National College of Ireland

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

SubmissionDue Date: 15 September 2025

Project Title: Configuration Manual

Word Count: 1420 **Page Count:** 13

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)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

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

Office Use Only	
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.

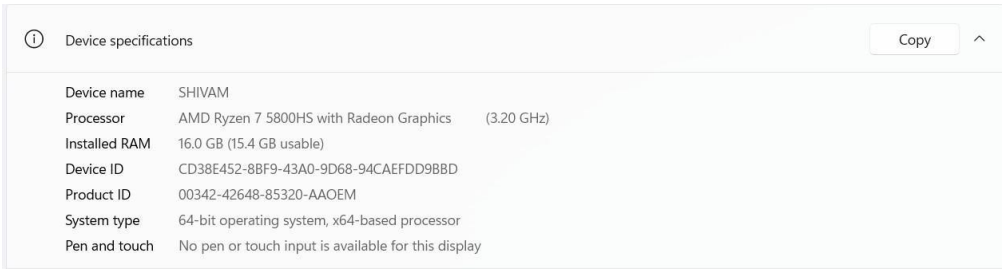


Figure 1: Device Hardware Specification

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.

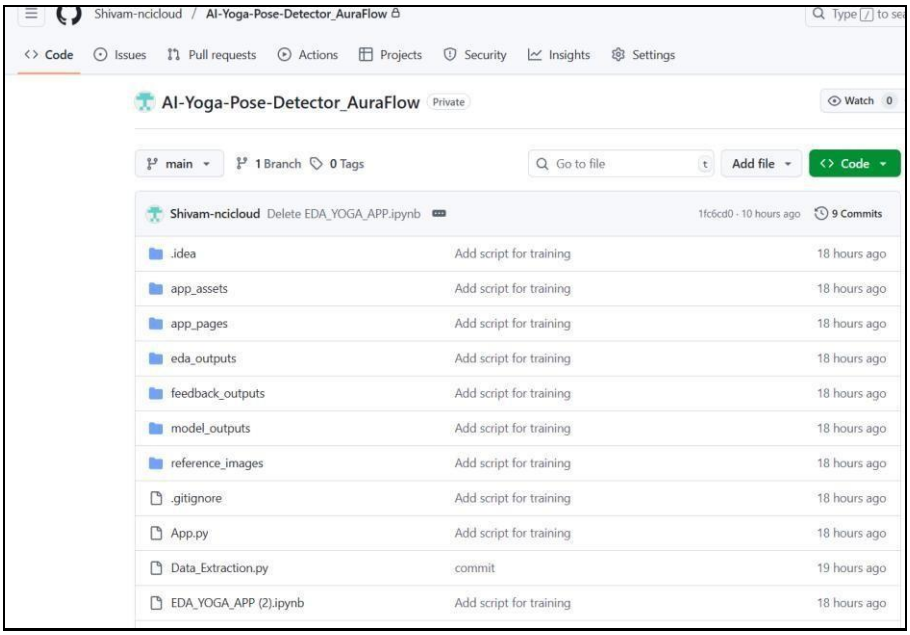
1. **Clone the Source Code Repository -** All the source code for this project is controlled on GitHub. To get started, open a terminal or command prompt and clone the repository to your local machine.

```
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:

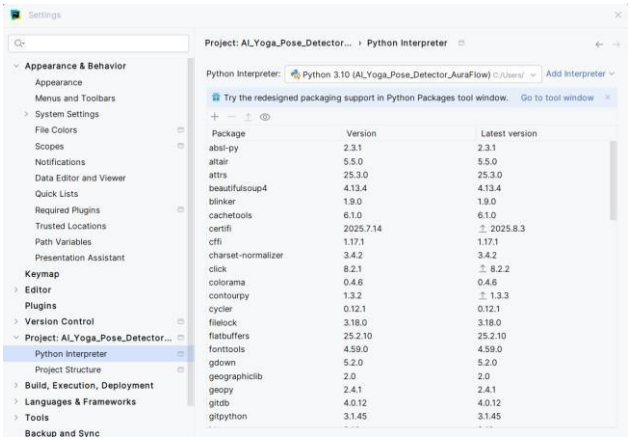


Figure 2: Pycharm and python library version details

