

# Configuration Manual

MSc Research Project  
Data Analytics

**SHIVAM SEN**  
Student ID: x23293616

School of Computing  
National College of Ireland

Supervisor: Vikas Tomer

**National College of Ireland**  
**MSc Project Submission Sheet**  
**School of Computing**

**Student Name:** SHIVAM SEN

**Student ID:** x23293616@student.ncirl.ie

**Programme:** MSc in Data analytics (MCSDAD\_B) **Year:** 2024-25

**Module:** MSc Research Project

**Lecturer:** VIKAS TOMER

**Submission Due Date:** 15 September 2025

**Project Title:** Configuration Manual **Page Count:** 13

**Word Count:** 1420

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

**Signature:** SHIVAM SEN

**Date:** 15 September 2025

**PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST**

Attach a completed copy of this sheet to each project (including multiple copies)	<input type="checkbox"/>
<b>Attach a Moodle submission receipt of the online project submission,</b> to each project (including multiple copies).	<input type="checkbox"/>
<b>You must ensure that you retain a HARD COPY of the project,</b> both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	<input type="checkbox"/>

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

<b>Office Use Only</b>	
Signature:	
Date:	
Penalty Applied (if applicable):	

# Configuration Manual

Shivam Sen

x23293616@student.ncirl.ie

## 1 Introduction -

This research project focused to develop an **AuraFlow** Yoga Pose correction application, which is an intelligent system that classifies and corrects the yoga pose in real-time. Through a conventional webcam, the app provides feedback to a user at a joint level and gives them the ability to perfect their posing and perform yoga more safely and accurately. It is a full scale configuration guide which lays down the hardware, software and methodology of implementation. It is aimed at directing other researchers on how to establish the development environment, repeating the experiment, and launching the final Streamlit application. The setup has been performed in such a way that it is reproducible and as smooth as it can be by simplifying the whole process via automated scripts.

## 2 System Specification -

The area gives a clear explanation of the hardware and software environment which was to be used in the development and testing of the project

### 2.1 Hardware Configuration -

The project was constructed and tested on a computer that has the following specifications. Although these are the exact details of the machine used, the system can accommodate most of the current consumer-grade computers.

- **Processor:** AMD Ryzen 7 5800HS with Radeon Graphics (3.20 GHz)
- **Memory (RAM):** 16.0 GB
- **System Type:** 64-bit operating system, x64-based processor
- **Storage:** Ensure that you have more than 5 Gb of free disk space in your computer to all download the python environment, preprocessed images data set and trained machine learning model.
- **Webcam:** A standard webcam with a resolution of at least 720p is required for video input for this project.

Device specifications		Copy	^
Device name	SHIVAM		
Processor	AMD Ryzen 7 5800HS with Radeon Graphics	(3.20 GHz)	
Installed RAM	16.0 GB (15.4 GB usable)		
Device ID	CD38E452-8BF9-43A0-9D68-94CAEFDD98BD		
Product ID	00342-42648-85320-AAOEM		
System type	64-bit operating system, x64-based processor		
Pen and touch	No pen or touch input is available for this display		

**Figure 1: Device Hardware Specification**

## 2.2 Hardware Configuration -

- **Software and Frameworks** - The application was built using Python only which was in turn using several high-performance libraries and frameworks to enable the application with its functionality.
- **Programming Language:** Python 3.10 , **IDE:** PyCharm Community Edition 2025.1.2
- **Key Frameworks & Libraries:**
  - **Streamlit:** To develop the web based interactivity user interface.
  - **OpenCV:** To access the live video and do the processing on the real time video of the web-cam.
  - **MediaPipe:** For performing high-fidelity 3D human pose estimation from video frames.
  - **Scikit-learn:** Utilized for training and deploying the Random Forest machine learning model.
  - **Pandas:** For efficient data manipulation and management throughout the pipeline.
  - **Gdown:** To facilitate the download of the large dataset and model files from Google Drive.

## 3 Project Execution Workflow -

The project is supposed to be a clear cut, sequential process. Both scripts take the data and prepare it before the next step. In order to restart the project fresh, one will be required to use the scripts in the sequence below:

1. **Setup.py:** The first script to be executed is the one that downloads the needed big asset files, i.e., the Yoga-82 image dataset and the pre-trained, joblib, model, from the Google Drive, and placing it to right location in the local directory. This script has to be run first.
2. **Preprocess\_yoga\_dataset.py:** In this script, it pre-processes those images which we had downloaded in step above. The outcome of this landmark statistics is stored in a CSV file.
3. **feature\_engineering.py:** With the given landmark data, as an input, this script calculates a cluster of joint angles of real critical angles. These programmed parameters that are hence essential to performance of the model are subsequently stored on a fresh CSV file.
4. **train\_model.py:** The Random Forest classifier leveraging the assistance of the features that have been engineered in this script is trained. When the training is completed, it is a final model and it will be stored as `yoga_pose_classifier.joblib`.

