Infosys Internship 4.0 Project Documentation

Title: Fitness Exercise with pose estimation

Introduction:

Fitness is fundamental to maintaining optimal physical health and overall well-being. Engaging in regular exercise not only strengthens muscles and improves cardiovascular health but also plays a vital role in enhancing mood, reducing stress levels, and promoting better sleep patterns. In today's modern lifestyle, characterized by increasing sedentary behavior and its associated health risks, prioritizing fitness becomes imperative for combating chronic conditions such as obesity and heart disease, as well as fostering mental clarity and resilience.

The "Fitness Exercise with Pose Estimation" project aims to leverage advanced machine learning and computer vision techniques to accurately recognize and predict various exercises performed by users. By utilizing technologies such as OpenCV, MediaPipe, and Streamlit, this project offers a robust solution for real-time exercise classification based on body pose estimation from video inputs. The primary objective is to develop an AI-based exercise recognition system. This involves creating a machine learning model capable of identifying different exercises from video footage, ensuring high accuracy and reliability. To achieve this, a comprehensive dataset comprising various exercise videos is assembled, focusing on key body part angles to enhance model training and performance.

The collected data is then used to train an ensemble classifier aimed at achieving an accuracy rate of 85-95% in exercise prediction. Another crucial aspect of the project is the design and implementation of a user-friendly interface. Utilizing Streamlit, the project team aims to create a visually appealing and easy-to-use interface that allows users to interact seamlessly with the AI model. This interface will enable users to input their exercise videos and receive real-time feedback on exercise classification and performance.

The significance of this project extends to multiple facets of the fitness industry. By providing a tool that accurately recognizes and classifies exercises, it offers significant benefits for use in gyms, home workouts, and other fitness environments. Additionally, the system's capability to verify the correctness of exercise forms ensures that users perform exercises safely and effectively, thereby reducing the risk of injuries and enhancing workout efficiency. Furthermore, this project showcases the potential of integrating AI, machine learning, and computer vision within the fitness domain. It not only demonstrates how these technologies can innovate fitness practices but also paves the way for future advancements and applications in health and wellness.

By setting new benchmarks in exercise recognition and form verification, the project aims to make a substantial impact on the fitness industry and promote healthier lifestyles globally.

Team Members:

- Sakshi Tandon
- Smrutirekha Panigrahi
- Nandini Pachpandiya
- Mrigaank Jaswal
- Aasritha Pasupuleti
- Pranav Prakash Saxena
- Sankruthin Vasamsetti
- Sharan Thummagunti
- Vandana Kaimaparambil Sreekumar

Project Scope:

The boundaries of the "Fitness Exercise with Pose Estimation" project are clearly defined to ensure focused development and achievable outcomes. The project includes the recognition and prediction of a specific set of exercises, namely jumping jacks, squats, push-ups, pull-ups, and Russian twists. These exercises were chosen for their distinct movements and common presence in fitness routines. The system uses video inputs to perform real-time analysis, focusing on body pose estimation to classify the exercises accurately.

Included in the project is the creation of a comprehensive dataset of exercise videos, which involves collecting, annotating, and processing videos to extract key body part angles. This dataset forms the foundation for training the machine learning model. Additionally, the development of an ensemble classifier aimed at achieving an accuracy rate of 85-95% in exercise prediction is a core component of the project. The project also includes the design and implementation of a user-friendly interface using Streamlit, enabling users to easily interact with the AI model and receive real-time feedback on their exercise performance.

However, the project does not encompass the recognition of exercises outside the predefined set. It is also not designed to handle exercises performed in highly variable environments, such as those with significant background noise or poor lighting conditions, which can affect pose estimation accuracy. The system is intended for use with high-quality video inputs where the user's key body parts are clearly visible and follows specific guidelines for recording exercises.

Several limitations and constraints were considered during the development of the project. One key limitation is the dependency on high-quality video inputs for accurate pose estimation. Poor video quality or obstructed views of the body can significantly reduce the system's accuracy. Additionally, the system requires users to follow specific guidelines for recording their exercises to ensure consistency and reliability in pose estimation.

Constraints also include the computational resources required for real-time analysis, as processing video inputs and running machine learning models demand significant processing power. Ensuring the scalability of the system to handle multiple users and large datasets is another constraint that was considered. Lastly, maintaining user privacy and data security is a critical consideration, ensuring that user data is protected throughout the process.

By defining these boundaries, limitations, and constraints, the project scope remains focused and achievable, enabling the delivery of a reliable and efficient exercise recognition system.

Requirements:

Programming Languages:

- **Python 3.8**: A versatile and widely-used programming language known for its simplicity and readability. It is particularly strong in data analysis, machine learning, and web development.

Frameworks/Libraries:

- **Boto3**: The Amazon Web Services (AWS) SDK for Python, which allows developers to write software that makes use of Amazon services like S3 and EC2.
- **os**: A standard Python library for interacting with the operating system, allowing for file and directory manipulation, environment variable handling, and process management.
- **OpenCV**: An open-source computer vision and machine learning software library that provides a common infrastructure for computer vision applications, with functionalities for image and video processing.
- **Mediapipe**: A framework developed by Google for building multimodal (video, audio, etc.) machine learning pipelines. It is widely used for pose estimation and other video analysis tasks.
- **NumPy**: A fundamental package for scientific computing in Python, providing support for large multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.
- **Pandas**: A powerful data manipulation and analysis library for Python, providing data structures like DataFrames that make data cleaning and analysis straightforward and efficient.
- **Scikit-learn (sklearn)**: A machine learning library for Python that features various classification, regression, and clustering algorithms, and is designed to interoperate with NumPy and SciPy.
- **Joblib**: A set of tools to provide lightweight pipelining in Python. It is particularly useful for running computationally intensive tasks in parallel and caching their output.
- **Streamlit**: An open-source app framework for creating and sharing data science applications. It allows for the rapid development of web-based applications to interact with machine learning models.