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

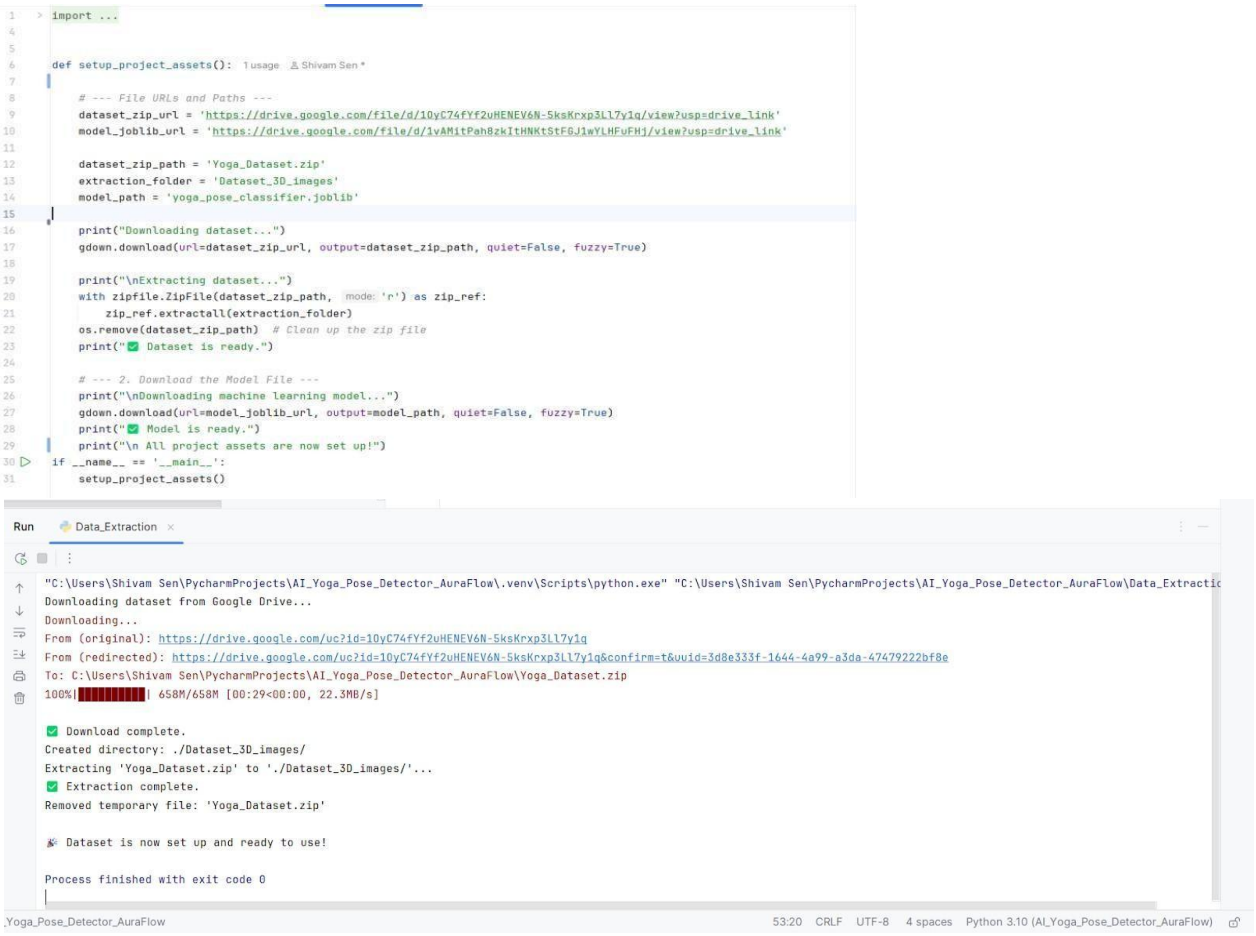


Figure 4: Data Extraction

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 ×

```python
import os
import cv2
import mediapipe as mp
import numpy as np
import pandas as pd
from tqdm import tqdm

# --- MediaPipe Setup ---
mp_pose = mp.solutions.pose
mp_drawing = mp.solutions.drawing_utils
mp_drawing_styles = mp.solutions.drawing_styles

# --- Landmark Normalization Function ---
def normalize_landmarks(landmarks):  1 usage

    if not landmarks:
        # print("No landmarks detected for normalization.")
        return None
    landmark_coords = np.array([[lm.x, lm.y, lm.z] for lm in landmarks.landmark])

    # Center based on mid-hip
    left_hip = landmark_coords[mp_pose.PoseLandmark.LEFT_HIP.value]
    right_hip = landmark_coords[mp_pose.PoseLandmark.RIGHT_HIP.value]
    mid_hip = (left_hip + right_hip) / 2
    centered_landmarks = landmark_coords - mid_hip

