令和 6 年度卒業研究報告書

FukuMeet の開発

情報技術科　及川結加　齋藤龍聖　米澤悠貴

指導教員　　高橋強

目次

第1章　　研究の背景と目的

オンラインショップで洋服を購入する際、実際に手に取って試着ができないため、イメージと異なる商品を選んでしまうことがある。こうした失敗を防ぐために、ユーザーがパソコンのカメラの前で、好きな服やアクセサリーの画像を画面上で重ねることで、実際の試着イメージを確認することができる「FukuMeet」というシステムを開発してユーザーの買い物をサポートしたいと考えた。

本研究では授業で学んだ知識の応用と新たな技術に加えて、システム開発のプロセスや問題解決力、メンバーや指導教員との協力を通じてコミュニケーション力やプレゼンテーション力も高めていくことを目的とする。

第 2 章　FukuMeet の概要

2.1 FukuMeet とは

オンラインショップで洋服を購入する際、「実際に着てみるとイメージと違った」と感じることが多くの人々にとって課題となっている。サイズ感やデザインが写真と異なって見えることがあり、購入後に後悔するケースも少なくない。FukuMeet は、このような課題を解決するために開発したオンライン試着システムである。カメラを使ってリアルタイムで服やアクセサリーをオーバーレイ表示し、購入前に試着イメージを確認できるようにすることで、オンラインショッピングの不安を軽減する。

2.2 FukuMeet の操作手順

【　画像をカタログに追加したい場合　】

① ［画像を追加］ボタンを押す。

画像 1.2.1　FukuMeet の操作手順

② ファイルから追加したい画像を追加し項目すべて入力し、［アップロード］ボタンを押す。

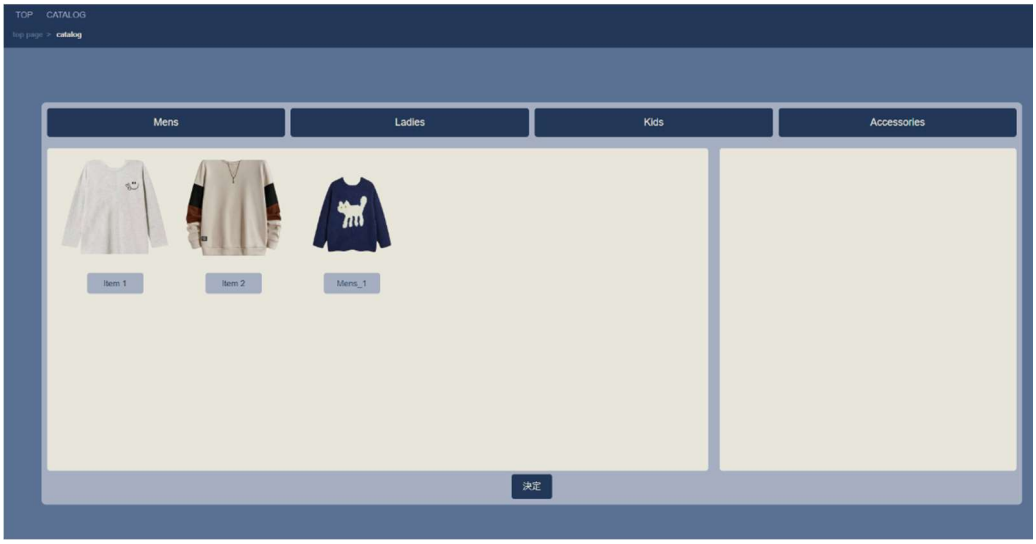
画像 1.2.2　FukuMeet の操作手順

【　試着したい場合　】

① ［START］ボタンを押す。

画像 1.2.3　FukuMeet の操作手順

② 服を選び［決定］ボタンを押す。



画像 1.2.4　FukuMeet の操作手順

③ 試着画面で試着をする。

画像 1.2.5　FukuMeet の操作手順

- 緑色のマイクボタンを押すことで音声入力を利用してアイテムの大きさや位置を調整することが可能である。
- 音声入力がうまくいかない場合などは、スライダーを使用してアイテムの位置を調整することが可能である。
- 試着状況を写真で取りたい場合は［写真を撮る］ボタンを押し、写真撮影が可能である。

第 3 章　開発環境と構築

3.1 開発環境

開発環境および使用ライブラリを表 1 に示す。

表 2.1.1　開発環境および使用ライブラリ

OS	Windows
ソフトウェア	Visual Studio Code
使用言語	Python,HTML,CSS,JavaScript
ライブラリ	OpenCV(CV2), Mediapipe, rembg Flask(Response, render, template, request, Jsonify, send_from_directiory)

3.2 プログラミング言語とライブラリ

3.2.1　プログラミング言語

・Python

Python は画像認識やポーズ推定、リアルタイム処理に特化したライブラリやフレームワークが豊富なプログラム言語である。「FukuMeet」では、レイアウト以外の全体の制御で Python を使用した。

・HTML（Hyper Text Markup Language）

HTML は web ページのテキスト情報の構成などを製作するためのマークアップ言語である。「FukuMeet」では、web ページの文章作成で使用した。

・CSS（Cascading Style Sheets）

CSS は web ページのサイズや色、レイアウトなどを製作するためのプログラム言語である。「FukuMeet」では、web ページでの全体的なレイアウトで使用した。

・JavaScript

JavaScript は web ページの動的な機能を作成するためのプログラム言語である。「FukuMeet」では、web ページでのポップアップウィンドウやスライダーなどの動的な機能を作成する際に使用した。

3.2.2　ライブラリ

・OpenCV（CV2）

OpenCV(CV2)は、画像やビデオの解析、認識、変換などを行うオープンソースライブラリである。「FukuMeet」では、画像の反転や動画の撮影、保存、画像の重ね合わせなどの処理で使用している。

・Mediapipe

Mediapipe はリアルタイムでポーズ推定や顔認識、動きの追跡、物体検出などを行うことができるライブラリである。「FukuMeet」ではポーズ推定を利用して、リアルタイムで両肩や左右の腰の位置を特定する際に使用している。

・rembg

rembg は画像の背景を自動で透過処理するライブラリである。「FukuMeet」では好きな画像を追加する際に、背景透過の処理で使用している。

・Flask

Flask は Web アプリケーションの作成やフォーム処理、HTML テンプレートのレンダリングが可能な軽量フレームワークである。「FukuMeet」ではフォーム処理、HTML テンプレートのレンダリング、OpenCV や Mediapipe などで行った処理を Web アプリケーションとして表示させている。

・Pillow (PIL)

Pillow は画像のサイズ変更や回転、フォーマットの返還などを行うことができるライブラリである。「FukuMeet」では、ユーザーがアップロードした画像のリサイズを行う際に使用している。

・Math

Math は対数や三角関数・浮動小数点数・平方根・絶対値などの複雑な計算を行う関数を集めたライブラリである。「FukuMeet」では、服とネックレスの位置計算をする際に使用している。

・Datetime

Datetime は現在の日時を取得したり、指定したフォーマットで日時を変換することができるライブラリである。「FukuMeet」では、撮影した画像のファイル名に日時を使う際の処理に使用している。

・Time

処理を特定の時間だけ一時停止することができるライブラリである。「FukuMeet」では、撮影前のカウントダウン時の処理に使用している。

・JSON

JSON は JSON 形式でデータを保存したり、読み込んだりするためのライブラリ。「FukuMeet」では、productData.json のカタログの情報を管理する際に使用している。