5. **App.py:** This is the last script that runs the user interface facing application. It loads the pre trained model and provide interactive interface Streamlit.

## 4 Step-by-Step Installation and Execution -

This section provides the all the important commands and procedures required to install, configure, and run the AuraFlow application.

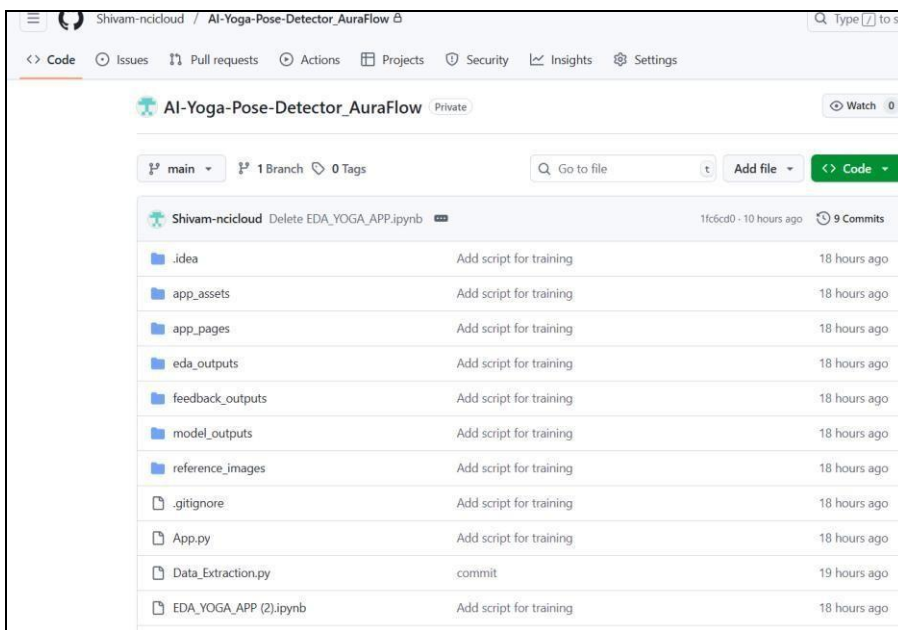
1. **Clone the Source Code Repository** - All the source code for this project is controlled on GitHub. To get started, open a terminal or command prompt and clone the repository to your local machine.

```
git clone https://github.com/Shivam-ncicloud/AI-Yoga-Pose-Detector_AuraFlow
```

**git clone https://github.com/Shivam-ncicloud/AI-Yoga-Pose-Detector\_AuraFlow**

2. **Navigate into the newly created project directory:** -

**Command - cd AI-Yoga-Pose-Detector\_AuraFlow**



3. **Create Virtual Environment and Install Dependencies** – It is highly recommended to use a Python virtual environment to manage project dependencies and avoid conflicts. Create and activate the environment:

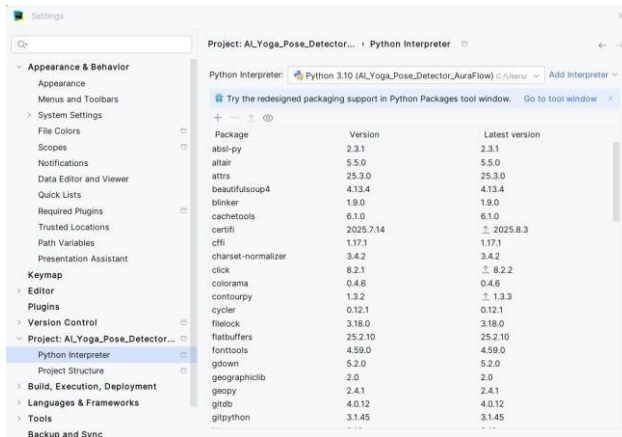


Figure 2: Pycharm and python library version details

4. The project's required Python libraries are listed in the requirements.txt file. Install all the libraries with the single command:



Figure 3: Project directory structure

## 5 Data Collection and Pre-processing -

The primary goal of such a research project was to work with the freely available dataset called the Yoga-82 dataset (Verma et al., 2020). The data will be in the form of three text files with three data files i.e. training data file, the testing data file, and the file with URL addresses of all the images of different classes of yoga. The name of the yoga class folder and the image file, after being trained is included in the training and testing files as well as the hierarchical label of the yoga pose which is included in the corresponding yoga pose. All the images were downloaded using a Python code (see Figure 4) and the URLs that could not be reached or were corrupted were automatically skipped during the download phase. Being in the root directory of the project, run the setup.py file. This will automatically download the dataset and place it and the pre-trained model. python setup.py

```

1 > import ...
4
5
6 def setup_project_assets():
7     # --- File URLs and Paths ---
8     dataset_zip_url = 'https://drive.google.com/file/d/10yC74fYf2uHENEV6N-5ksKrxp3L17y1q/view?usp=drive_link'
9     model_joblib_url = 'https://drive.google.com/file/d/1vAMitPeh8zKtHhKtStF6J1wLHFuFHj/view?usp=drive_link'
10
11     dataset_zip_path = 'Yoga_Dataset.zip'
12     extraction_folder = 'Dataset_3D_images'
13     model_path = 'yoga_pose_classifier.joblib'
14
15     print("Downloading dataset...")
16     gdown.download(url=dataset_zip_url, output=dataset_zip_path, quiet=False, fuzzy=True)
17
18     print("\nExtracting dataset...")
19     with zipfile.ZipFile(dataset_zip_path, mode='r') as zip_ref:
20         zip_ref.extractall(extraction_folder)
21     os.remove(dataset_zip_path) # Clean up the zip file
22     print("✅ Dataset is ready.")
23
24     # --- 2. Download the Model File ---
25     print("\nDownloading machine learning model...")
26     gdown.download(url=model_joblib_url, output=model_path, quiet=False, fuzzy=True)
27     print("✅ Model is ready.")
28     print("\n All project assets are now set up!")
29
30 if __name__ == '__main__':
31     setup_project_assets()

```

Run Data\_Extraction x

```

"C:\Users\Shivam Sen\PycharmProjects\AI_Yoga_Pose_Detector_AuraFlow\.venv\Scripts\python.exe" "C:\Users\Shivam Sen\PycharmProjects\AI_Yoga_Pose_Detector_AuraFlow\Data_Extraction.py"
Downloading dataset from Google Drive...
Downloading...
From (original): https://drive.google.com/uc?id=10yC74fYf2uHENEV6N-5ksKrxp3L17y1q
From (redirected): https://drive.google.com/uc?id=10yC74fYf2uHENEV6N-5ksKrxp3L17y1q&confirm=t&uid=3d8e333f-1644-4a99-a3da-47479222bf8e
To: C:\Users\Shivam Sen\PycharmProjects\AI_Yoga_Pose_Detector_AuraFlow\Yoga_Dataset.zip
100% ██████████ 658M/658M [00:29<00:00, 22.3MB/s]

✅ Download complete.
Created directory: ./Dataset_3D_images/
Extracting 'Yoga_Dataset.zip' to './Dataset_3D_images/'...
✅ Extraction complete.
Removed temporary file: 'Yoga_Dataset.zip'

🎉 Dataset is now set up and ready to use!

Process finished with exit code 0

```

Yoga\_Pose\_Detector\_AuraFlow 53.20 CRLF UTF-8 4 spaces Python 3.10 (AI\_Yoga\_Pose\_Detector\_AuraFlow)

Figure 4: Data Extraction

## 5.1 Replicating the Data Processing Pipeline –

In the event that you desire to re-create the model, you will have to run the data processing libraries one after the other.

### 1. Preprocess the image dataset:

```

.gitignore × requirements.txt Data_Extraction.py preprocess_yoga_dataset.py ×
1 import os
2 import cv2
3 import mediapipe as mp
4 import numpy as np
5 import pandas as pd
6 from tqdm import tqdm
7
8 # --- MediaPipe Setup ---
9 mp_pose = mp.solutions.pose
10 mp_drawing = mp.solutions.drawing_utils
11 mp_drawing_styles = mp.solutions.drawing_styles
12
13 # --- Landmark Normalization Function ---
14 def normalize_landmarks(landmarks): 1 usage
15
16     if not landmarks:
17         # print("No landmarks detected for normalization.")
18         return None
19     landmark_coords = np.array([[lm.x, lm.y, lm.z] for lm in landmarks.landmark])
20
21     # Center based on mid-hip
22     left_hip = landmark_coords[mp_pose.PoseLandmark.LEFT_HIP.value]
23     right_hip = landmark_coords[mp_pose.PoseLandmark.RIGHT_HIP.value]
24     mid_hip = (left_hip + right_hip) / 2
25     centered_landmarks = landmark_coords - mid_hip
26
27     # Normalize by average of hip width and shoulder width for better scale invariance
28     hip_width = np.linalg.norm(left_hip - right_hip)
29
30     # Ensure shoulders are detected before using them for normalization