    # Normalize by average of hip width and shoulder width for better scale invariance
    hip_width = np.linalg.norm(left_hip - right_hip)

    # Ensure shoulders are detected before using them for normalization
```

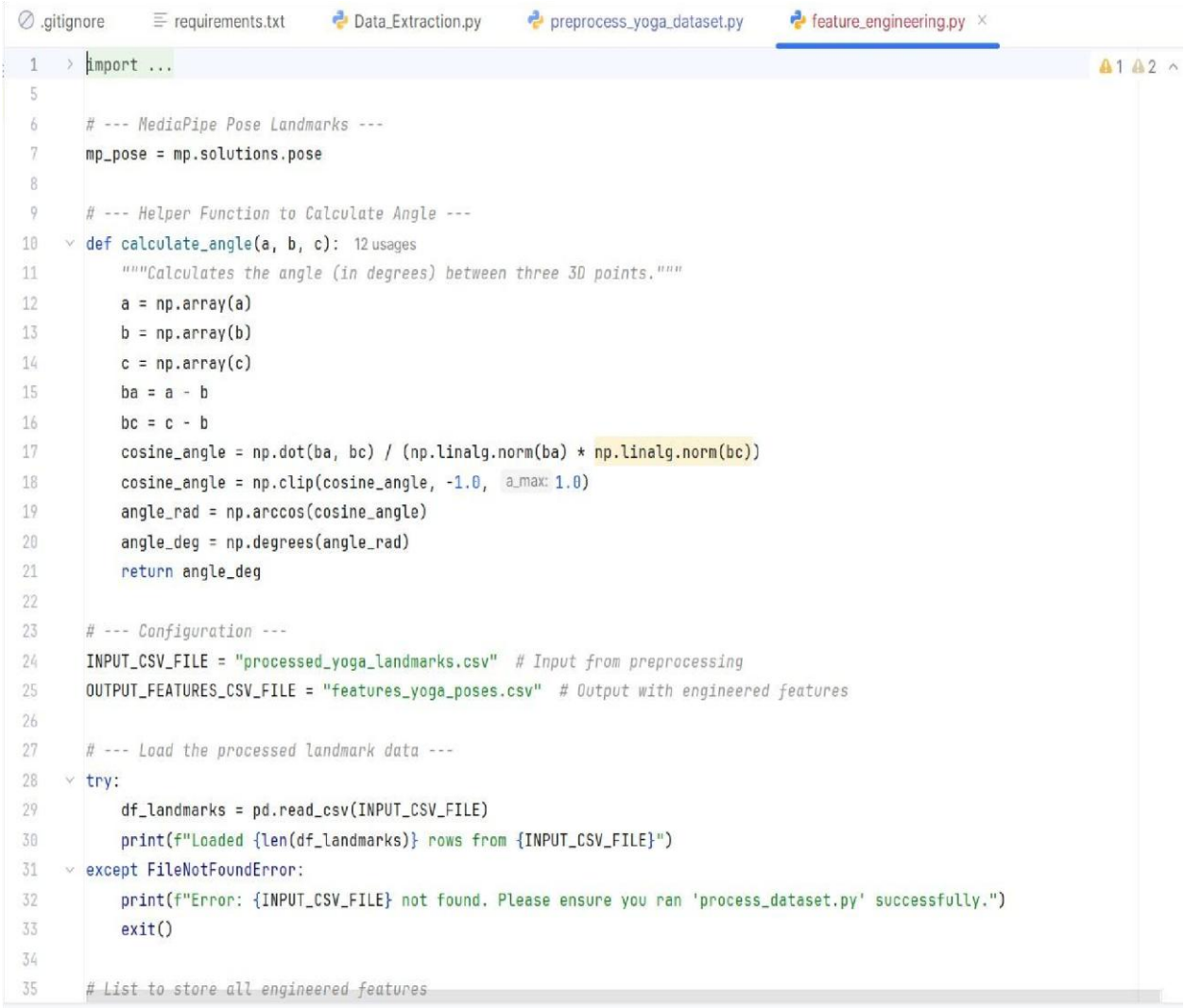
```
Images in Warrior_I_Pose_or_Virabhadrasana_I: 100%|██████████| 51/51 [00:07<00:00,  6.86it/s]
Images in Wide-Angle_Seated_Forward_Bend_pose_or_Upavistha_Konasana:   0%|          | 0/58 [00:00<?, ?it/s]Processing pose: Wide-Angle_Seated_Forward_Bend_pose_or_Upavistha_Kor
Found 58 images in Wide-Angle_Seated_Forward_Bend_pose_or_Upavistha_Konasana
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_Padottanas
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
