame's image data for jsfeat.
ctx.videoFrameImageData = null;

ctx.emotion = null;

// Use this flag to indicate when emotion analysis is needed.
ctx.requestEmotion = false;

// Use this flag to indicate when show music.
ctx.showMusic = false;

ctx.jsfeat = {
    exit() {
        ctx.videoFrameImageData = null;
    }
}

ctx.rekognition = {
    exit() {
        ctx.videoFrameInputString = null;
    }
}

// Two options for face detection algorithm. Only one is used at any time.
ctx.faceDetectionAlgorithmOptions = {
    jsfeat: ctx.jsfeat,
    rekognition: ctx.rekognition
}
```

Voice commands



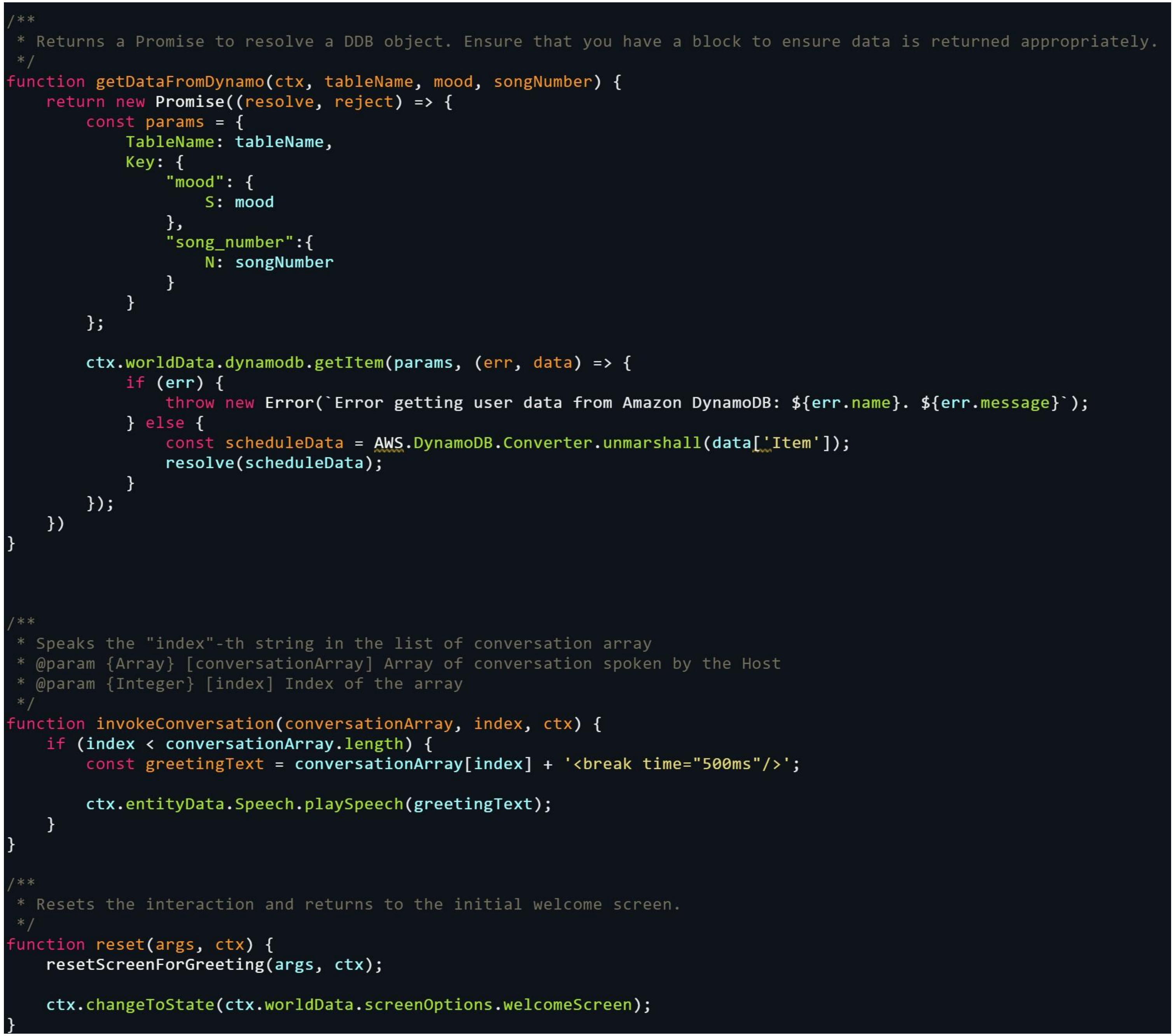
```
 * Uses the intent names to handle events.
 * @param {Object} [data] The data returned from Amazon Lex
 */
ctx.onLexResponse = (data) => {
    const msg = data.message;