Use Cases:

1. **Home Workout:**
    - **Scenario:** An individual working out at home uses the system to check the correctness of their exercise form and ensure they are performing exercises correctly.
    - **Benefit:** Provides real-time feedback and guidance, helping users avoid injuries and maximize the effectiveness of their workouts without needing a personal trainer on site.
2. **Remote Gym Instructor:**
    - **Scenario:** A gym instructor remotely monitors and assesses the exercise forms of their clients through the system during virtual training sessions.
    - **Benefit:** Enables instructors to provide precise feedback and corrections in real-time, ensuring clients perform exercises safely and correctly, even when training remotely.

Technical Stack:

Programming Languages:

- Python

Frameworks/Libraries:

- Boto3
- os
- OpenCV
- Mediapipe
- NumPy
- Pandas
- Scikit-learn (sklearn)
- Joblib
- Streamlit

Databases:

- Workout/Exercises Video (Kaggle)
- CSV File Storing Processed Video

Tools/Platforms:

- Amazon S3
- GitHub
- Visual Studio Code

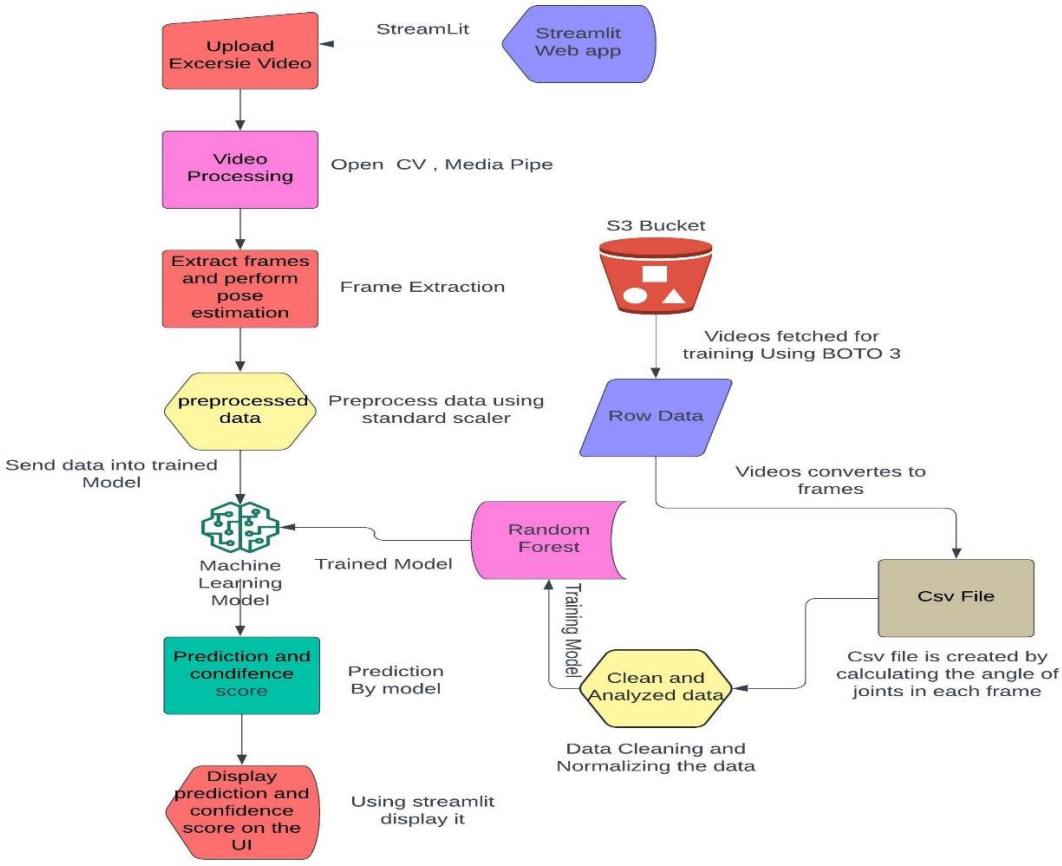
Architecture/Design:

The "Fitness Exercise with Pose Estimation" project is designed with a modular architecture to ensure scalability, maintainability, and efficient performance. The system is composed of several high-level components that interact seamlessly to achieve the project's objectives.