```

ose_Detector_AuraFlow › preprocess_yoga_dataset.py 54:1 CRLF UTF-8 4 spaces Python 3.10 (AI_Yoga_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.



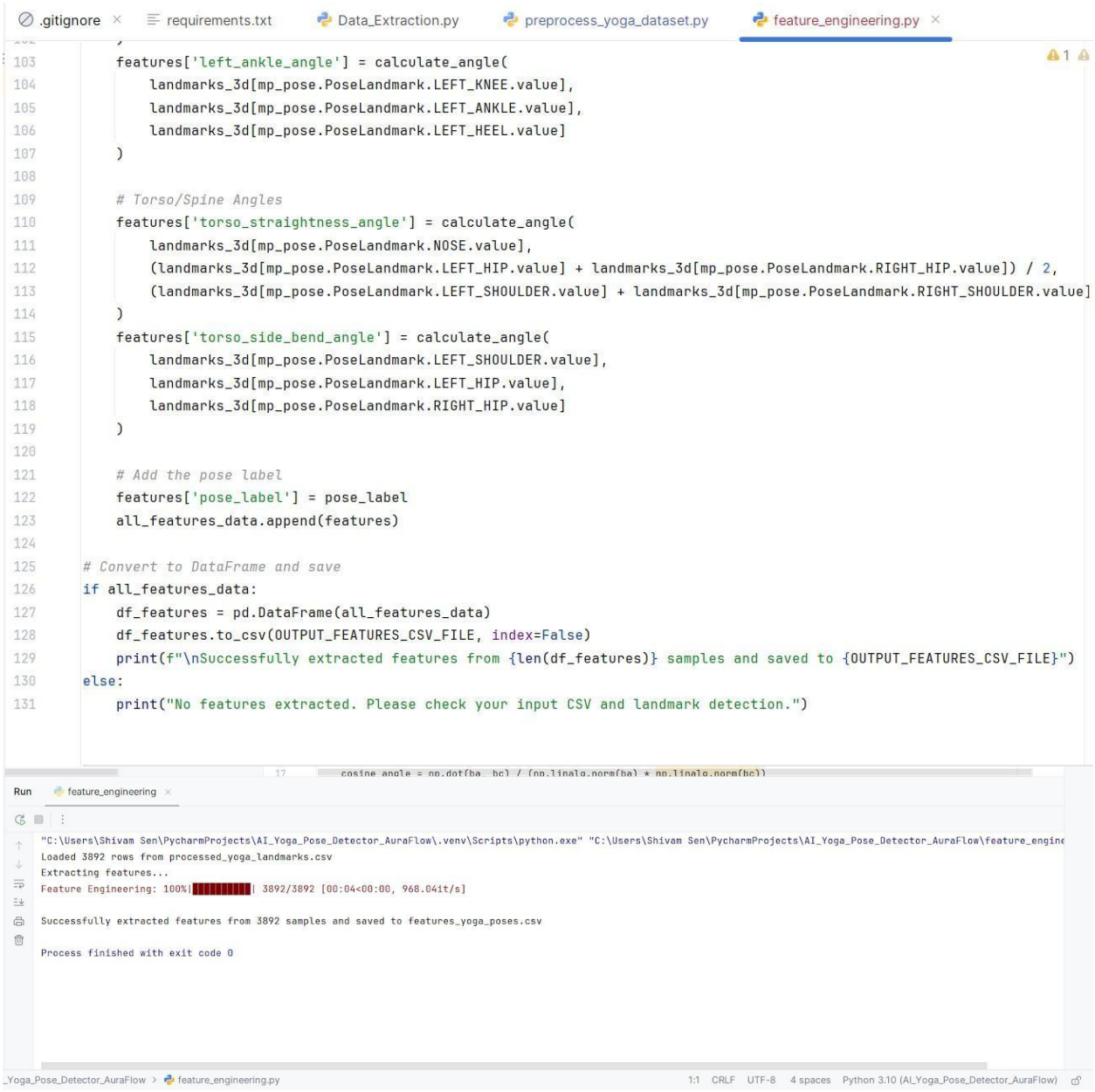


Figure 6: Feature Engineering

5.3 Train the Classification Model -

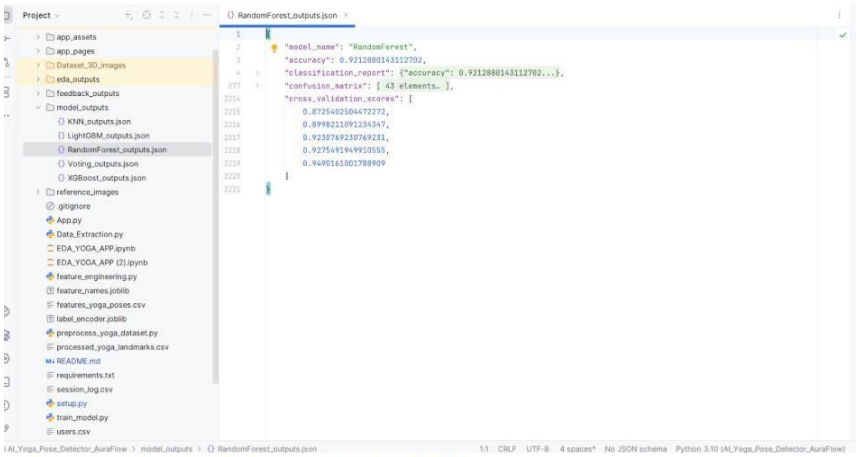
The data prepared, run training script. This will train the feed back of the Random Forest and test the model performance and save the final name of the model (yoga pose classifier.joblib). The RandomForest algorithm has been selected based on its better indicators, where the accuracy of 92.13 percent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

train_model.py