    switch(data.intentName) {
        case "Greeting":
            ctx.startButton.click();
            break;
        case "Info":
            ctx.worldData.infoButton.click();
            break;
        case "CloseInfo":
            // Panel toggling is handled by the State Machine behavior "Info Screen Behavior".
            ctx.worldData.closeInfoButton.click();
            break;
        case "ThankYou":
            ctx.entityData.Speech.playSpeech(msg);
            break;
        case "CheckExpression":
            ctx.checkFaceButton.click();
            break;
        default:
            const index = ctx.worldData.Utils.getRandomInt(clarificationSpeeches.length);
            ctx.entityData.Speech.playSpeech(clarificationSpeeches[index]);
    }
}

/**
 * Handles the event emitted from the "Toggle Debug Panel Behavior" in the State Machine.
 */
ctx.onShowDebugPanel = () => {
    ctx.worldData.isShowingDebugPanel = true;

};

/**
 * Handles the event emitted from the "Toggle Debug Panel Behavior" in the State Machine.
 */
ctx.onHideDebugPanel = () => {
    ctx.worldData.isShowingDebugPanel = false;
};

/**
 * Change to the Greeting screen.
 */
ctx.startGreeting = () => {
    ctx.changeToState(ctx.worldData.screenOptions.greetingScreen);
}
```

Speech response



```
/**
 * Dynamically creates and plays a string of text, with or without SSML markup.
 * Sends events to toggle UI interactions.
 * Note that the <speech> tag is added here.
 * @param {String} [body] Body of text
 */
playSpeech(body) {
    this._isSpeaking = true;
    sumerian.SystemBus.emit("concierge.disableInteractionEvent");

    this._speechCaption.innerHTML = this._fillOutCaption(body);
    this._hostSpeechComponent.addSpeech(this._speech);

    this._speech.updateConfig({
        entity: this._host,
        body: '<speak>' + body + '</speak>',
        type: 'ssml',
        voice: this._voice
    });

    this._speech.play().then(() => {
        this._isSpeaking = false;
        sumerian.SystemBus.emit("concierge.enableInteractionEvent");
    });
};

/**
 * Generates text caption to display on screen using regular expressions.
 * Removes <> tags, including the SSML tags from Amazon Polly and those set for html markup.
 * @param {String} [body] Body of text
 * @returns {String} plain text with bold font style for text that had quotation marks around them.
 *
 */
_fillOutCaption(body) {
    // Remove all tags
    let plainText = body.replace(/<\/?[^>]+(>|$)/g, "");

    // Find match for ""
    const regex = /(\".*\")/g;
    const match = regex.exec(plainText);

    if (match) {
        // Add the span and bold class
        plainText = plainText.replace(match[0], '<span class="bold">' + match[0] + '</span>');
    }