3.3 システムの環境構築

3.3.1Visual Studio Code の導入

FukuMeet 用の Windows PC に導入する。

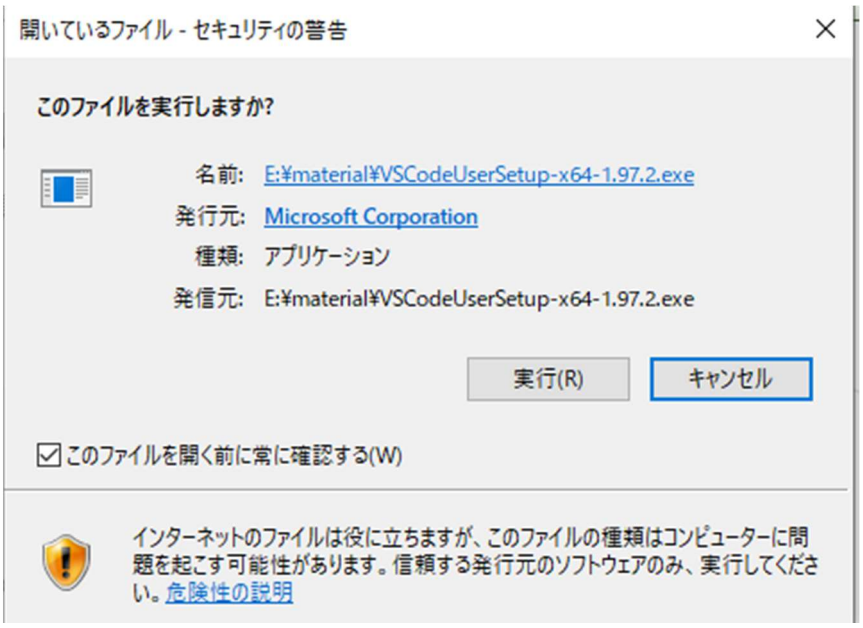
①Visual Studio Code のダウンロードページの Download for Windows をクリックする。

画像 3.3.1　Visual Studio Code のダウンロードページ画面

②ダウンロードした Visual Studio Code の exe ファイルをダブルクリックで実行し、Visual Studio Code をインストールする。

画像 3.3.2　Visual Studio Code のダウンロードした exe ファイル

③インストールする際にセキュリティ警告画面が表示されるので「実行」をクリックする。



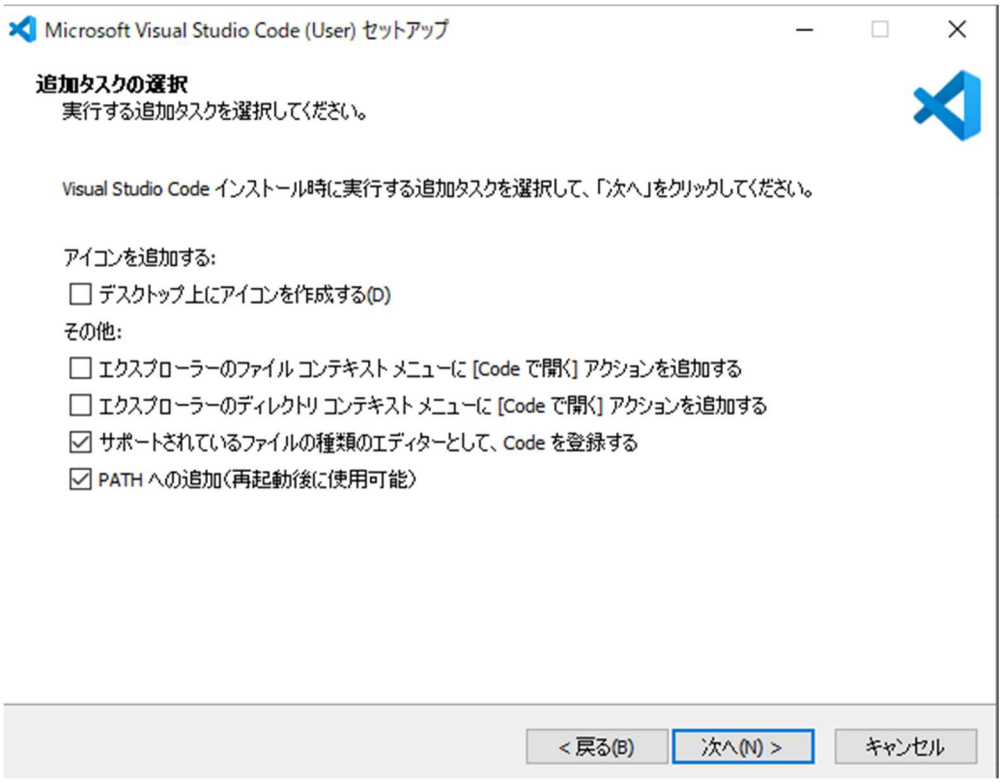
画像 3.3.3　Visual Studio Code のセキュリティ警告画面

④「同意する」を選択し、「次へ」をクリックする。

画像 3.3.4　Visual Studio Code の同意確認画面

⑤フォルダー指定は各自選択する。

必要であれば「デスクトップ画面にアイコンを作成する」にチェックを入れ、「次へ」をクリックする。



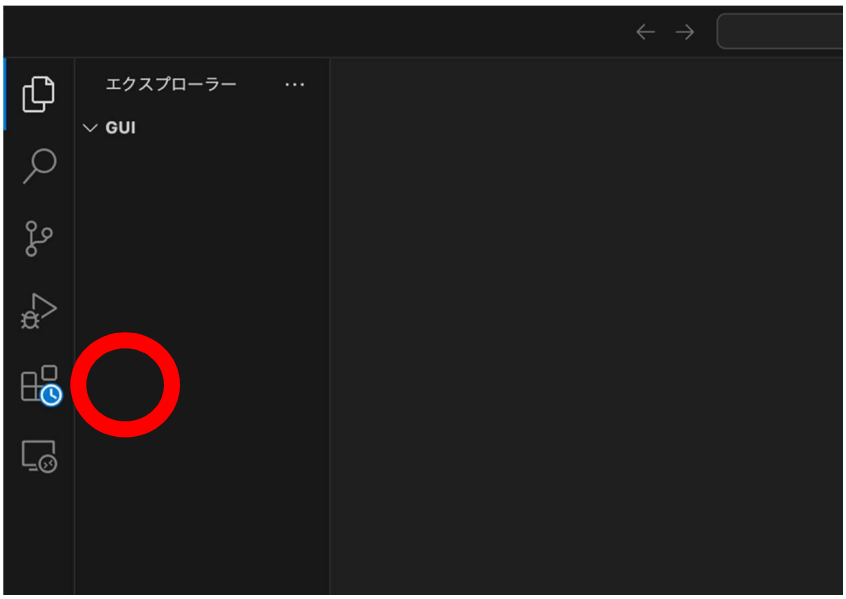
画像 3.3.5　Visual Studio Code の追加タスク画面

⑦「インストール」をクリックする。

Visual Studio Code の導入終了である。

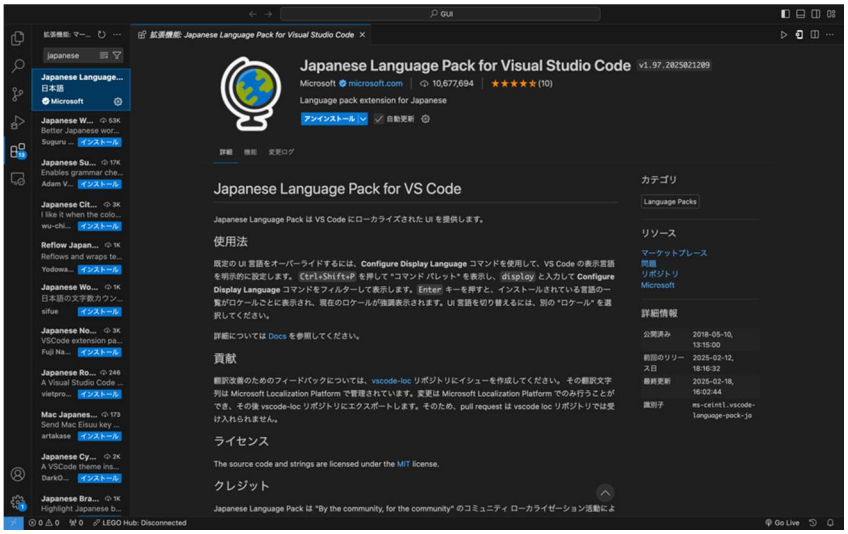
画像 3.36~画像 3.3.9 に各拡張機能の追加と python の設定方法を示す。