```python
import ...

# --- Configuration ---
INPUT_FEATURES_CSV_FILE = "features_yoga_poses.csv"  # Path to the preprocessed features CSV
OUTPUT_MODEL_FILE = "yoga_pose_classifier.joblib"    # File to save the trained model
OUTPUT_DIR = "model_outputs"                         # Directory to save model outputs

# --- Create output directory if it doesn't exist ---
if not os.path.exists(OUTPUT_DIR):
    os.makedirs(OUTPUT_DIR)

# --- Load the engineered features data ---
try:
    df_features = pd.read_csv(INPUT_FEATURES_CSV_FILE)
    print(f"Loaded {len(df_features)} rows from {INPUT_FEATURES_CSV_FILE}")
    print("Class distribution:\n", df_features['pose_label'].value_counts())
except FileNotFoundError:
    print(f"Error: {INPUT_FEATURES_CSV_FILE} ")
    exit()

# --- Prepare Data for Training ---
# Separate features (X) and target (y)
X = df_features.drop( labels: 'pose_label', axis=1)  # All columns except 'pose_label' are features
y = df_features['pose_label']

# Check for any NaN values after feature engineering
if X.isnull().any().any():
    print("Warning: NaN values found in features. Filling with mean values.")
    X = X.fillna(X.mean())  # Simple imputation

# Encode string labels to integers
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
```




```
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.