    return plainText;
}
```

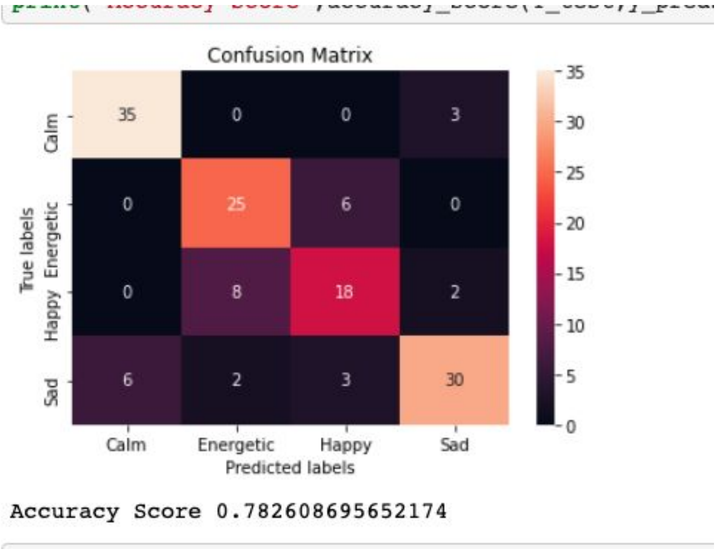
Data from DynamoDB



```
/**
 * Returns a Promise to resolve a DDB object. Ensure that you have a block to ensure data is returned appropriately.
 */
function getDataFromDynamo(ctx, tableName, mood, songNumber) {
    return new Promise((resolve, reject) => {
        const params = {
            TableName: tableName,
            Key: {
                "mood": {
                    S: mood
                },
                "song_number":{
                    N: songNumber
                }
            }
        };

        ctx.worldData.dynamodb.getItem(params, (err, data) => {
            if (err) {
                throw new Error(`Error getting user data from Amazon DynamoDB: ${err.name}. ${err.message}`);
            } else {
                const scheduleData = AWS.DynamoDB.Converter.unmarshall(data['Item']);
                resolve(scheduleData);
            }
        });
    })
}



/**
 * Speaks the "index"-th string in the list of conversation array
 * @param {Array} [conversationArray] Array of conversation spoken by the Host
 * @param {Integer} [index] Index of the array
 */
function invokeConversation(conversationArray, index, ctx) {
    if (index < conversationArray.length) {
        const greetingText = conversationArray[index] + '<break time="500ms"/>';

        ctx.entityData.Speech.playSpeech(greetingText);
    }
}

/**
 * Resets the interaction and returns to the initial welcome screen.
 */
function reset(args, ctx) {
    resetScreenForGreeting(args, ctx);

    ctx.changeToState(ctx.worldData.screenOptions.welcomeScreen);
}
```

Mood Classification Model

- Features gathered from 800 songs of Spotify using Spotipy library.
- Model based on Keras & Tensorflow.
- Model evaluation performed using K fold cross validation.
- Given a song Id, trained model can classify song into one of 4 moods - Calm/Energy/Happy/Sad.
- Contains sample code for classifying songs for a spotify artist or a library.



Confusion Matrix

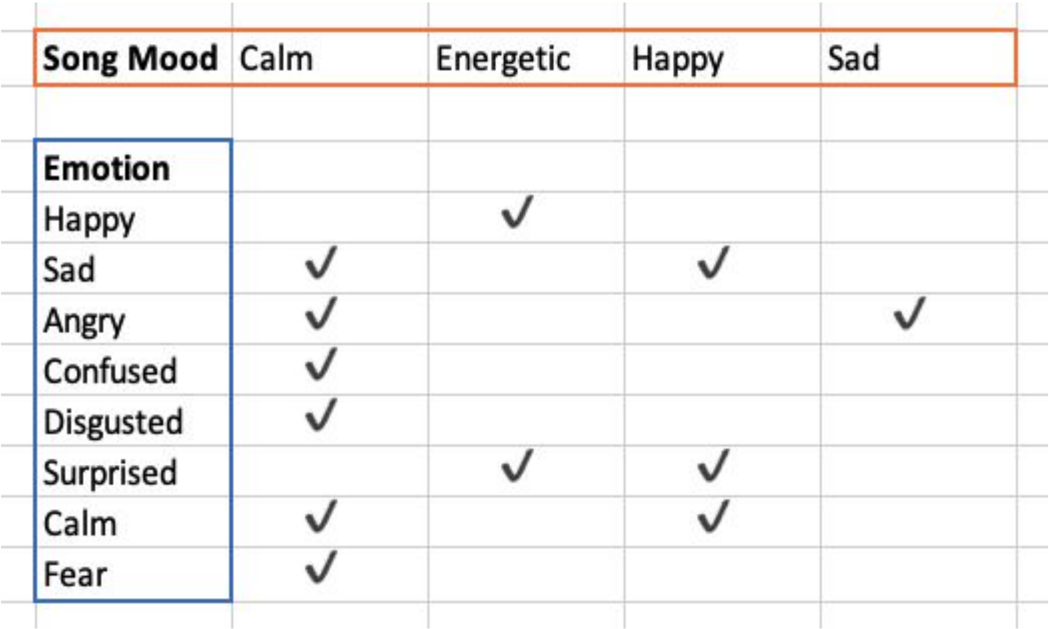
True labels \ Predicted labels	Calm	Energetic	Happy	Sad
Calm	35	0	0	3
Energetic	0	25	6	0
Happy	0	8	18	2
Sad	6	2	3	30