①Visual Studio Code を起動し、左下の「Extension」をクリックする。



画像 3.3.6　Visual Studio Code の起動画面

② 検索欄に「japanese」を入力し「Japanese Language Pac for VS Code」をインストールする。

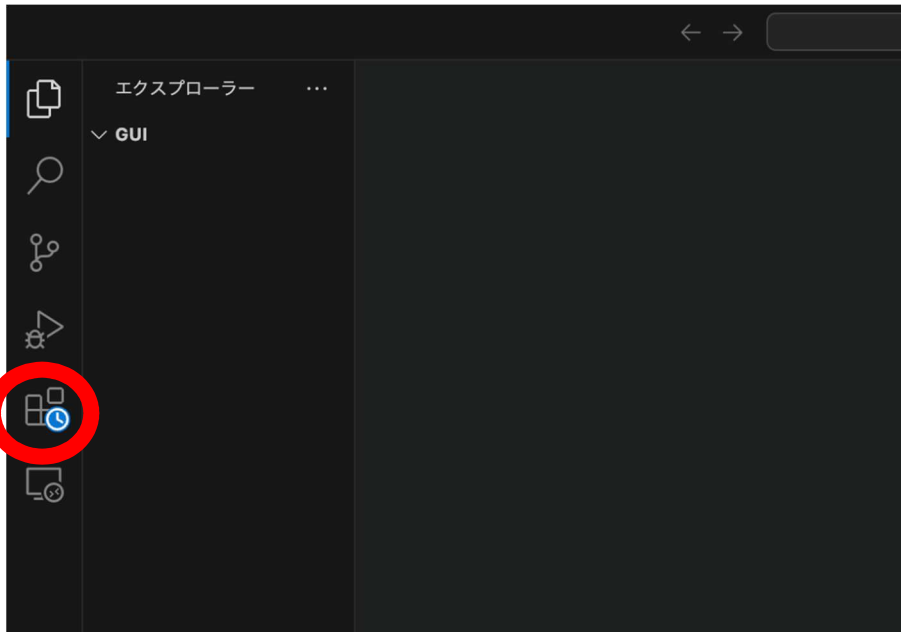


画像 3.3.7　Visual Studio Code の日本語導入画面

③インストール後、右下に表示される「Change Language and Restart」ボタンをクリックして、Visual Studio Code を再起動する。

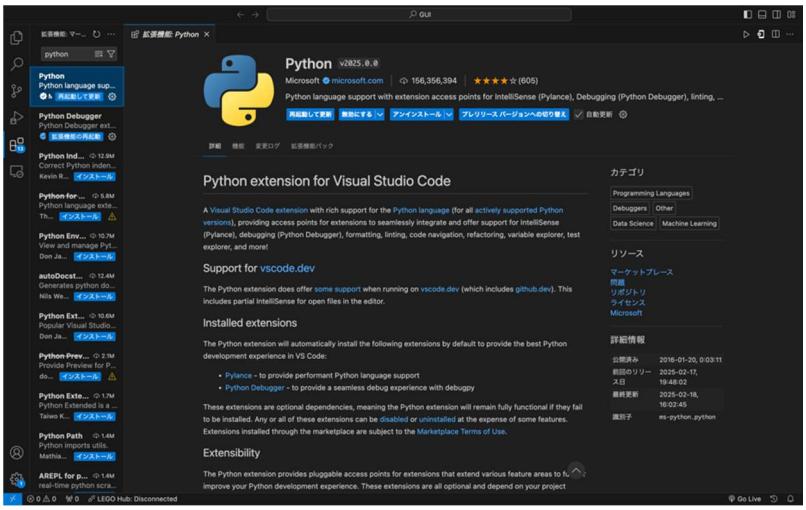
.3.2Python のインストール

①赤丸の拡張機能をクリックし、検索欄で「python」と入力する。



画像 3.3.8　python のインストール

②python をインストールし、設定は完了である。



画像 3.3.9　python のインストール

3.3 ライブラリのインストール

3.3.1　ライブラリのインストールソフト

表 2.3.1　インストールソフト

Windows	コマンドプロンプト
MacBook	ターミナル

※以下ターミナルを使用する

3.3.2　ライブラリのインストールコマンド一覧

表 2.3.2 インストールライブラリ一覧

Flask	pip install flask
Werkzeug	pip install werkzeug
PIL	pip install pillow
OpenCV	pip install opencv-python-headless
MediaPipe	pip install mediapipe
rembg	pip install rembg

※Math、Datetime、os、JSON、time は Python の標準ライブラリに含まれているため、インストールする必要はない。

第 4 章　システム構築

4.1 ディレクトリ構成図とファイルの役割

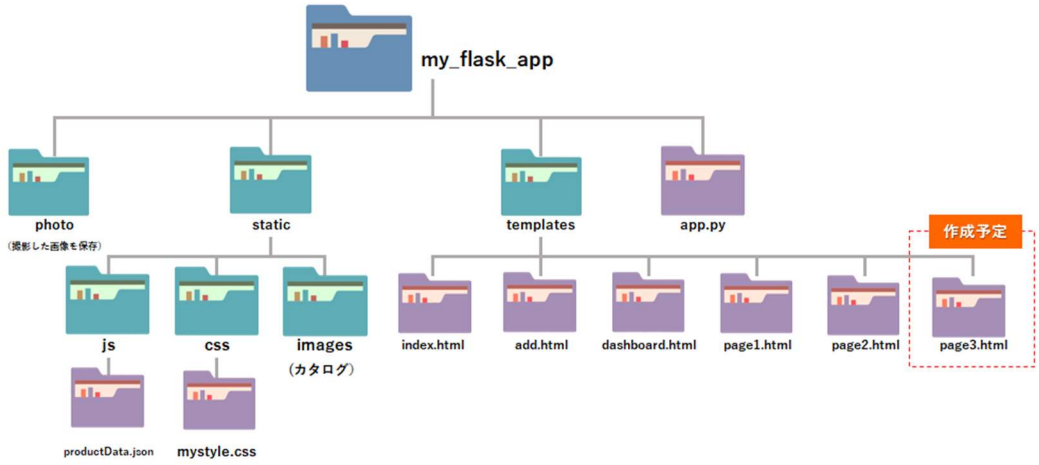


図 3.1 ディレクトリ構成図

表 3.1　ファイルの役割

photo	撮影した画像の保存先
productData.json	服とアクセサリーの情報と画像パスの格納先
mystyle.css	全体のデザインとレイアウト
images	服とアクセサリーの画像の格納先
app.py	全体の制御処理
index.html	タイトルページ
add.html	新規カタログの追加ページ
dashboard.html	服とアクセサリーの選択ページ
page1.html	アクセサリーの画像の重ね合わせを行うページ
page2.html	服の画像の重ね合わせを行うページ

第 5 章　機能の詳細

■ 新規カタログの追加

入力した**商品名**や**カテゴリ**、選択した**画像**は Flask サーバーに送信後、背景透過を行いカテゴリごとに projectData.json に保存される。


```
"Mens": [
    {
        "image": "static/images/Mens/mens1.png",
        "name": "Item 1",
        "size": "M",
        "material": "綿",
        "price": "3000円",
        "color": "黒",
        "description": "快適なメンズシャツです。"
    },
    {
        "image": "static/images/Mens/mens2.png",
        "name": "Item 2",
        "size": "L",
        "material": "ポリエステル",
        "price": "3500円",
        "color": "青",
        "description": "スポーティなメンズジャケットです。"
    },
    "..."
]
```