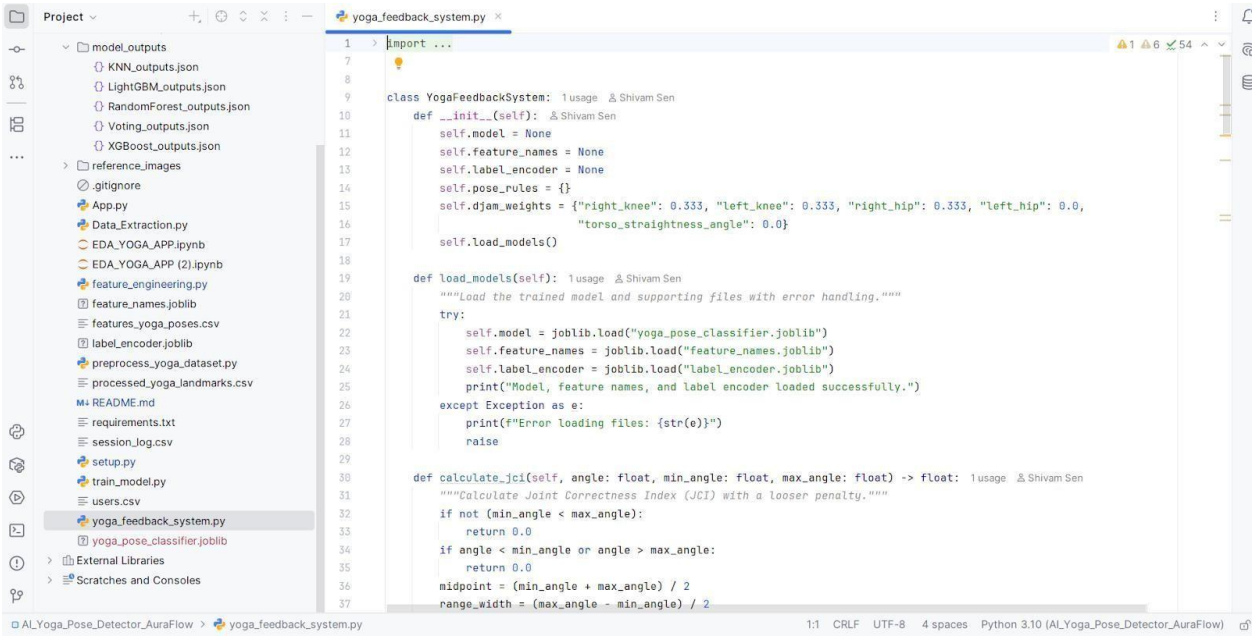
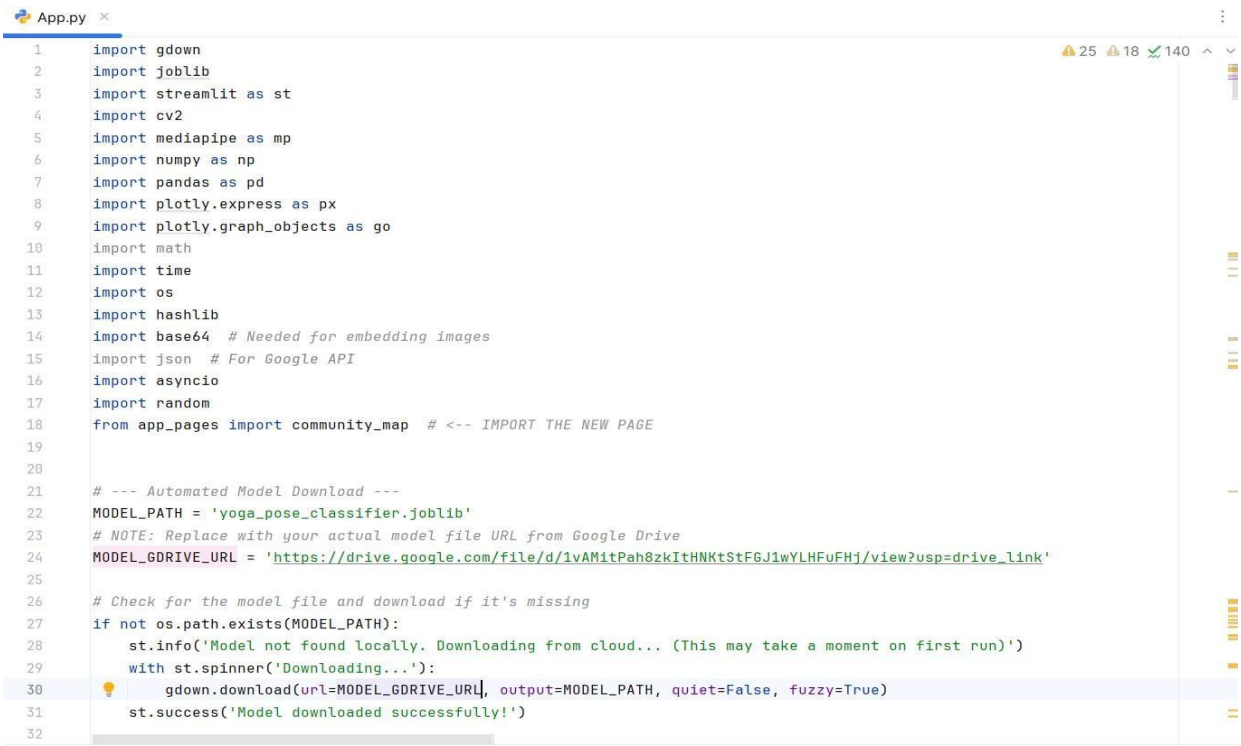