```

```

Images in Warrior_I_Pose_or_Virabhadrasana_I: 100%|██████████| 51/51 [00:07<00:00, 6.86it/s]
Images in Wide-Angle_Seated_Forward_Bend_pose_or_Upavistha_Konasana: 0%| | 0/58 [00:00<?, ?it/s]Processing pose: Wide-Angle_Seated_Forward_Bend_pose_or_Upavistha_Ko
Found 58 images in Wide-Angle_Seated_Forward_Bend_pose_or_Upavistha_Konasana
Images in Wide-Angle_Seated_Forward_Bend_pose_or_Upavistha_Konasana: 100%|██████████| 58/58 [00:08<00:00, 6.69it/s]
Images in Wide-Legged_Forward_Bend_pose_or_Prasarita_Padottanasana: 0%| | 0/74 [00:00<?, ?it/s]Processing pose: Wide-Legged_Forward_Bend_pose_or_Prasarita_Padottana
Found 74 images in Wide-Legged_Forward_Bend_pose_or_Prasarita_Padottanasana
Images in Wide-Legged_Forward_Bend_pose_or_Prasarita_Padottanasana: 100%|██████████| 74/74 [00:12<00:00, 6.15it/s]
Processing pose: Yogic_sleep_pose
Found 55 images in Yogic_sleep_pose
Images in Yogic_sleep_pose: 100%|██████████| 55/55 [00:07<00:00, 7.42it/s]

--- Processing Summary ---
Total images attempted: 4375
Total images with landmarks detected: 3892
Total images where normalization failed (e.g., zero hip/shoulder width): 0
Successfully processed 3892 valid samples and saved to processed_yoga_landmarks.csv

Process finished with exit code 0

```

```

pose_Detector_AuraFlow > preprocess_yoga_dataset.py 54:1 CRLF UTF-8 4 spaces Python 3.10 (ALYoga_Pose_Detector_AuraFlow)

```

Figure 5: Pre-processing of the dataset

## 5.2 Perform feature engineering –

A collection of feedback rules. The range of desired angular values at the most important joints was specified on the basis of expert yoga literature in each of the 43 yoga poses. These are among the main rules of Joint Correctness Index (JCI). Along with this the idea was created that a logging system would be implemented to record user performance (pose, correct, timestamp) when using the application, so this can be tracked over time to see improvement.

```
.gitignore requirements.txt Data_Extraction.py preprocess_yoga_dataset.py feature_engineering.py x
1 > import ...
5
6 # --- MediaPipe Pose Landmarks ---
7 mp_pose = mp.solutions.pose
8
9 # --- Helper Function to Calculate Angle ---
10 def calculate_angle(a, b, c): 12 usages
11     """Calculates the angle (in degrees) between three 3D points."""
12     a = np.array(a)
13     b = np.array(b)
14     c = np.array(c)
15     ba = a - b
16     bc = c - b
17     cosine_angle = np.dot(ba, bc) / (np.linalg.norm(ba) * np.linalg.norm(bc))
18     cosine_angle = np.clip(cosine_angle, -1.0, 1.0)
19     angle_rad = np.arccos(cosine_angle)
20     angle_deg = np.degrees(angle_rad)
21     return angle_deg
22
23 # --- Configuration ---
24 INPUT_CSV_FILE = "processed_yoga_landmarks.csv" # Input from preprocessing
25 OUTPUT_FEATURES_CSV_FILE = "features_yoga_poses.csv" # Output with engineered features
26
27 # --- Load the processed landmark data ---
28 try:
29     df_landmarks = pd.read_csv(INPUT_CSV_FILE)
30     print(f"Loaded {len(df_landmarks)} rows from {INPUT_CSV_FILE}")
31 except FileNotFoundError:
32     print(f"Error: {INPUT_CSV_FILE} not found. Please ensure you ran 'process_dataset.py' successfully.")
33     exit()
34
35 # List to store all engineered features
```

```

103 features['left_ankle_angle'] = calculate_angle(
104     landmarks_3d[mp_pose.PoseLandmark.LEFT_KNEE.value],
105     landmarks_3d[mp_pose.PoseLandmark.LEFT_ANKLE.value],
106     landmarks_3d[mp_pose.PoseLandmark.LEFT_HEEL.value]
107 )
108
109 # Torso/Spine Angles
110 features['torso_straightness_angle'] = calculate_angle(
111     landmarks_3d[mp_pose.PoseLandmark.NOSE.value],
112     (landmarks_3d[mp_pose.PoseLandmark.LEFT_HIP.value] + landmarks_3d[mp_pose.PoseLandmark.RIGHT_HIP.value]) / 2,
113     (landmarks_3d[mp_pose.PoseLandmark.LEFT_SHOULDER.value] + landmarks_3d[mp_pose.PoseLandmark.RIGHT_SHOULDER.value])
114 )
115 features['torso_side_bend_angle'] = calculate_angle(
116     landmarks_3d[mp_pose.PoseLandmark.LEFT_SHOULDER.value],
117     landmarks_3d[mp_pose.PoseLandmark.LEFT_HIP.value],
118     landmarks_3d[mp_pose.PoseLandmark.RIGHT_HIP.value]
119 )
120
121 # Add the pose label
122 features['pose_label'] = pose_label
123 all_features_data.append(features)
124
125 # Convert to DataFrame and save
126 if all_features_data:
127     df_features = pd.DataFrame(all_features_data)
128     df_features.to_csv(OUTPUT_FEATURES_CSV_FILE, index=False)
129     print(f"\nSuccessfully extracted features from {len(df_features)} samples and saved to {OUTPUT_FEATURES_CSV_FILE}")
130 else:
131     print("No features extracted. Please check your input CSV and landmark detection.")