画像 4.1　projectData.json

【プログラム解説】

```
<main class="add-1">
 <h1>アイテム画像の追加</h1>
 <form id="uploadForm" action="/upload_image" method="post”enctype="multipart/form-data">
①    <label for="image" class="custom-file-label">画像をファイルから選択</label>
     <input type="file" id="image" name="image" accept="image/*" required hidden>

       <br><br>

     <label for="category">カテゴリを選択</label>
②    <select id="category" name="category" required>
          <option value="Mens">Mens</option>
          <option value="Ladies">Ladies</option>
          <option value="Kids">Kids</option>
          <option value="Accessories">Accessories</option>
     </select>
       <br><br>

③    <label for="name">画像の名前</label>
     <input type="text" id="name" name="name" required>
       <br><br>

     <label for="description">画像の詳細</label>
④    <textarea id="description" name="description" rows="4" cols="50"required></textarea>
       <br><br>

     <canvas id="canvas" style="display: none;"></canvas>
     <button class="btn_26" type="submit">アップロード</button>
    </form>
</main>
```

add.html

① ローカルから画像の選択をする。

② カテゴリ（Mens、Ladies、Kids、Accessories）を選択する。

③ 画像の名前を入力する。

④ 画像の詳細を入力する。

これらのデータを Flask サーバー（app.py）に送信する。

⑦’

```
def make_background_transparent(image_path):
    # 画像を読み込む
    with open(image_path, 'rb') as input_file:
        input_data = input_file.read()

    # rembg で背景を透過
    output_data = remove(input_data)

    # 透過後の画像を保存
    transparent_image_path = image_path.replace(os.sep, '/').split('.')[-2] + '_transparent.png'
    with open(transparent_image_path, 'wb') as output_file:
        output_file.write(output_data)

    return transparent_image_path
```

⑤

```
# 画像をアップロードし、カテゴリに追加する処理
@app.route('/upload_image', methods=['POST'])
def upload_image():
    # フォームデータを受け取る
    image = request.files['image']
    category = request.form['category']
    name = request.form['name']
    description = request.form['description']
```

⑥

```
# 画像を保存する場所
    image_dir = os.path.join('static', 'images', category)
    if not os.path.exists(image_dir):
        os.makedirs(image_dir)

    # 画像ファイル名を保存
    image_filename = secure_filename(image.filename)
    image_path = os.path.join(image_dir, image_filename)

    # 画像を保存
    image.save(image_path)
```

⑦

```
    # 透過処理
    transparent_image_path = make_background_transparent(image_path)

    # JSON ファイルに保存する前にパスの区切り文字をスラッシュに統一
    transparent_image_path = transparent_image_path.replace(os.sep, '/')
```

⑧

```
    # 画像情報を JSON ファイルに保存
    if os.path.exists(IMAGE_PATHS_JSON):
        with open(IMAGE_PATHS_JSON, 'r', encoding='utf-8') as file:
            data = json.load(file)
    else:
        data = {}

    # カテゴリが存在しない場合は作成
    if category not in data:
        data[category] = []

    # 新しい画像の情報を追加
    data[category].append({
        'image': transparent_image_path,
        'name': name,
        'details': description
```

```
    })

    # JSON ファイルに保存
    with open(IMAGE_PATHS_JSON, 'w', encoding='utf-8') as file:
        json.dump(data, file, ensure_ascii=False, indent=4)

    response_text = (
        f"ファイル名: {image.filename}<br>"
        f"メッセージ: アップロードが成功しました！<br>"
        f"保存先: {image_path}"
    )

    time.sleep(5)

    # 5 秒後に index.html にリダイレクト
    seconds_until_redirect = 3

returnrender_template('index.html',message=response_text,
seconds=seconds_until_redirect)
```

app.py

⑤ フォームデータを受け取る。

⑥ 指定したカテゴリのフォルダに画像を保存する。

⑦ rembg を使用して画像の背景を透過する。(具体的な処理は⑦‘)

⑧ 画像を productData.json に保存する。

■ カタログページでの服やアクセサリーの表示・選択

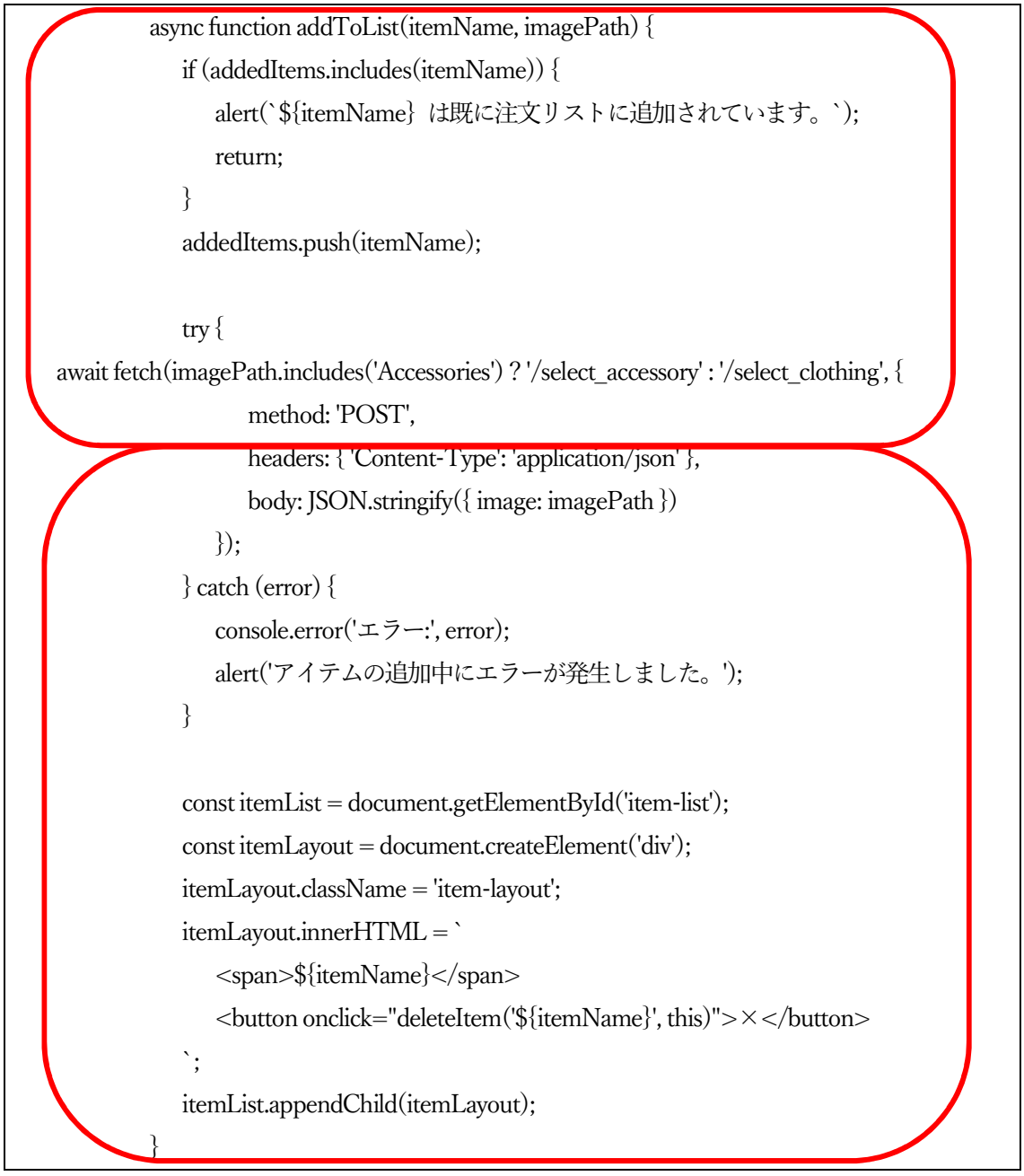
- projectData.json に格納している画像のデータを dashboard.html でカタログごとに表示させる。