Figure 8: Yoga_feedback_system.py

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



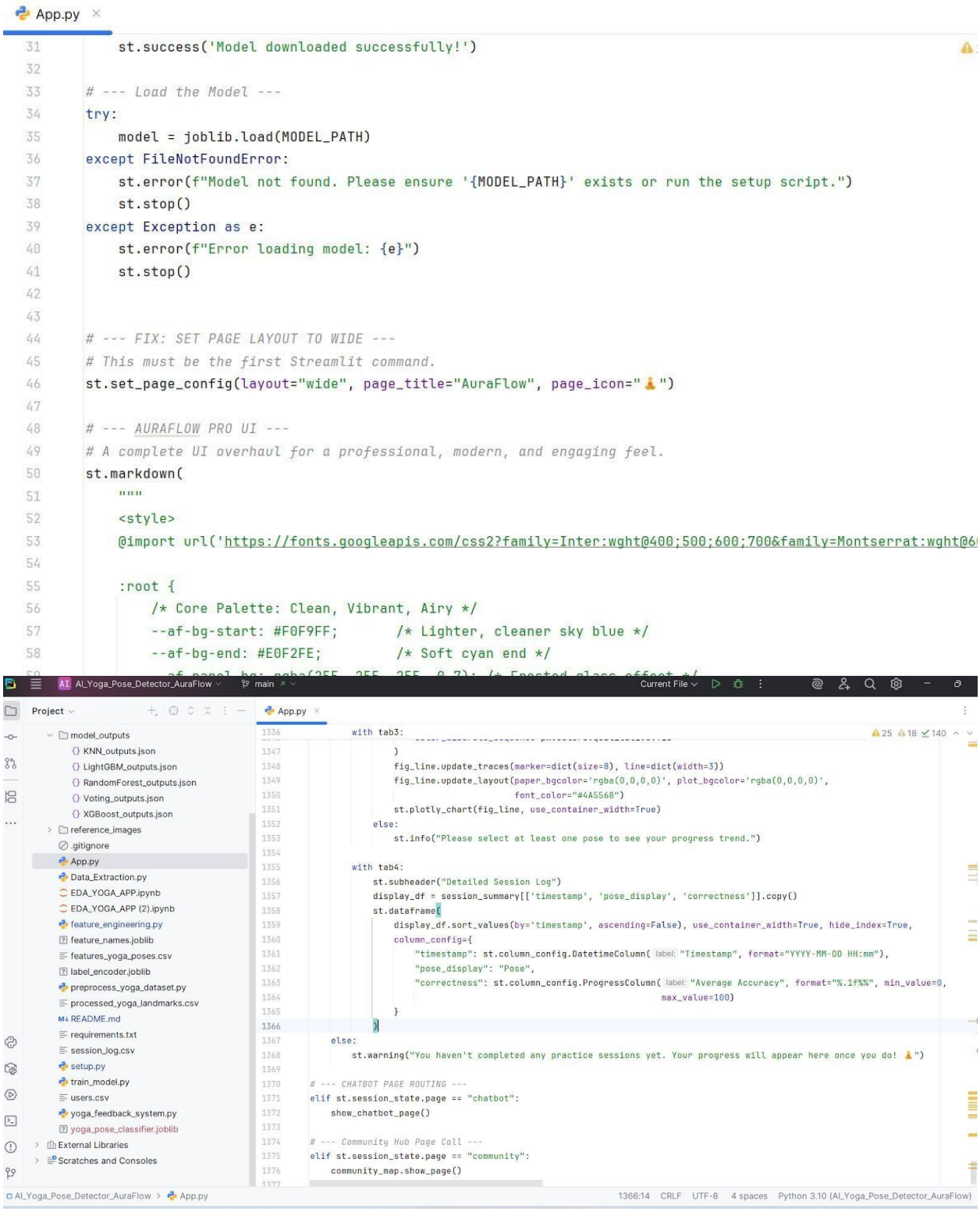


Figure 9: App.py file a glimpse

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"