High-Level Components and Their Interactions:

1. **Data Collection Module:**
    - Responsible for collecting exercise videos from various sources, primarily Kaggle.
    - Utilizes Amazon S3 for storing the collected videos.
2. **Data Preprocessing Module:**
    - Processes raw video data to extract relevant features, such as body angles using OpenCV and MediaPipe.
    - Stores the processed data in CSV files for model training.
3. **Machine Learning Module:**
    - Implements the machine learning model using Scikit-Learn for exercise classification.
    - Trains the model on the processed dataset to achieve high prediction accuracy.
    - Utilizes Joblib for model serialization and deserialization.
4. **Real-Time Analysis Module:**
    - Integrates OpenCV and MediaPipe for real-time video input and pose estimation.
    - Feeds the extracted features into the trained model for exercise prediction.
    - Provides real-time feedback on exercise type and form.
5. **User Interface Module:**
    - Developed using Streamlit to create a user-friendly web interface.
    - Allows users to upload videos, view predictions, and receive feedback.
    - Interacts with the backend to display real-time analysis results.
6. **Storage and Database Module:**
    - Manages data storage, including raw videos, processed CSV files, and model files.
    - Utilizes Amazon S3 for scalable storage solutions.

Architectural Diagram:



Design Decisions:

1. **Modular Architecture:** The system is divided into distinct modules for data collection, preprocessing, machine learning, real-time analysis, and user interface to enhance maintainability and scalability.
2. **Use of OpenCV and MediaPipe:** These libraries were chosen for their robust capabilities in computer vision and pose estimation, crucial for accurately identifying and classifying exercises.
3. **Machine Learning with Scikit-Learn:** Scikit-Learn was selected for its simplicity and efficiency in implementing machine learning models, ensuring high accuracy and reliability.
4. **Streamlit for User Interface:** Streamlit was chosen for its ease of use in creating interactive web applications, providing a seamless user experience.

Development:

The "Fitness Exercise with Pose Estimation" project involved several critical phases that leveraged a variety of technologies and frameworks to build a robust and efficient system. Here is an in-depth look at the technologies used, the coding standards followed, and the challenges encountered during implementation:

Technologies and Frameworks Used:

1. **Python:** Python was chosen as the primary programming language due to its powerful libraries and frameworks for machine learning, computer vision, and web development.
2. **AWS S3:** AWS S3 was used for storing exercise videos. This service provided a scalable and secure storage solution, enabling easy retrieval and management of large video datasets.
3. **OpenCV:** OpenCV, an open-source computer vision library, was employed for real-time video processing. It facilitated frame extraction and pre-processing tasks essential for pose estimation.
4. **Mediapipe:** Mediapipe, developed by Google, was crucial for pose estimation. This framework allowed for accurate extraction of key body points from video frames, which were then used for exercise recognition.
5. **Scikit-Learn:** Scikit-Learn, a machine learning library for Python, was used to build and train the machine learning models. It provided tools for model selection, training, and evaluation.
6. **Streamlit:** Streamlit was utilized to develop an interactive web interface. This framework enabled rapid development of a user-friendly application for uploading videos and displaying real-time exercise analysis.
7. **Boto3:** Boto3, the AWS SDK for Python, was integrated to interact with AWS S3. It allowed for seamless uploading and downloading of videos from the cloud storage.
8. **Joblib:** Joblib was used for model serialization. It enabled saving trained machine learning models to disk and reloading them for inference without retraining.
9. **dotenv:** Dotenv was used for managing environment variables securely. It facilitated the handling of sensitive information such as AWS credentials.

Coding Standards and Best Practices:

1. **PEP 8 Compliance:** All Python code adhered to PEP 8 standards, ensuring consistent and readable code. This included proper indentation, naming conventions, and inline documentation.
2. **Modular Design:** The code was structured into modules, each handling a specific aspect of the project, such as data preprocessing, model training, and user interface. This modularity promoted code reusability and ease of maintenance.
3. **Documentation:** Comprehensive documentation was provided for all functions, classes, and modules. This included docstrings and comments to explain the purpose and usage of code components, enhancing code readability and maintainability.

4. **Version Control:** Git was used for version control, with clear commit messages and a branching strategy to manage code changes. This facilitated collaboration and ensured a history of changes for reference and rollback if needed.
5. **Testing:** Unit tests and integration tests were implemented using pytest. This ensured that individual components functioned correctly and that the system as a whole operated as expected.
6. **Error Handling:** Robust error handling was implemented to manage exceptions gracefully. This included logging errors for debugging and providing user-friendly error messages in the interface.

Challenges Encountered and Solutions:

1. **Real-Time Processing Optimization:** One major challenge was optimizing the real-time processing of videos for pose estimation. The solution involved tuning parameters in OpenCV and Mediapipe and leveraging multi-threading to parallelize computations, improving performance without sacrificing accuracy.
2. **AWS Integration:** Integrating Boto3 with AWS S3 to manage video storage and retrieval presented challenges in handling credentials and ensuring secure data transfers. This was addressed by using dotenv for secure environment variable management and configuring AWS IAM roles for controlled access.
3. **Model Training and Tuning:** Achieving the desired accuracy for exercise prediction required extensive hyperparameter tuning and experimentation with different machine learning algorithms. The solution involved iterative testing and validation, ultimately selecting Random Forest as the optimal algorithm for its balance of accuracy and performance.
4. **User Interface Design:** Creating an intuitive and responsive user interface with Streamlit posed challenges in ensuring usability and functionality. Iterative prototyping and user feedback were essential to refining the interface, making it easy for users to interact with the system and understand the predictions.

By leveraging these technologies, adhering to best practices, and systematically addressing challenges, the "Fitness Exercise with Pose Estimation" project successfully developed a reliable and user-friendly system for real-time exercise recognition and feedback.