画像 4.2　カタログページ

【プログラム解説】

①

```
async function fetchProductData() {
        try {
    const response = await fetch('{{ url_for('static', filename='js/productData.json') }}');
            if (response.ok) {
                productData = await response.json();
                updateProductBox('Mens');
            } else {
                console.error('Failed to load product data.');
            }
        } catch (error) {
            console.error('エラー:', error);
        }
    }
```



```
async function addToList(itemName, imagePath) {
    if (addedItems.includes(itemName)) {
        alert(`${itemName} は既に注文リストに追加されています。`);
        return;
    }
    addedItems.push(itemName);

    try {
await fetch(imagePath.includes('Accessories') ? '/select_accessory' : '/select_clothing', {
            method: 'POST',
            headers: { 'Content-Type': 'application/json' },
            body: JSON.stringify({ image: imagePath })
        });
    } catch (error) {
        console.error('エラー:', error);
        alert('アイテムの追加中にエラーが発生しました。');
    }

    const itemList = document.getElementById('item-list');
    const itemLayout = document.createElement('div');
    itemLayout.className = 'item-layout';
    itemLayout.innerHTML = `
        <span>${itemName}</span>
        <button onclick="deleteItem('${itemName}', this)">×</button>
    `;
    itemList.appendChild(itemLayout);
}
```

② ②

dashboard.html

① fetchProductData()を使って productData.json のデータを取得する。

② addToList で選択した商品をリストに追加する。

■ **カメラ映像の起動・表示**

- 選択した**画像**と**ファイル名**や**カテゴリ**などが Flask サーバーに送信されて、受信したデータからカテゴリの判定を行う。アクセサリーは page1.html、それ以外は page2.html に移動後、カメラが起動する。

画像 4.3　カメラの起動条件

- カメラから取得したフレームを Flask サーバーが**左右反転**の処理を行い、JPEG 形式に変換する。
- 処理されたフレームを HTML の img タグで連続的に送信することで実現している。

画像 4.4　カメラ起動後

【プログラム解説】

①

```
function updateProductBox(category) {
        const productBox = document.getElementById('product-box');
        productBox.innerHTML = '';

        productData[category].forEach(product => {
            const item = document.createElement('div');
            item.className = 'product-item';
            item.innerHTML = `
    <img src="${product.image}" alt="${product.name}" class="product-image"
                onmouseover="showDetails('${product.details}')"
                onmouseout="hideDetails()">
      <buttononclick="addToList('${product.name}',
       '${product.image}')">${product.name}</button>

     <button onclick="removeFromList('${product.name}')">削除</button>
            `;
            productBox.appendChild(item);
        });
```

①’

```
const confirmButton = document.getElementById('confirm-button');
confirmButton.onclick = category === 'Accessories'
    ? () => window.location.href = "{{ url_for('page1') }}"
    : () => window.location.href = "{{ url_for('page2') }}";
}
```

dashboard.html

②

②’

```
def generate_frames_page1():
    global necklace_percentage, vertical_adjustment, selected_accessory_path

    while True:
        ret, frame = cap.read()
        if not ret:
            break

        flipped_frame = cv2.flip(frame, 1)
        rgb_frame = cv2.cvtColor(flipped_frame, cv2.COLOR_BGR2RGB)
        results = pose.process(rgb_frame)
        ――――――――――――――――アクセサリーの重ね処理は後程解説――――――――――――――――
        # 映像を JPEG にエンコード
        _, buffer = cv2.imencode('.jpg', flipped_frame)
        frame = buffer.tobytes()
        yield (b'--frame¥r¥n'
               b'Content-Type: image/jpeg¥r¥n¥r¥n' + frame + b'¥r¥n')
```

```
def generate_frames_page2():
    global slider_values, selected_clothing_path

    while True:
        ret, frame = cap.read()
        if not ret:
            break

        flipped_frame = cv2.flip(frame, 1)
        rgb_frame = cv2.cvtColor(flipped_frame, cv2.COLOR_BGR2RGB)
        results = pose.process(rgb_frame)
        ――――――――――――――――――――服の重ね処理は後程解説――――――――――――――――――――
        # フレームを JPEG に変換
        _, buffer = cv2.imencode('.jpg', flipped_frame)
        frame = buffer.tobytes()

        yield (b'--frame¥r¥n'
               b'Content-Type: image/jpeg¥r¥n¥r¥n' + frame + b'¥r¥n')
```

③ ②’

app.py

① updateProductBox で選択した画像を Flask サーバーに送信し、

①’でカテゴリの判断を行っている。

（選択した画像のカテゴリが Accessories の場合は、②でカメラを起動し、Mens、Ladies、Kids の場合は、③でカメラを起動する。）

②’でカメラ映像の左右反転を行っている。

■ 服とアクセサリーのモニター表示

＜服の処理＞

服は **Mediapipe** で両肩と両腰の位置を取得して、**肩と腰の中心値**を求め、首から腰の範囲で表示する。

画像 4.5　服の処理

【プログラム解説】

①

```
# 肩と腰の座標取得
left_shoulder = (int(landmarks[mp.solutions.pose.PoseLandmark.LEFT_SHOULDER].x * w),
                 int(landmarks[mp.solutions.pose.PoseLandmark.LEFT_SHOULDER].y * h))
right_shoulder = (int(landmarks[mp.solutions.pose.PoseLandmark.RIGHT_SHOULDER].x * w),
                  int(landmarks[mp.solutions.pose.PoseLandmark.RIGHT_SHOULDER].y * h))
left_hip = (int(landmarks[mp.solutions.pose.PoseLandmark.LEFT_HIP].x * w),
            int(landmarks[mp.solutions.pose.PoseLandmark.LEFT_HIP].y * h))
right_hip = (int(landmarks[mp.solutions.pose.PoseLandmark.RIGHT_HIP].x * w),
             int(landmarks[mp.solutions.pose.PoseLandmark.RIGHT_HIP].y * h))
```

②

```
        # 中央位置の計算
        clothing_x = (left_shoulder[0] + right_shoulder[0]) // 2 + slider_values['horizontal']
    clothing_y = ((left_hip[1] + right_hip[1]) // 2 + (left_shoulder[1] + right_shoulder[1]) //
    2) // 2 - 15 + slider_values['vertical']
        shoulder_length = math.sqrt((right_shoulder[0] - left_shoulder[0]) ** 2 +
                                    (right_shoulder[1] - left_shoulder[1]) ** 2)
```