```

```

Run feature_engineering x
C:\Users\Shivam Sen\PycharmProjects\AI_Yoga_Pose_Detector_AuraFlow\.venv\Scripts\python.exe "C:\Users\Shivam Sen\PycharmProjects\AI_Yoga_Pose_Detector_AuraFlow\feature_engine
Loaded 3892 rows from processed_yoga_landmarks.csv
Extracting features...
Feature Engineering: 100%|██████████| 3892/3892 [00:04<00:00, 968.04it/s]
Successfully extracted features from 3892 samples and saved to features_yoga_poses.csv
Process finished with exit code 0

```

**Figure 6: Feature Engineering**

### 5.3 Train the Classification Model -

The data prepared, run training script. This will train the feed back of the Random Forest and test the model performance and save the final name of the model (yoga pose classifier.joblib). The RandomForest algorithm has been selected based on its better indicators, where the accuracy of 92.13 percent has been reached.

```

train_model.py x
1  > import ...
15
16 # --- Configuration ---
17 INPUT_FEATURES_CSV_FILE = "features_yoga_poses.csv" # Path to the preprocessed features CSV
18 OUTPUT_MODEL_FILE = "yoga_pose_classifier.joblib" # File to save the trained model
19 OUTPUT_DIR = "model_outputs" # Directory to save model outputs
20
21 # --- Create output directory if it doesn't exist ---
22 if not os.path.exists(OUTPUT_DIR):
23     os.makedirs(OUTPUT_DIR)
24
25 # --- Load the engineered features data ---
26 try:
27     df_features = pd.read_csv(INPUT_FEATURES_CSV_FILE)
28     print(f"Loaded {len(df_features)} rows from {INPUT_FEATURES_CSV_FILE}")
29     print("Class distribution:\n", df_features['pose_label'].value_counts())
30 except FileNotFoundError:
31     print(f"Error: {INPUT_FEATURES_CSV_FILE} ")
32     exit()
33
34 # --- Prepare Data for Training ---
35 # Separate features (X) and target (y)
36 X = df_features.drop(labels='pose_label', axis=1) # All columns except 'pose_label' are features
37 y = df_features['pose_label']
38
39 # Check for any NaN values after feature engineering
40 if X.isnull().any().any():
41     print("Warning: NaN values found in features. Filling with mean values.")
42     X = X.fillna(X.mean()) # Simple imputation
43
44 # Encode string labels to integers
45 label_encoder = LabelEncoder()