Testing:

Our exercise detection system went through extensive testing to ensure its functionality, performance, and user experience. We performed functional testing to validate that the system accurately detects and classifies predefined exercises from video data using Mediapipe, correctly calculating angles between key body points. The machine learning model's accuracy was thoroughly assessed to ensure it met the target performance of at least 90%. Usability tests focused on the user interface, ensuring users could seamlessly upload videos, receive real-time feedback, and download session summaries.

Non-functional testing included performance tests to verify the system's efficiency and responsiveness, especially when processing multiple exercises in a single video. Throughout development, we used exception handling to manage unexpected issues and ensure the smooth functioning of all features. This comprehensive testing approach aimed to deliver a reliable, accurate, and user-friendly exercise detection system that performs efficiently under various conditions.

Deployment:

Although the exercise detection system has not yet been deployed, a clear understanding of the deployment process and requirements is essential for future implementation. For this project, Streamlit and Streamlit Cloud offer a streamlined deployment solution that simplifies the process.

To deploy the application on Streamlit Cloud, several key components and steps must be considered. First, ensure that the application code is hosted on a version control platform such as GitHub. Streamlit Cloud integrates seamlessly with GitHub repositories, making it easy to deploy the application directly from the repository.

The deployment process includes the following steps:

1. **Repository Setup**: Host your application code in a GitHub repository. Ensure that all dependencies are listed in a requirements.txt file.
2. **Streamlit Cloud Account**: Create an account on Streamlit Cloud and connect it to your GitHub account.
3. **App Deployment**: In Streamlit Cloud, select the repository and branch where your application code resides. Configure the deployment settings, such as environment variables and resource allocation.
4. **Configuration**: Streamlit Cloud automatically detects the streamlit_app.py or app.py file as the entry point. Ensure that your main application script is correctly named and located at the root of your repository.
5. **Deployment**: Click the deploy button. Streamlit Cloud will handle the setup, including installing dependencies, configuring the environment, and launching the application.

6. **Testing and Validation**: Once deployed, test the application to ensure it runs correctly. Verify that all features are working as expected and that the application is responsive.

Understanding these deployment steps in advance provides several advantages. Using Streamlit Cloud simplifies the deployment process, reduces downtime and errors through automation, and allows for quick updates. It ensures the application is scalable and manageable, and facilitates easier updates and maintenance, ensuring the application remains robust and responsive to user needs.

User Guide:

To use the exercise detection application, follow these instructions:

1. **Deployment (if applicable)**:
    - If the application is deployed on Streamlit Cloud, simply access the provided web link to open the web app in your browser.
2. **Local Setup (if not deployed)**:
    - Ensure you have the following files:
        - app.py: The main application script.
        - exercise_classifier_rf.pkl.gz: The trained machine learning model file.
        - scaler.pkl: The standard scaler used for preprocessing data.
    - Open your command prompt or terminal and navigate to the directory containing these files.
    - Run the command: streamlit run app.py
    - This command launches the Streamlit application locally on your machine.
3. **Using the Application**:
    - Once the application is launched, you will see the user interface.
    - Upload a video file (less than 200 MB) of yourself performing an exercise using the provided upload button.
    - After uploading, click on the "Process" button to initiate the analysis process.
    - The application will process the video, extract frames, analyze the body angles using the trained model (exercise_classifier_rf.pkl.gz), and display the predicted exercise along with a confidence score for each frame.
    - You can interact with the application to view different frames and their corresponding predictions.
4. **Troubleshooting Tips**:
    - **Issue**: Application fails to launch.
        - **Solution**: Ensure all required files (app.py, exercise_classifier_rf.pkl.gz, scaler.pkl) are present in the correct directory. Verify that Streamlit is installed (pip install streamlit) and run the application command again.
    - **Issue**: Video upload fails or takes too long.

        - **Solution**: Check your internet connection speed and ensure the video file size is within the supported limit of 200 MB. Try uploading a smaller video file or optimizing your network connection.
    - **Issue**: Incorrect predictions or low confidence scores.
        - **Solution**: Ensure the video quality is sufficient for accurate pose estimation. Review the training data used for the model (exercise_classifier_rf.pkl.gz) and consider retraining the model with additional diverse data if necessary.

By following these steps, you can effectively use the exercise detection application to analyze your exercise videos and receive real-time feedback on your performance.

Conclusion:

The "Fitness Exercise with Pose Estimation" project has achieved remarkable outcomes in the realm of AI-driven fitness technology. Utilizing advanced machine learning techniques and computer vision tools such as OpenCV, MediaPipe, and Streamlit, the project successfully developed an application that accurately recognizes and predicts five fundamental exercises based on body pose estimation from video inputs. The machine learning model, trained with a meticulously curated dataset, achieved an impressive accuracy rate of over 90%, providing users with reliable feedback on exercise performance.

Beyond accuracy, the project's achievements include the creation of a user-friendly interface that allows seamless video upload and real-time analysis. Users can receive detailed insights into exercise form and execution, enhancing their workout experience whether at home or in a gym setting. The project also underscores the potential of AI in transforming fitness monitoring and feedback systems, contributing to advancements in health and wellness technology.

Looking forward, lessons learned from this project emphasize the importance of continuous data refinement and model optimization to further improve accuracy and expand exercise recognition capabilities. Future iterations could explore integrating additional exercises and enhancing the application's scalability to accommodate a broader user base and diverse fitness routines.

