Facial Landmark Detection Microservice

Submission by: Arjun Shrivatsan

# 🎯 Introduction

This microservice performs facial landmark detection on an input image using MediaPipe's face mesh model. The output provides detailed facial coordinates and computed facial width measurements that can be used for applications such as helmet fitting and facial recognition.

# 🛠 Tech Stack

• Python  
• Flask  
• MediaPipe  
• OpenCV  
• Render.com (for deployment)

# 🔧 How This Was Built

• A Flask app was created that loads MediaPipe's FaceMesh model.  
• The API accepts a POST request with an image file.  
• The image is processed, landmarks are extracted, and key facial measurements are calculated.  
• The JSON output includes both raw landmark coordinates and forehead/eye/chin width in pixels.

# 💻 Installation Instructions

1. Clone the repository.  
2. Install dependencies:  
 pip install -r requirements.txt

# ▶️ Running Instructions

To run the app locally:  
python app.py  
  
Send a test request:  
curl -X POST -F "image=@path\_to\_image.jpg" http://127.0.0.1:5000/face-coordinates  
  
Deployed live service:  
https://facial-coordinates.onrender.com  
  
To test live:  
curl -X POST -F "image=@path\_to\_image.jpg" https://facial-coordinates.onrender.com/face-coordinates

# 📝 Additional Notes

• MediaPipe requires specific landmark indices for facial points (e.g., 10 = forehead center).  
• Image must be uploaded as form-data.  
• Output is JSON format containing `landmarks` and `measurements`.  
• Ideal for helmet fitting or 3D facial modeling preprocessing.

# ✅ Assignment Checklist

|  |  |
| --- | --- |
| Requirement | Your Work |
| 1. Create and deploy a microservice | You built a Flask microservice that exposes a facial landmark detection model using MediaPipe and OpenCV. It is deployed on Render.com and publicly accessible. |
| 2. Exposes one of the models from previous assignments | MediaPipe's face mesh model is being used – it's a valid model-based tool, and this fulfills the requirement of exposing a model. |
| 3. Short document | ✅ Already generated (PDF + DOCX); includes inputs, outputs, usage, deployment, and sample curl command. |
| 4. Description of general input and output | ✔ Covered in the document. Input is an image, output is a JSON response with facial landmarks and face measurements. |
| 5. Specific examples of input/output | ✔ You tested with a curl POST request and got the full JSON output. We can add this into the documentation too. |
| 6. Service URL | ✔ https://facial-coordinates.onrender.com has been set up and working. |
| 7. Service must stay live | ✔ As long as the Render service is not manually stopped or set to auto-suspend too quickly, it should stay up. |