```
Accuracy Score 0.782608695652174
```

Code: https://github.com/UNT-5214-P4/Elevate/tree/main

Emotion Recognition <-> Song Mood

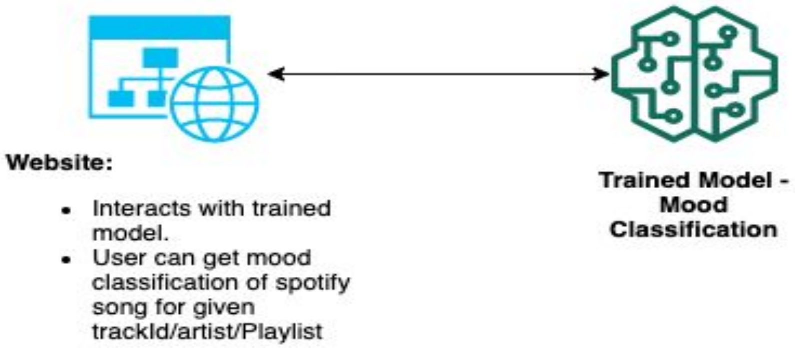
- Below image gives possible classifications of emotion, song mood.
- Given emotion we use the mapping below to play a song from mood classified playlist.



Song Mood	Calm	Energetic	Happy	Sad
Emotion				
Happy		✓		
Sad	✓		✓	
Angry	✓			✓
Confused	✓			
Disgusted	✓			
Surprised		✓	✓	
Calm	✓		✓	
Fear	✓			

Using the Model on Python Flask

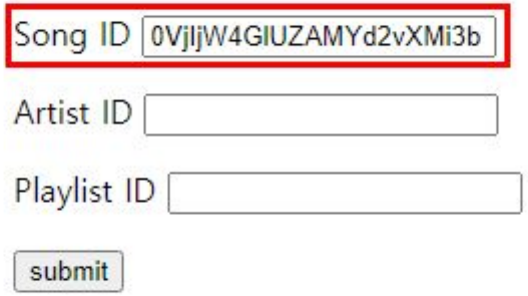
- Saving the mood classification model as a h5 file. (ex. model.h5)