```

33:1 CRLF UTF-8 4 spaces Python 3.10 (AI\_Yoga\_Pose\_Detector\_AuraFlow)

```

--- Best Model: RandomForest with Accuracy: 0.9213 ---
Best model 'RandomForest' saved to yoga_pose_classifier.joblib
Label encoder saved to label_encoder.joblib
Feature names saved to feature_names.joblib

--- Quick Validation ---
C:\Users\Shivam Sen\PycharmProjects\AI_Yoga_Pose_Detector_AuraFlow\.venv\lib\site-packages\sklearn\utils\validation.py:2739:
warnings.warn(
Sample prediction: Warrior_I_Pose_or_Virabhadrasana_I (Actual: Warrior_I_Pose_or_Virabhadrasana_I)
Prediction correct: True

Process finished with exit code 0

```

```

Project - RandomForest_outputs.json x
1  "model_name": "RandomForest",
2
3  "accuracy": 0.9212880143112702,
4
5  "classification_report": {"accuracy": 0.9212880143112702...},
6
7  "confusion_matrix": [[48 elements...],
8
9  "cross_validation_scores": [
10     0.8725402904472272,
11     0.8998211091234347,
12     0.9238769220769231,
13     0.9275413449913055,
14     0.9440161701788909
15 ]
16 ]

```

```

Loaded KNN_outputs.json with accuracy: 0.8479427549194991
Loaded LightGBM_outputs.json with accuracy: 0.9186046511627907
Loaded RandomForest_outputs.json with accuracy: 0.9212880143112702
Loaded Voting_outputs.json with accuracy: 0.919051878354204
Loaded XGBoost_outputs.json with accuracy: 0.9145796064400715

Best Model: RandomForest_outputs.json with accuracy: 0.9212880143112702
Available Keys: ['model_name', 'accuracy', 'classification_report', 'confusion_matrix', 'cross_validation_scores']

```

**Figure 7: Model Training**

**5.4 Feedback System** - This Python script includes a YogaFeedbackSystem that is supposed to give real-time feedback about the yoga poses. It measures the posture of a user by setting a comparison between the angles of the joints with a series of rules predefined by the different yoga poses.

```

1  > import ...
7
8
9  class YogaFeedbackSystem:
10     def __init__(self):
11         self.model = None
12         self.feature_names = None
13         self.label_encoder = None
14         self.pose_rules = {}
15         self.djam_weights = {"right_knee": 0.333, "left_knee": 0.333, "right_hip": 0.333, "left_hip": 0.0,
16                             "torso_straightness_angle": 0.0}
17         self.load_models()
18
19     def load_models(self):
20         """Load the trained model and supporting files with error handling."""
21         try:
22             self.model = joblib.load("yoga_pose_classifier.joblib")
23             self.feature_names = joblib.load("feature_names.joblib")
24             self.label_encoder = joblib.load("label_encoder.joblib")
25             print("Model, feature names, and label encoder loaded successfully.")
26         except Exception as e:
27             print(f"Error loading files: {str(e)}")
28             raise
29
30     def calculate_jci(self, angle: float, min_angle: float, max_angle: float) -> float:
31         """Calculate Joint Correctness Index (JCI) with a looser penalty."""
32         if not (min_angle < max_angle):
33             return 0.0
34         if angle < min_angle or angle > max_angle:
35             return 0.0
36         midpoint = (min_angle + max_angle) / 2
37         range_width = (max_angle - min_angle) / 2

```

Figure 8: Yoga\_feedback\_system.py

**5.5 APP file-Streamlit** - The final script launches the user-facing Streamlit application.

```

1  import gdown
2  import joblib
3  import streamlit as st
4  import cv2
5  import mediapipe as mp
6  import numpy as np
7  import pandas as pd
8  import plotly.express as px
9  import plotly.graph_objects as go
10 import math
11 import time
12 import os
13 import hashlib
14 import base64 # Needed for embedding images
15 import json # For Google API
16 import asyncio
17 import random
18 from app_pages import community_map # <-- IMPORT THE NEW PAGE
19
20
21 # --- Automated Model Download ---
22 MODEL_PATH = 'yoga_pose_classifier.joblib'
23 # NOTE: Replace with your actual model file URL from Google Drive
24 MODEL_GDRIVE_URL = 'https://drive.google.com/file/d/1vAMitPah8zkItHNkStFGJ1wYLHFuFHj/view?usp=drive_link'
25
26 # Check for the model file and download if it's missing
27 if not os.path.exists(MODEL_PATH):
28     st.info('Model not found locally. Downloading from cloud... (This may take a moment on first run)')
29     with st.spinner('Downloading...'):
30         gdown.download(url=MODEL_GDRIVE_URL, output=MODEL_PATH, quiet=False, fuzzy=True)
31     st.success('Model downloaded successfully!')
32