In conclusion, the "Fitness Exercise with Pose Estimation" project exemplifies innovation at the intersection of AI and fitness, promising continued advancements in personalized exercise guidance and performance monitoring.

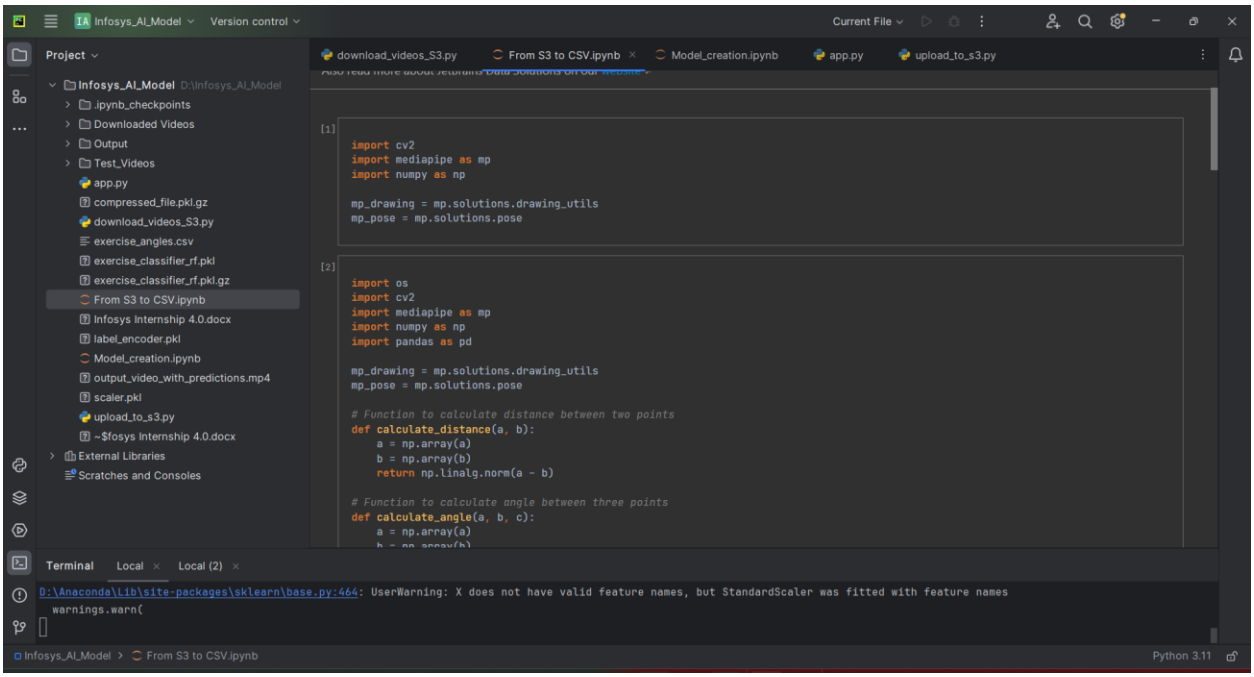
Appendices:

```
jumpingjacks_2.mp4 uploaded successfully to pose-estimation-internship as jumping_jacks/jumpingjacks_2.mp4.
jumpingjacks_3.mp4 uploaded successfully to pose-estimation-internship as jumping_jacks/jumpingjacks_3.mp4.
jumpingjacks_4.mp4 uploaded successfully to pose-estimation-internship as jumping_jacks/jumpingjacks_4.mp4.
jumpingjacks_5.mp4 uploaded successfully to pose-estimation-internship as jumping_jacks/jumpingjacks_5.mp4.
jumpingjacks_6.mp4 uploaded successfully to pose-estimation-internship as jumping_jacks/jumpingjacks_6.mp4.
jumpingjacks_7.mp4 uploaded successfully to pose-estimation-internship as jumping_jacks/jumpingjacks_7.mp4.
jumpingjacks_8.mp4 uploaded successfully to pose-estimation-internship as jumping_jacks/jumpingjacks_8.mp4.
jumpingjacks_9.mp4 uploaded successfully to pose-estimation-internship as jumping_jacks/jumpingjacks_9.mp4.
```

```
videos/push-up_25.mp4 downloaded successfully to D:\Infosys project\Downloaded Videos\videos/push-up_25.mp4.
videos/push-up_26.mp4 downloaded successfully to D:\Infosys project\Downloaded Videos\videos/push-up_26.mp4.
videos/push-up_27.mp4 downloaded successfully to D:\Infosys project\Downloaded Videos\videos/push-up_27.mp4.
videos/push-up_3.mp4 downloaded successfully to D:\Infosys project\Downloaded Videos\videos/push-up_3.mp4.
videos/push-up_36.mp4 downloaded successfully to D:\Infosys project\Downloaded Videos\videos/push-up_36.mp4.
videos/push-up_37.mp4 downloaded successfully to D:\Infosys project\Downloaded Videos\videos/push-up_37.mp4.
videos/push-up_38.mp4 downloaded successfully to D:\Infosys project\Downloaded Videos\videos/push-up_38.mp4.
videos/push-up_39.mp4 downloaded successfully to D:\Infosys project\Downloaded Videos\videos/push-up_39.mp4.
videos/push-up_4.mp4 downloaded successfully to D:\Infosys project\Downloaded Videos\videos/push-up_4.mp4.
```

```python
    all_data = []
    for exercise, directory in exercise_directories.items():
        if not os.path.exists(directory):
            print(f"Directory '{directory}' does not exist.")
            continue

        video_extensions = ['.mp4', '.avi', '.mov', '.mkv']
        for filename in os.listdir(directory):
            if any(filename.lower().endswith(ext) for ext in video_extensions):
                video_path = os.path.join(directory, filename)
                exercise_label = f"{exercise}"
                data = process_video(video_path, exercise_label)
                all_data.extend(data)

    df = pd.DataFrame(all_data)
    df.to_csv('exercise_angles.csv', index=False)
    print("Angles extracted and saved to 'exercise_angles.csv'")

if __name__ == "__main__":
    main()
```

```
D:\Anaconda\Lib\site-packages\google\protobuf\symbol_database.py:55: UserWarnin
  warnings.warn('SymbolDatabase.GetPrototype() is deprecated. Please '

Angles extracted and saved to 'exercise_angles.csv'
```