③

```
        # 服画像の読み込み
    clothing_img = cv2.imread(selected_clothing_path, cv2.IMREAD_UNCHANGED) if
      selected_clothing_path else None
        if clothing_img is not None:
            new_width = int(shoulder_length * slider_values['zoom'])
          resized_clothing_img = cv2.resize(clothing_img, (new_width, clothing_img.shape[0]))
            overlay_image(flipped_frame, resized_clothing_img, clothing_x, clothing_y)
        else:
            print("服の画像が選択されていません")
```

app.py

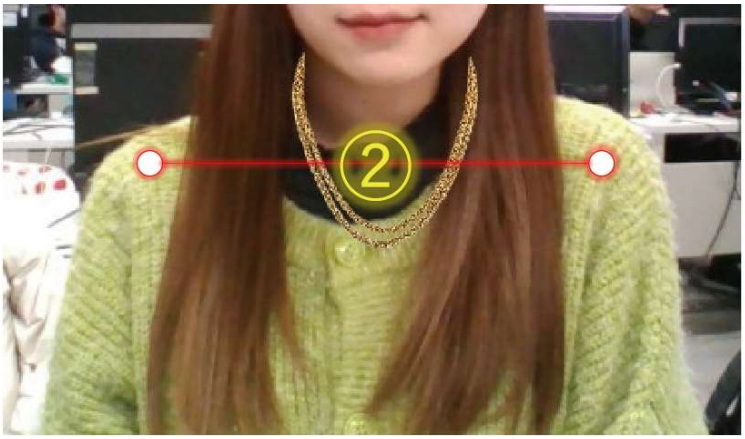
① 肩と腰の座標を取得する。

② math を使用して、右肩と右腰、左肩と左腰の中心をもとに中心を計算する。

③ overlay_image でカメラ映像を背景として画像を重ねる。

＜アクセサリーの処理＞

アクセサリーは両肩から**肩の中心値**を求めて首元に表示をする。



画像 4.6　アクセサリーの処理

【プログラム解説】

①
```
# 肩の座標
        left_shoulder = (int(landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER].x * w),
                         int(landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER].y * h))
        right_shoulder = (int(landmarks[mp_pose.PoseLandmark.RIGHT_SHOULDER].x * w),
                          int(landmarks[mp_pose.PoseLandmark.RIGHT_SHOULDER].y * h))
```

②
```
# ネックレスの位置とサイズ計算
neck_x = (left_shoulder[0] + right_shoulder[0]) // 2 + slider_values['horizontal']
neck_y = (left_shoulder[1] + right_shoulder[1]) // 2 - 20 + vertical_adjustment
shoulder_length = math.sqrt((right_shoulder[0] - left_shoulder[0]) ** 2 +
                            (right_shoulder[1] - left_shoulder[1]) ** 2)
```

③
```
# ネックレス画像のサイズ調整と重ね合わせ
necklace_img = cv2.imread(selected_accessory_path, cv2.IMREAD_UNCHANGED)
if necklace_img is not None:
    new_width = int(shoulder_length * necklace_percentage)
    new_height = int(necklace_img.shape[0] * (new_width / necklace_img.shape[1]))
    resized_necklace_img = cv2.resize(necklace_img, (new_width, new_height))
    overlay_image(flipped_frame, resized_necklace_img, neck_x, neck_y)
```

app.py

① 両肩の座標を取得する。

② 両肩の中心を計算する。

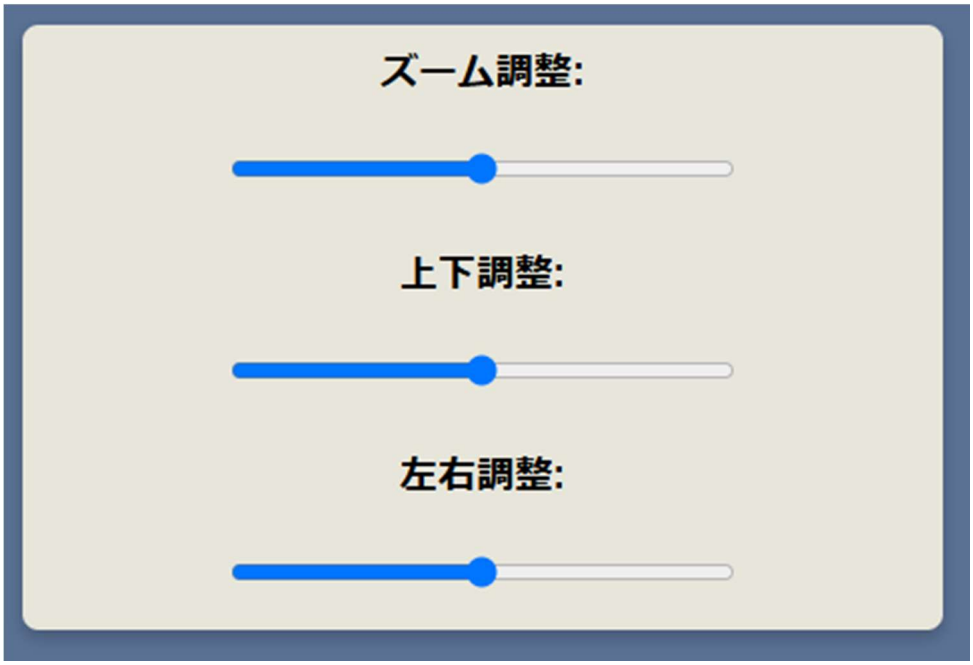
③ overlay_image でカメラ映像を背景として画像を重ねる。

表示中は横を向くと、**画像の横幅**が自動で調整される。

画像 4.7　横幅の調整

■ 画像の位置や大きさを調整するスライダー

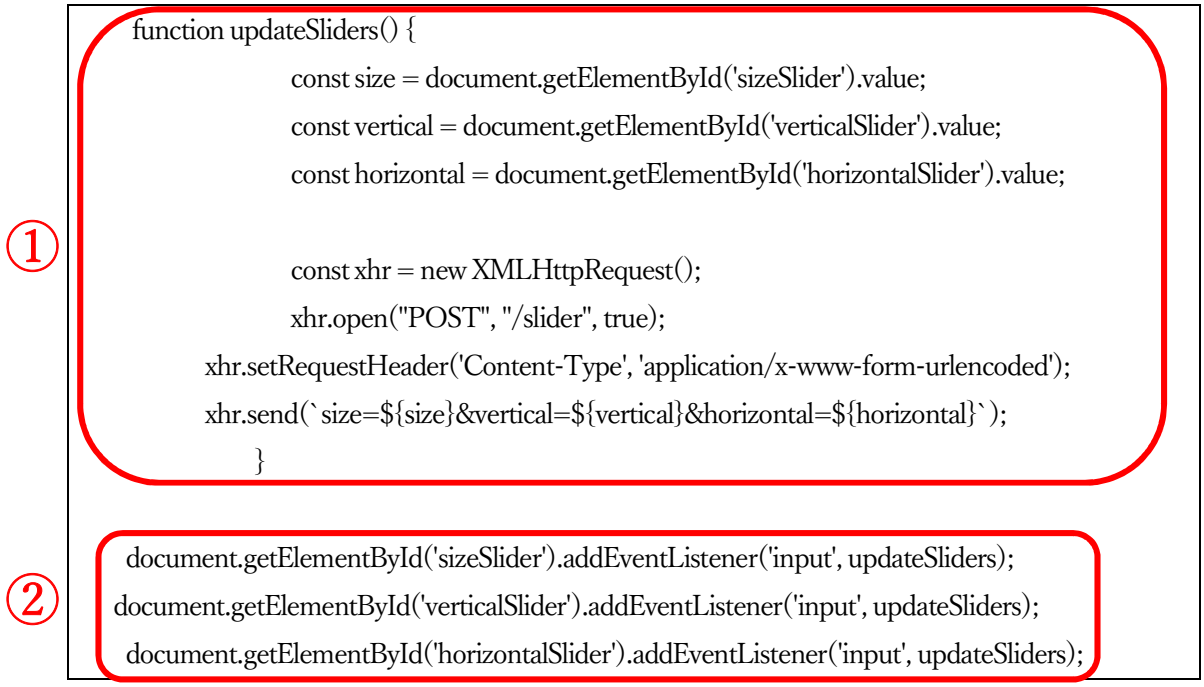
スライダーの値をブラウザが取得してサーバーに送信する。サーバーは画像の調整処理後、ブラウザに値を返して反映させる。