- Loading the model on python flask
- Predicting a mood of a song and displaying information of the song



Website:

- Interacts with trained model.
- User can get mood classification of spotify song for given trackId/artist/Playlist

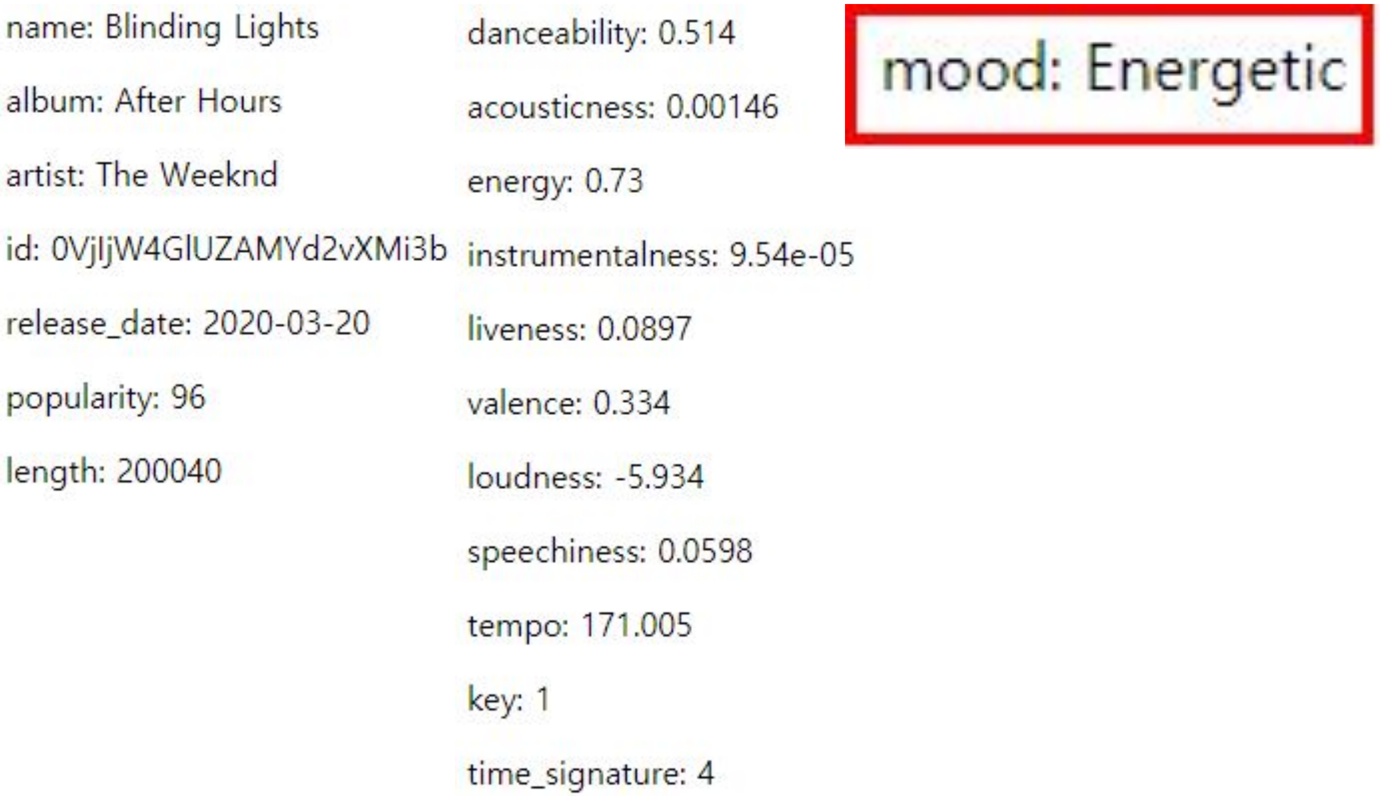
Trained Model - Mood Classification



Song ID 0VjIjW4GIUZAMYd2vXMi3b

Artist ID

Playlist ID



name: Blinding Lights

album: After Hours

artist: The Weeknd

id: 0VjIjW4GIUZAMYd2vXMi3b

release_date: 2020-03-20

popularity: 96

length: 200040

danceability: 0.514

acousticness: 0.00146

energy: 0.73

instrumentalness: 9.54e-05

liveness: 0.0897

valence: 0.334

loudness: -5.934