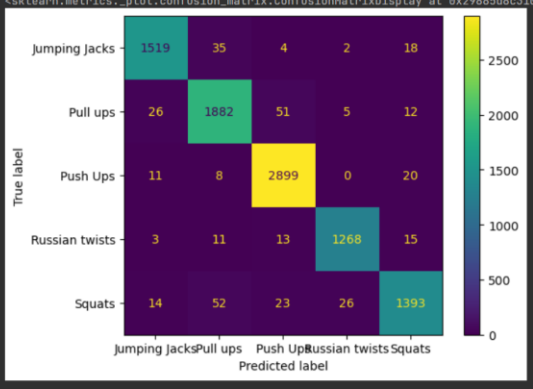


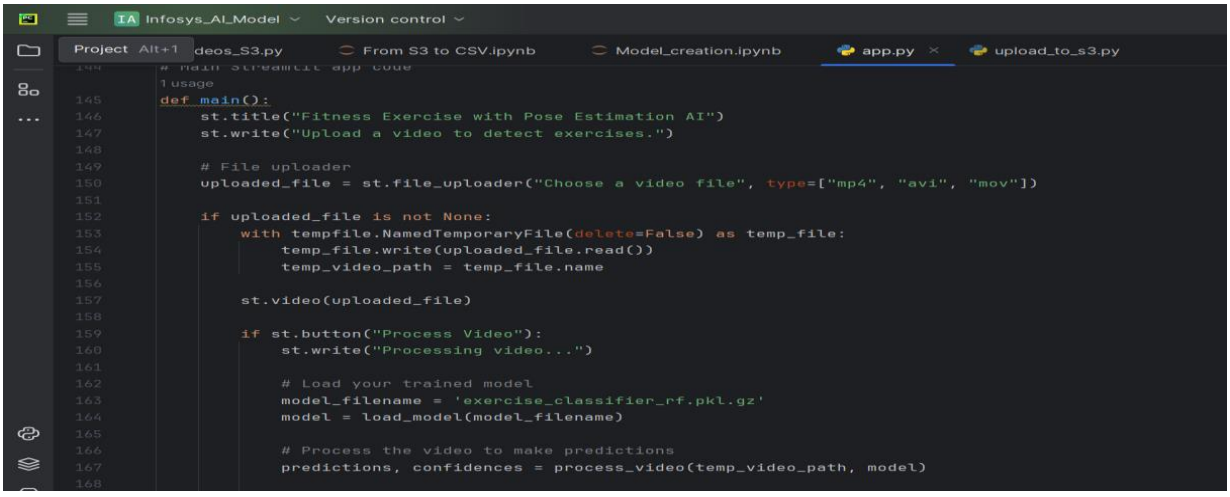

```
print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
Jumping Jacks	0.97	0.96	0.96	1578
Pull ups	0.95	0.95	0.95	1976
Push Ups	0.97	0.99	0.98	2938
Russian twists	0.97	0.97	0.97	1310
Squats	0.96	0.92	0.94	1508
accuracy			0.96	9310