画像 4.8　画像の位置や大きさを調整するスライダー

【プログラム解説】

```
<form id="slider-form">
                    <label>ズーム調整:</label>
                     <input type="range" id="sizeSlider" name="zoom" min="0" max="100" value="50" class="slider">
                    <label>上下調整:</label>
                          <input type="range" id="verticalSlider" name="vertical" min="0" max="100" value="50" class="slider">
                    <label>左右調整:</label>
                      <input type="range" id="horizontalSlider" name="horizontal" min="0" max="100" value="50" class="slider">
                </form>
```



```
function updateSliders() {
        const size = document.getElementById('sizeSlider').value;
        const vertical = document.getElementById('verticalSlider').value;
        const horizontal = document.getElementById('horizontalSlider').value;

        const xhr = new XMLHttpRequest();
        xhr.open("POST", "/slider", true);
    xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
    xhr.send(`size=${size}&vertical=${vertical}&horizontal=${horizontal}`);
    }

document.getElementById('sizeSlider').addEventListener('input', updateSliders);
document.getElementById('verticalSlider').addEventListener('input', updateSliders);
document.getElementById('horizontalSlider').addEventListener('input', updateSliders);
```

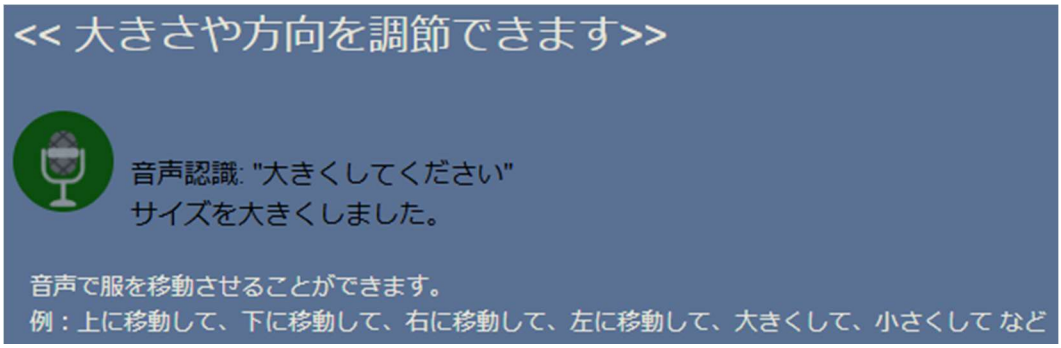

page1.html、page2.html

① updateSliders でスライダーの値を取得し、XMLHttpRequest で Flask サーバーに送信する。Flask サーバーは値をもとに画像の大きさや位置を調整する。

② スライダーを動かすたびに updateSliders()が実行されて、sizeSlider (大きさ)、verticalSlider (上下)、horizontalSlider (左右) の値が更新される。

■ **音声入力による大きさと位置の調整**

- マイクの画像をクリックすると、10 秒間音声認識が可能になり、「大きく、小さく、上、下、右、左」と話すとテキストとして認識する。
- 認識したテキストに応じて値の変更を行い、サーバーに送信後カメラ映像に反映される。



画像 4.9　音声入力による画像調整

【プログラム解説】

①

```
Const   recognition=new(window.SpeechRecognition||window.webkitSpeechRecognition)();
                recognition.lang = 'ja-JP'; //  日本語を指定
                recognition.continuous = true; //  継続的に音声を取得
                recognition.interimResults = false; //  中間結果を表示しない

                //  音声入力開始ボタンの処理
        document.getElementById('startVoiceButton').addEventListener('click', function () {
                    recognition.start(); //  音声入力開始
                    console.log("音声入力開始");
```

②

```
            // 「おおきく」「小さく」などのキーワードを認識して、スライダーの値を調整
            if (transcript.includes('大きく')) {
                currentSizeValue = Math.min(currentSizeValue + 10, 100); // サイズを 10 増加
                actionMessage += 'サイズを大きくしました。<br>';
            } else if (transcript.includes('小さく')) {
                currentSizeValue = Math.max(currentSizeValue - 10, 0); // サイズを 10 減少
                actionMessage += 'サイズを小さくしました。<br>';
            }
```

③

```
            // 「上」「下」「右」「左」の指示で縦横スライダーを変更
            if (transcript.includes('上')) {
                currentVerticalValue = Math.max(currentVerticalValue - 10, 0); // 縦位置を 10 減少
                actionMessage += '上に移動しました。<br>';
            } else if (transcript.includes('した')) {
                currentVerticalValue = Math.min(currentVerticalValue + 10, 100);
// 縦位置を 10 増加
                actionMessage += '下に移動しました。<br>';
            } else if (transcript.includes('右')) {
                currentHorizontalValue = Math.min(currentHorizontalValue + 10, 50);
// 横位置を 10 増加
                actionMessage += '右側に移動しました。<br>';
            } else if (transcript.includes('左')) {
                currentHorizontalValue = Math.max(currentHorizontalValue - 10, -50); // 横位置を 10 減少
                actionMessage += '左側に移動しました。<br>';
            }
```

④

⑤

```
            // スライダーの値を更新
            sizeSlider.value = currentSizeValue;
            verticalSlider.value = currentVerticalValue;
            horizontalSlider.value = currentHorizontalValue;
```

```
                // サーバーに新しい値を送信
                updateSliders();

                // 音声認識結果を表示
                messageDisplay.innerHTML = actionMessage; // 結果表示エリアに更新
            }
```

page1.html、page2.html

① 音声入力はブラウザの Web Speech API を使用して行っている。

② 大きくは＋１０、小さくは－１０とサイズの値を変更する。

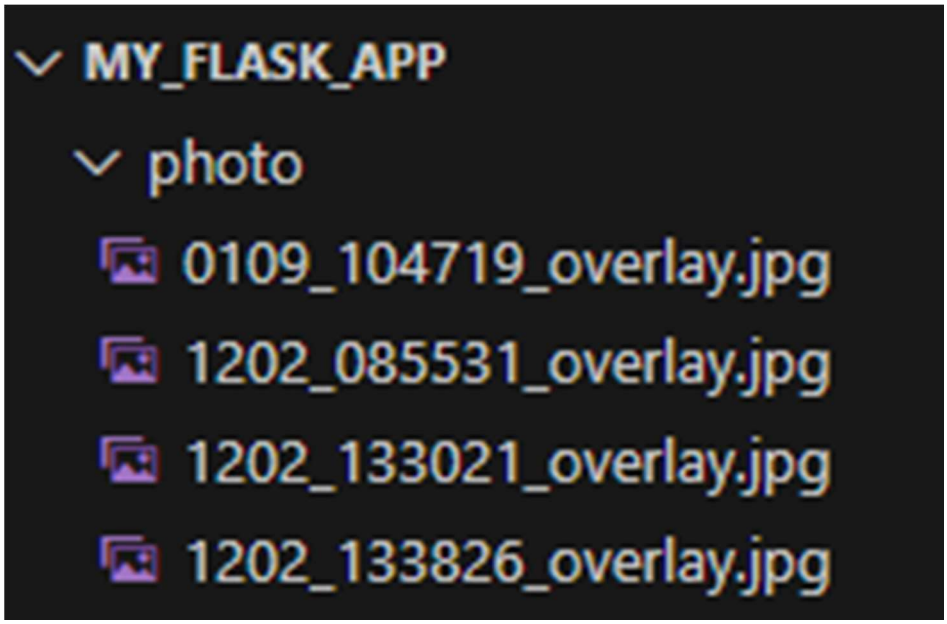
③ 上は－１０、下は＋１０と上下の値を変更する。

④ 右は＋１０、左は－１０と左右の値を変更する。

⑤ スライダーの値を更新する。

■ **動画の撮影・保存**

写真は**日付と時間**を含むファイル名で photo ファイルに保存される。



画像 4.10　photo ファイル

【プログラム解説】

```
filename = f"{datetime.datetime.now().strftime('%m%d_%H%M%S')}_overlay.jpg"
        file_path = os.path.join(PHOTO_DIR, filename)
        cv2.imwrite(file_path, flipped_frame)
```