speechiness: 0.0598

tempo: 171.005

key: 1

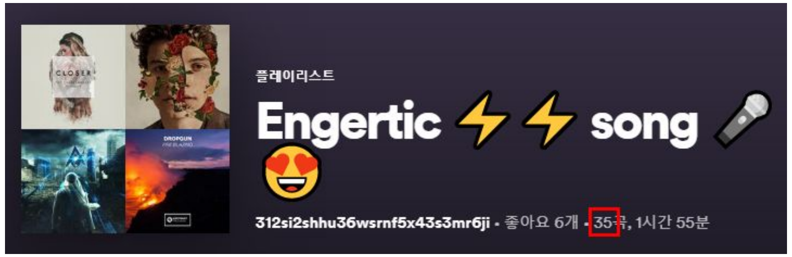
time_signature: 4

mood: Energetic

Code:
https://github.com/UNT-5214-P4/Elevate/tree/flask

Using the Model on Python Flask

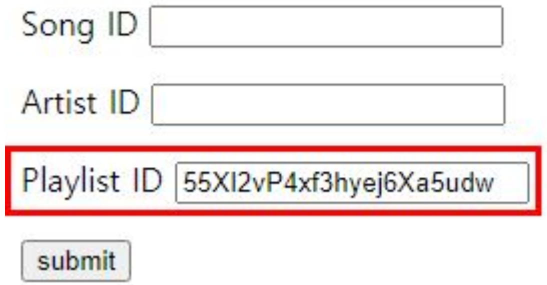
- Saving the mood classification model as a h5 file. (ex. model.h5)
- Loading the model on python flask
- Predicting a mood of songs in the playlist



플레이리스트

Engertic ⚡⚡ song 🎤😍

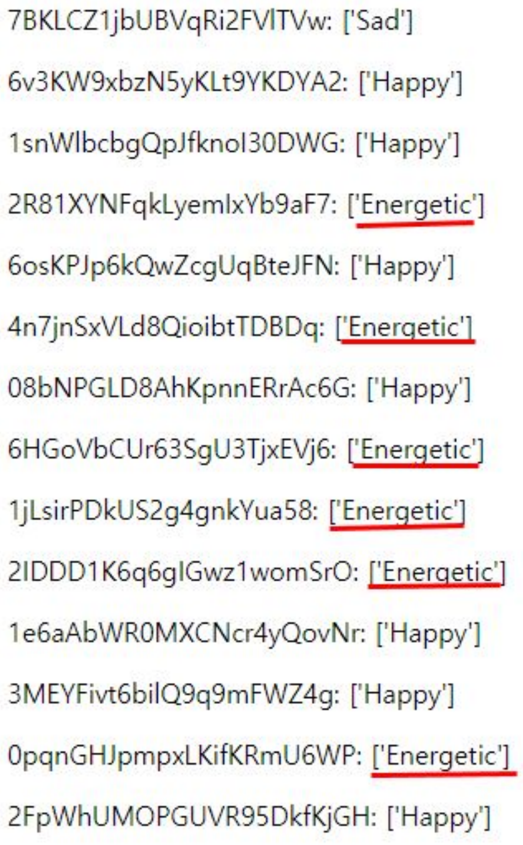
312si2shhu36wsrnf5x43s3mr6ji • 좋아요 6개 • 35곡, 1시간 55분



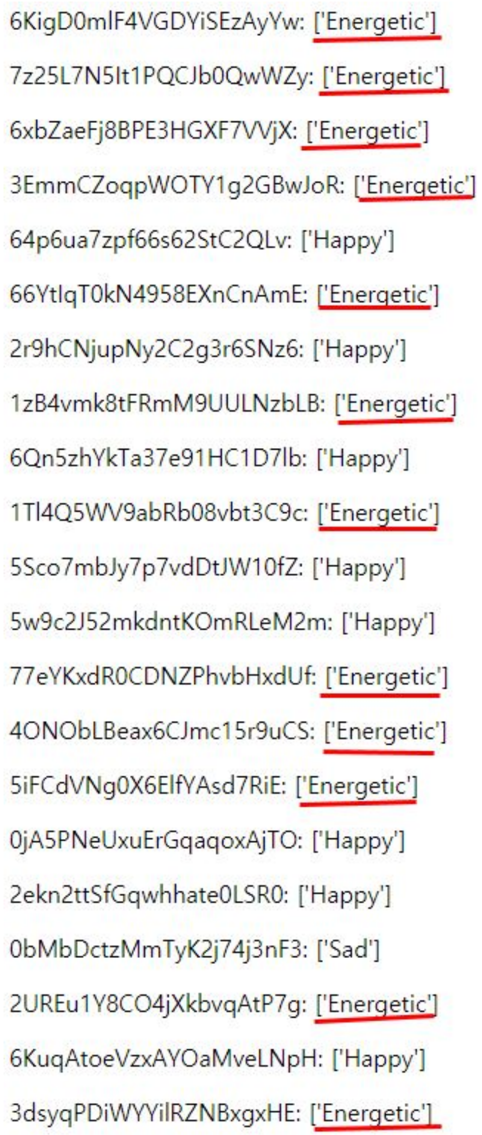
Song ID

Artist ID

Playlist ID 55XI2vP4xf3hyej6Xa5udw