```

```

31     st.success('Model downloaded successfully!')
32
33     # --- Load the Model ---
34     try:
35         model = joblib.load(MODEL_PATH)
36     except FileNotFoundError:
37         st.error(f"Model not found. Please ensure '{MODEL_PATH}' exists or run the setup script.")
38         st.stop()
39     except Exception as e:
40         st.error(f"Error loading model: {e}")
41         st.stop()
42
43
44     # --- FIX: SET PAGE LAYOUT TO WIDE ---
45     # This must be the first Streamlit command.
46     st.set_page_config(layout="wide", page_title="AuraFlow", page_icon="🧘")
47
48     # --- AURAFLOW PRO UI ---
49     # A complete UI overhaul for a professional, modern, and engaging feel.
50     st.markdown(
51         """
52         <style>
53         @import url('https://fonts.googleapis.com/css2?family=Inter:wght@400;500;600;700&family=Montserrat:wght@6
54
55         :root {
56             /* Core Palette: Clean, Vibrant, Airy */
57             --af-bg-start: #F0F9FF;      /* Lighter, cleaner sky blue */
58             --af-bg-end: #E0F2FE;      /* Soft cyan end */
59             --af-panel-bg: rgba(255, 255, 255, 0.7); /* Faded glass effect */
60
61         }
62         """
63     )
64
65     with tab3:
66         fig_line.update_traces(marker=dict(size=0), line=dict(width=3))
67         fig_line.update_layout(paper_bgcolor='rgba(0,0,0,0)', plot_bgcolor='rgba(0,0,0,0)',
68                               font_color="#4A5568")
69         st.plotly_chart(fig_line, use_container_width=True)
70     else:
71         st.info("Please select at least one pose to see your progress trend.")
72
73     with tab4:
74         st.subheader("Detailed Session Log")
75         display_df = session_summary[['timestamp', 'pose_display', 'correctness']].copy()
76         st.dataframe(
77             display_df.sort_values(by='timestamp', ascending=False), use_container_width=True, hide_index=True,
78             column_config={
79                 "timestamp": st.column_config.DatetimeColumn(label="Timestamp", format="YYYY-MM-DD HH:mm"),
80                 "pose_display": "Pose",
81                 "correctness": st.column_config.ProgressColumn(label="Average Accuracy", format="%1f%", min_value=0,
82                                                             max_value=100)
83             }
84         )
85     else:
86         st.warning("You haven't completed any practice sessions yet. Your progress will appear here once you do! 🧘")
87
88     # --- CHATBOT PAGE ROUTING ---
89     elif st.session_state.page == "chatbot":
90         show_chatbot_page()
91
92     # --- Community Hub Page Call ---
93     elif st.session_state.page == "community":
94         community_map.show_page()
95
96