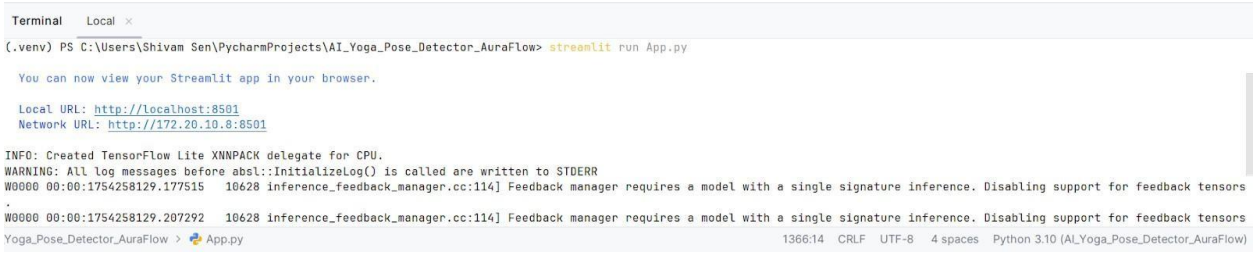
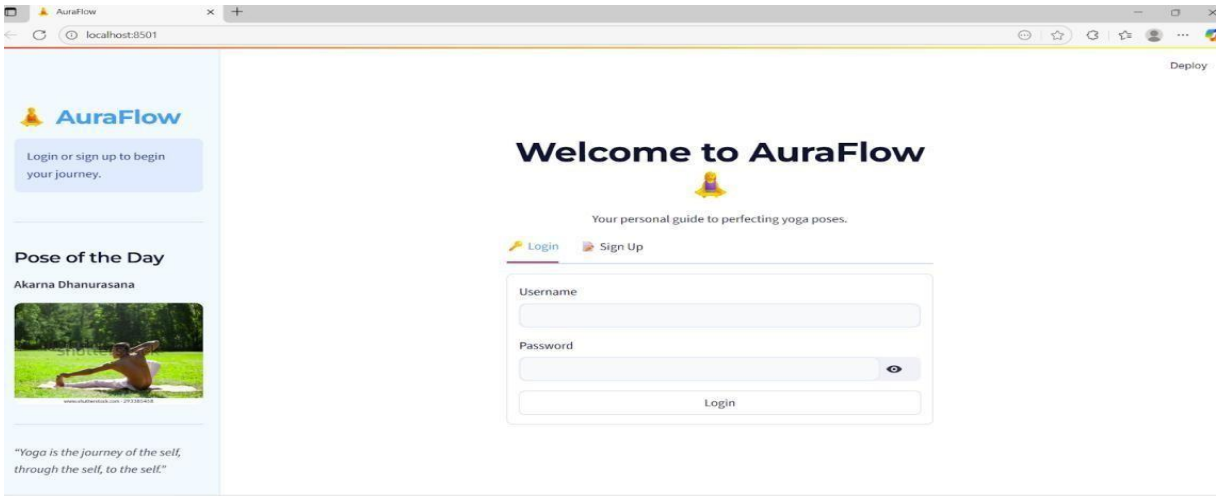


Figure 10: Running the Application

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.

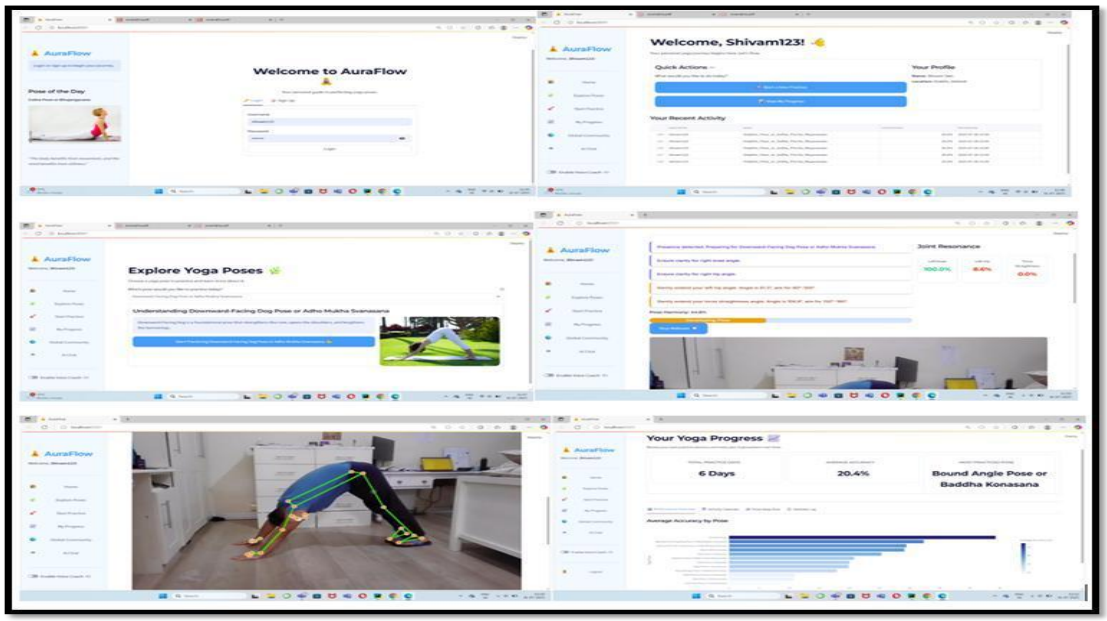


Figure 11: Main Application Interface

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