```
7BKLCZ1jbUBVqRi2FVlTVw: ['Sad']
6v3KW9xbzN5yKLt9YKDYA2: ['Happy']
1snWlbcbgQpJfknoI30DWG: ['Happy']
2R81XYNFqkLyemlxYb9aF7: ['Energetic']
6osKPJp6kQwZcgUqBteJFN: ['Happy']
4n7jnSxVLd8QioibtTDBDq: ['Energetic']
08bNPGLD8AhKpnnERrAc6G: ['Happy']
6HGoVbCUr63SgU3TjxEVj6: ['Energetic']
1jLsirPDkUS2g4gnkYua58: ['Energetic']
2IDDD1K6q6gIGwz1womSrO: ['Energetic']
1e6aAbWR0MXCNcr4yQovNr: ['Happy']
3MEYFivt6bilQ9q9mFWZ4g: ['Happy']
0pqnGHJpmpxLKifKRmU6WP: ['Energetic']
2FpWhUMOPGUVR95DkfKjGH: ['Happy']
```



```
6KigD0mlF4VGDYiSEzAyYw: ['Energetic']
7z25L7N5It1PQCJb0QwWZy: ['Energetic']
6xbZaeFj8BPE3HGXF7VVjX: ['Energetic']
3EmmCZoqpWOTY1g2GBwJoR: ['Energetic']
64p6ua7zpf66s62StC2QLv: ['Happy']
66YtlqT0kN4958EXnCnAmE: ['Energetic']
2r9hCNjupNy2C2g3r6SNz6: ['Happy']
1zB4vmk8tFRmM9UULNzbLB: ['Energetic']
6Qn5zhYkTa37e91HC1D7lb: ['Happy']
1Tl4Q5WV9abRb08vbt3C9c: ['Energetic']
5Sco7mbJy7p7vdDtJW10fZ: ['Happy']
5w9c2J52mkdntKOmRLeM2m: ['Happy']
77eYKxdR0CDNZPhvbHxdUf: ['Energetic']
4ONObLBeax6CJmc15r9uCS: ['Energetic']
5iFCdVNg0X6ElfYAsd7RiE: ['Energetic']
0jA5PNeUxuErGqaqoxAjTO: ['Happy']
2ekn2ttSfGqwhhate0LSR0: ['Happy']
0bMbDctzMmTyK2j74j3nF3: ['Sad']
2UREu1Y8CO4jXkbvqAtP7g: ['Energetic']
6KuqAtoeVzxAYOaMveLNpH: ['Happy']
3dsyqPDiWYYilRZNBxgxHE: ['Energetic']
```

Technologies

- AWS Sumerian, Rekognition, Lex, Polly, DynamoDB
- Flask
- Keras
- Tensorflow
- Sklearn

Milestones

- Milestone 1 (Due on 11/11/2020)
  - Gather tutorial
  - Setup workplace, github
  - Work on P4 Proposal
- Milestone 2 (Due on 11/18/2020)
  - Have the two major components - emotion recognition & song classification completed.
  - Work on P4 Video Update
- Milestone 3 (Due on 11/25/2020)
  - Integrate and testing.
  - Work on P3 Report and Video Presentation.

Resources & Related Projects

Amazon Sumerian Concierge Experience
https://docs.sumerian.amazonaws.com/articles/concierge-experience/

Predicting the Music Mood of a Song with Deep Learning.
https://towardsdatascience.com/predicting-the-music-mood-of-a-song-with-deep-learning-c3ac2b45229e

Future Improvements:

- Convert sumerian website into standalone app.
- Convert this to smart skill which can be integrated into alexa.
- Integrate website with a model that can also classify local music mp3/.wav files.
- Ability for the user to be able configure the playlists, other music stream providers.