app.py

ファイル名は"{月日_時分秒}_overlay.jpg"の形式で保存される。

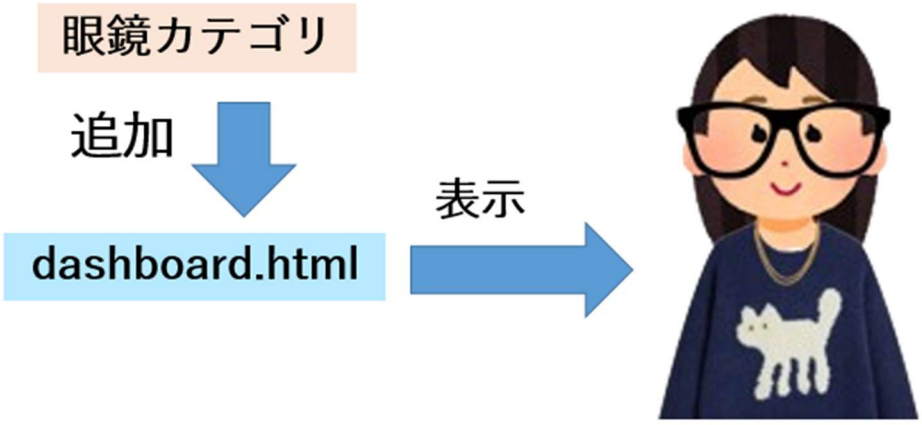
第 6 章　未実装の機能

■ 服とアクセサリーを同時に重ねる

画像 5. 1 服とアクセサリーを同時に重ねる

服とアクセサリーを同時に重ねるために表示ページを一つに統合しようとしたが、それぞれの重ね合わせ処理が異なるため、同じページ内での実装が難しく、最終的に実現できなかった。

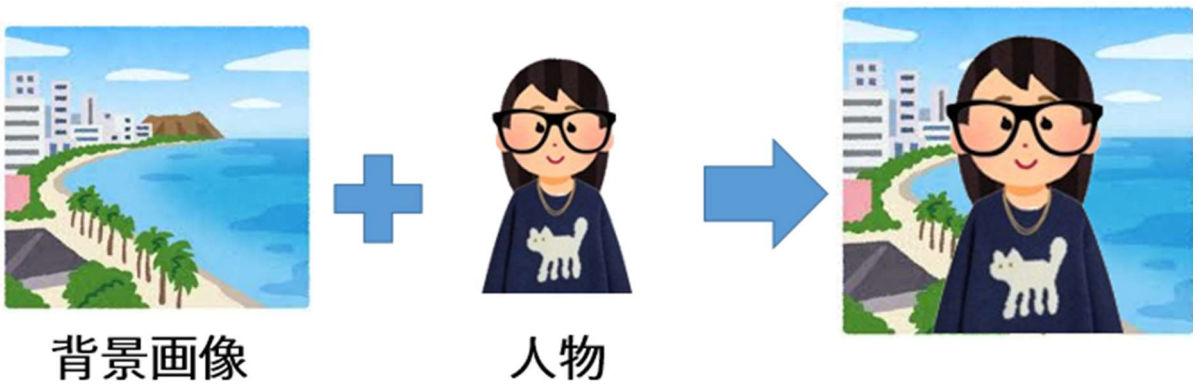
■ 眼鏡カテゴリの追加



画像 5.2　眼鏡カテゴリの追加

服とアクセサリーに加えて眼鏡を重ねる機能を実装予定であったが、服とアクセサリーとの処理の違いにより眼鏡を実装する作業が難航し、別の作業に切り替えたため、実装することができなかった。

■ **背景画像の挿入**



画像 5.3　背景画像の挿入

　カメラ映像の背景ではなく、好きな背景画像を挿入する機能を好きな場面や情景でコーディネートをイメージすることができるようにする予定であったが、別の作業で時間を要しため、実装の作業に取り組むことができなかった。

第 7 章　おわり

開発を進める中で、二つの大きな課題に直面した。一つ目は、JavaScript の配列を使って画像を管理していたが、新しい画像を追加できず、柔軟なデータ管理が難しいという問題である。二つ目は、フロントエンドとバックエンドの連携がうまくいかず、データのやり取りや処理の統合に問題が生じていたことである。そこで、インターネットで関連する技術や解決策を調べ、試行錯誤を繰り返しながら改善を試みた。その結果、一つ目の画像管理の問題については、JSON ファイルを活用することで、ユーザーが試着したい服やアクセサリーを自由に追加できる仕組みを構築し、より柔軟なデータ管理が可能になった。二つ目のフロントエンドとバックエンドの統合については、Flask を導入することでスムーズなデータのやり取りを実現し、システム全体の連携を向上させた。

これらの課題は特に時間を要し、当初の作業計画通りには進められなかったものの、メンバーや指導教員と計画を見直しながら調整を重ねた結果、最終的に現在の「FukuMeet」を開発し、技術的な課題を解決しながらより実用的なシステムへと改善を進めることができた。

開発は個人ではなく、メンバーや指導教員と協力しながら進め、意見を

共有することで、より良いアイデアを生み出し、課題に対する理解を深めることができた。開発中に発生した課題に対しては、インターネットで関連する技術や解決策を調べ、試行錯誤を繰り返すことで、問題解決力を高めることができた。

第 8 章　付録

■ 参考文献

・ [1]HTML 要素リファレンス

https://developer.mozilla.org/ja/docs/Web/HTML/Element

・ [2]CSS 要素リファレンス

https://developer.mozilla.org/ja/docs/Web/CSS/Reference

・ [3]JavaScript リファレンス

https://developer.mozilla.org/ja/docs/Web/JavaScript/Reference

・ [4]python ドキュメント

https://docs.python.org/ja/3/

・ [5]OpenCV Python Tutorials

https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html

・ [6]cv2.VideoCapture で動画ファイル・Web カメラ映像の読み込みと再生

https://python.joho.info/opencv/opencv-videocapture-mp4-movie-py/#toc1

・ [7]Python、OpenCV で画像ファイルの読み込み、保存

https://note.nkmk.me/python-opencv-imread-imwrite/

・ [8]MediaPipe ソリューション ガイド

https://ai.google.dev/edge/mediapipe/solutions/guide?hl=ja

・ [9]姿勢ランドマーク、検出ガイド

https://medium.com/@tayyabjavedbrw789/pose-detection-using-mediapipe-solutions-dabmove-detection-4c4e39080142

・ [10]Flask Tutorials

https://realpython.com/tutorials/flask/

・ [11] Flask で画像とファイルを効率的にアップロードする方法を徹底解説

https://ittrip.xyz/python/flask-file-image-upload#index_id11

・ [12]Flask で画像ファイルをアップロード

https://qiita.com/keimoriyama/items/7c935c91e95d857714fb

・ [13]Python でいろいろ POST して Flask で受け取る

https://qiita.com/tamyu/items/54db82b1c30a3966d8a1

・ [14]背景除去する Background-remove

https://qiita.com/kotai2003/items/2cddf1b3e17c728439b0

・ [15] ウェブ音声 API の使用-Web API | MDN

https://developer.mozilla.org/ja/docs/Web/API/Web_Speech_API/Using_the_Web_Speech_API

・ [16] Web ページでブラウザの音声認識機能を使おう

https://qiita.com/hmmrjn/items/4b77a86030ed0071f548?utm_source=chatgpt.com

・ [17] WebSpeechAPI を利用して Web ブラウザで音声認識を行う

https://zenn.dev/micronn/articles/b654ceca1bdf13