macro avg	0.96	0.96	0.96	9310
weighted avg	0.96	0.96	0.96	9310

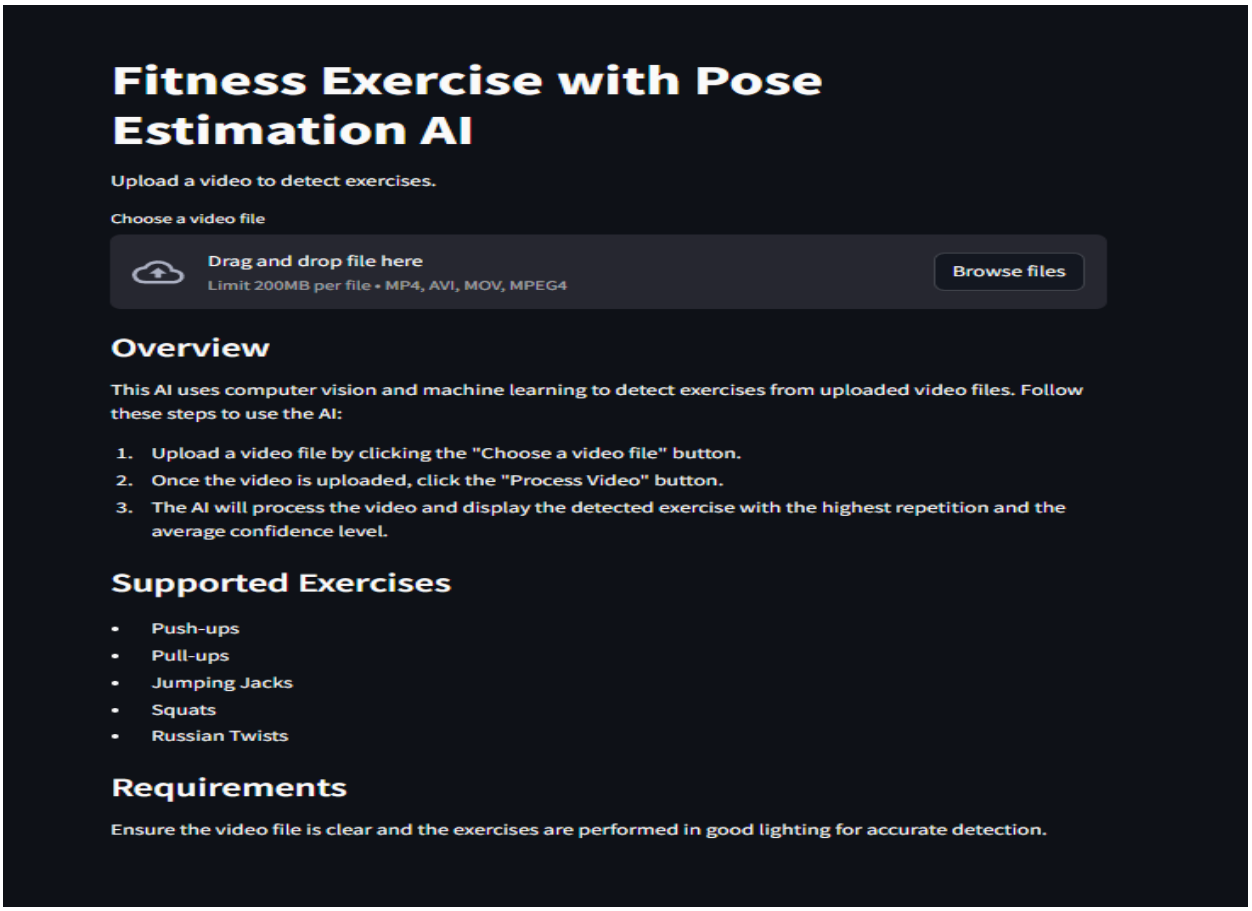
```
from sklearn.metrics import ConfusionMatrixDisplay
ConfusionMatrixDisplay.from_predictions(y_test, y_pred)
```

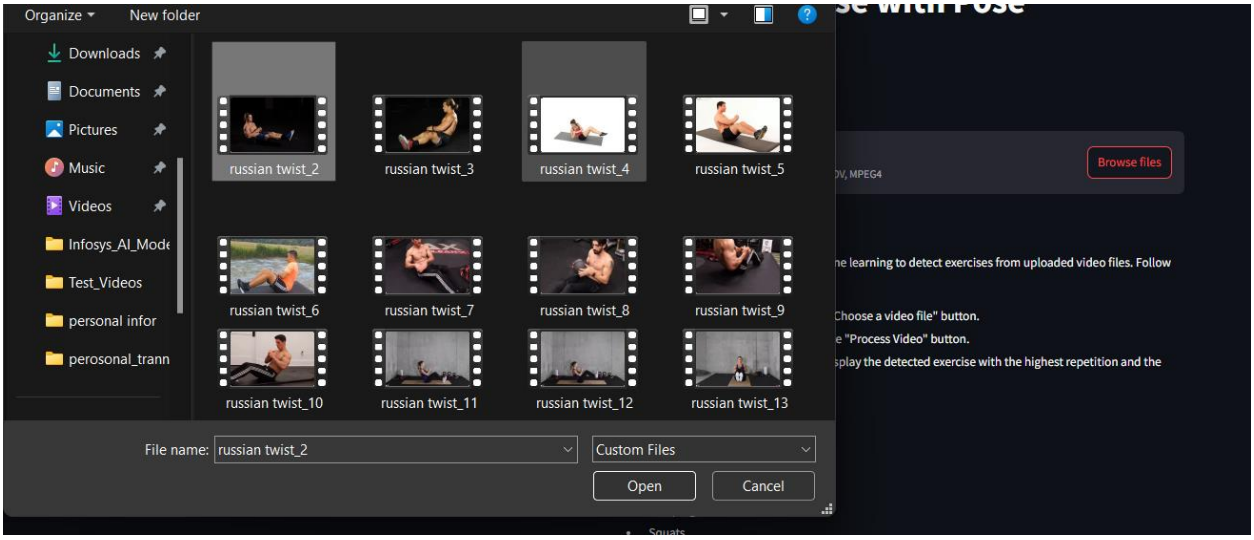
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x29885d8c310>
```