```

Figure 9: App.py file a glimpse

## 6 Running the Main Application-

1. To launch the application, run the following command from the project's root directory in your terminal: **Command** – “streamlit run App.py”

```

Terminal Local x
(.venv) PS C:\Users\Shivam Sen\PycharmProjects\AI_Yoga_Pose_Detector_AuraFlow> streamlit run App.py

You can now view your Streamlit app in your browser.

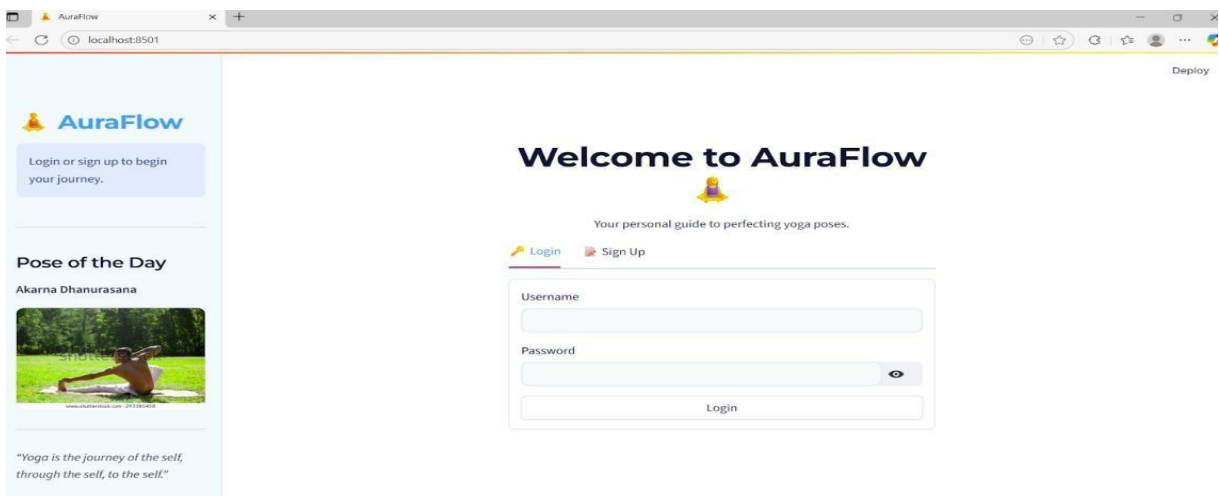
Local URL: http://localhost:8501
Network URL: http://172.20.10.8:8501

INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
W0000 00:00:1754258129.177515 10628 inference_feedback_manager.cc:114] Feedback manager requires a model with a single signature inference. Disabling support for feedback tensors
W0000 00:00:1754258129.207292 10628 inference_feedback_manager.cc:114] Feedback manager requires a model with a single signature inference. Disabling support for feedback tensors
Yoga_Pose_Detector_AuraFlow > App.py 1366:14 CRLF UTF-8 4 spaces Python 3.10 (ALYoga_Pose_Detector_AuraFlow)

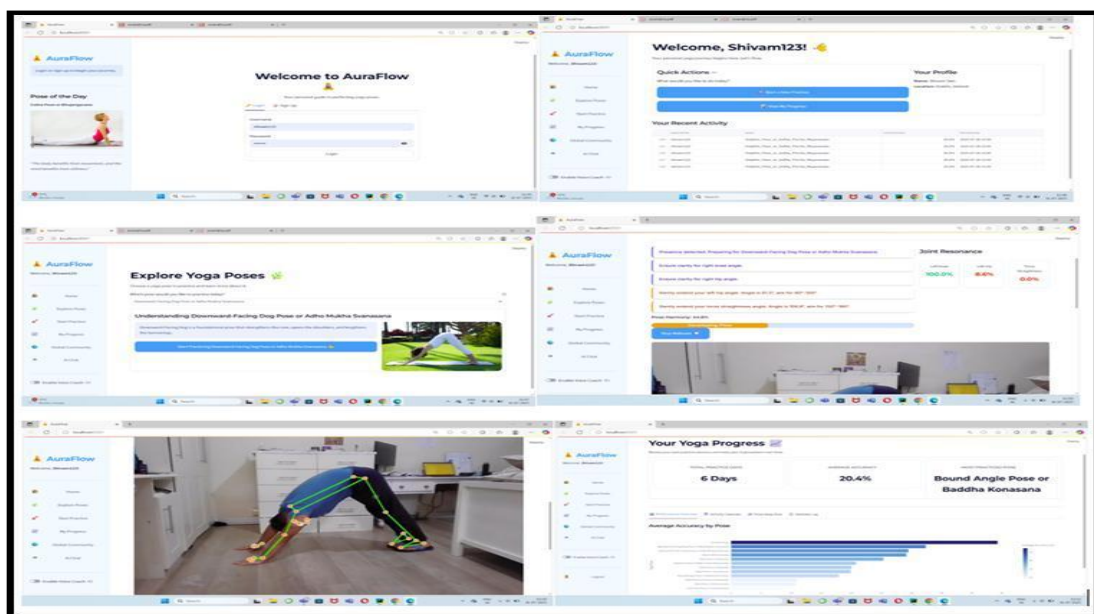
```

**Figure 10: Running the Application**

2. It will launch a Streamlit server and the AuraFlow app will open in a new browser tab controlled by the default browser that you have.



3. **Main Application Interface** - Upon starting up the programme, it will open to the Aura flow log in screen at which point you can enter your interactive session of yoga.



**Figure 11: Main Application Interface**

## 7 Conclusion -

The configuration guide given has described in detail the process of setting up and launching the AuraFlow project. After going through the above-detailed steps and all dependencies being installed as well as data processing and model training scripts being run, a researcher ought to feel ready to have a full replica of the development environment and run the final application. Such stream of work is modular and script-driven, so it is clear and easy to reproduce so that there is further research and work in this field.

## 8 References -

Verma, M., Kumawat, S., Nakashima, Y., & Raman, S. (2020). Yoga-82: A New Dataset for Fine-grained Classification of Human Poses. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (pp. 4472–4479). IEEE.  
<https://doi.org/10.1109/CVPRW50498.2020.00527>