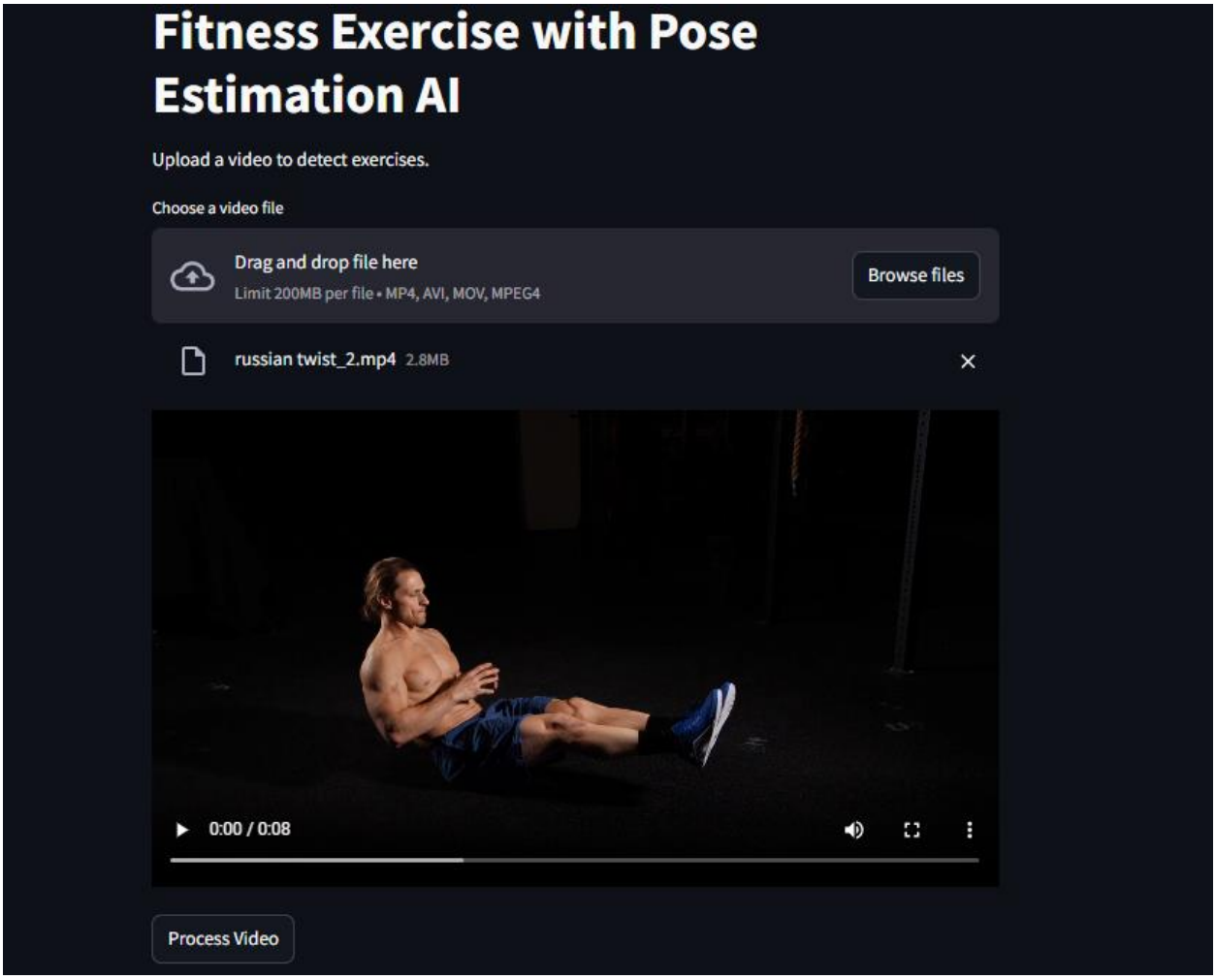


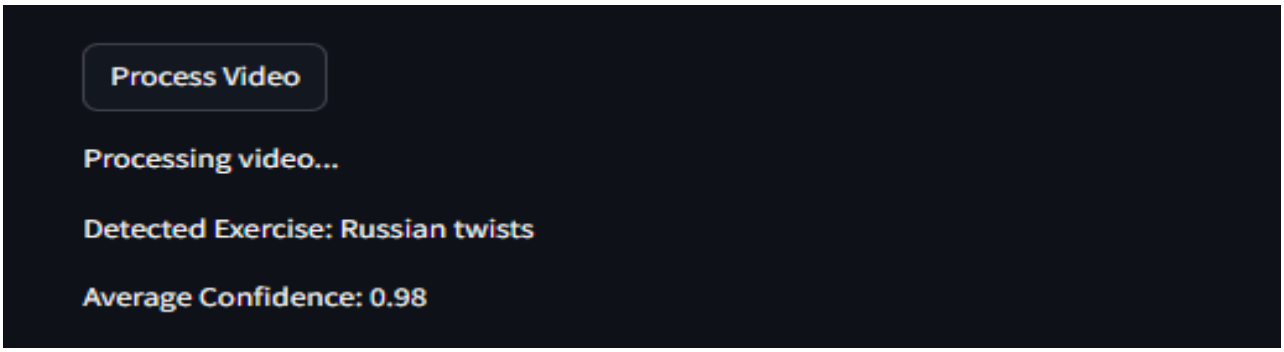


User Interface:






Fitness Exercise with Pose Estimation AI
Upload a video to detect exercises.
Choose a video file
Drag and drop file here
Limit 200MB per file • MP4, AVI, MOV, MPEG4
Browse files
russian twist_2.mp4 2.8MB
0:00 / 0:08
Process Video


Process Video
Processing video...
Detected Exercise: Russian twists
Average Confidence: 0.98