Alessandro Di Patria, Alberto Cotumaccio

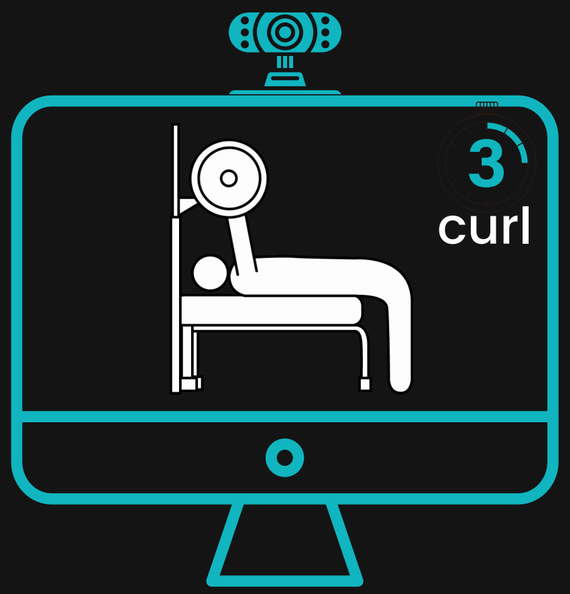
AI Gym Tracker

Multimodal Interaction 2022/2023

SAPIENZA
UNIVERSITÀ DI ROMA

CONTENT

WHAT'S THE IDEA ?



- Track and monitor users' physical activities during real-time workouts
- Provide intuitive and user-friendly interface
- Importance of Correct Technique
  - feedback on execution speed
- Flexibility on training duration
- Complete final overview of exercises and estimation of calories burned

MULTIMODALITY

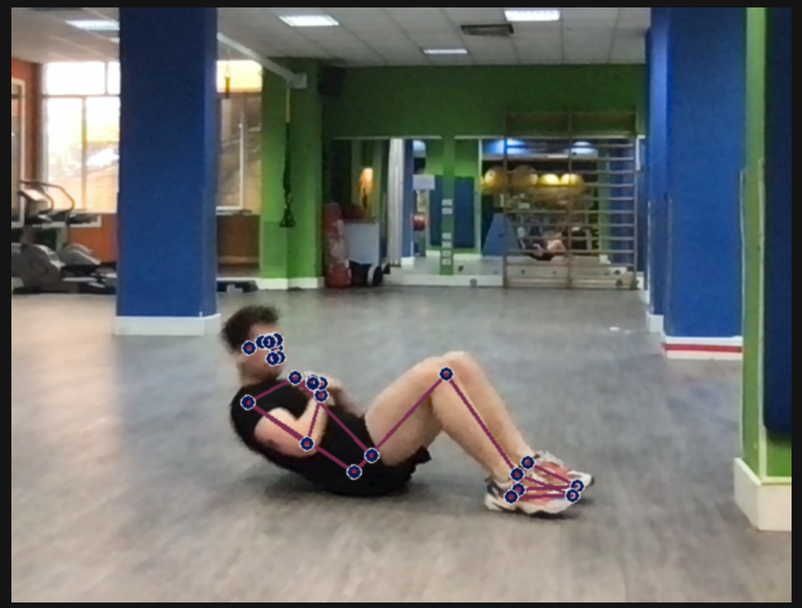


Fig: Gesture-based exercise recognition using body landmarks

- Gestural interaction
  - Webcam-based
  - understanding user's body movements
  - precise exercise recognition with AI
- Enabling Technologies (input)
  - perceptual input
- System response (output)
  - Essential visual output
  - Audio feedback with appropriate sounds

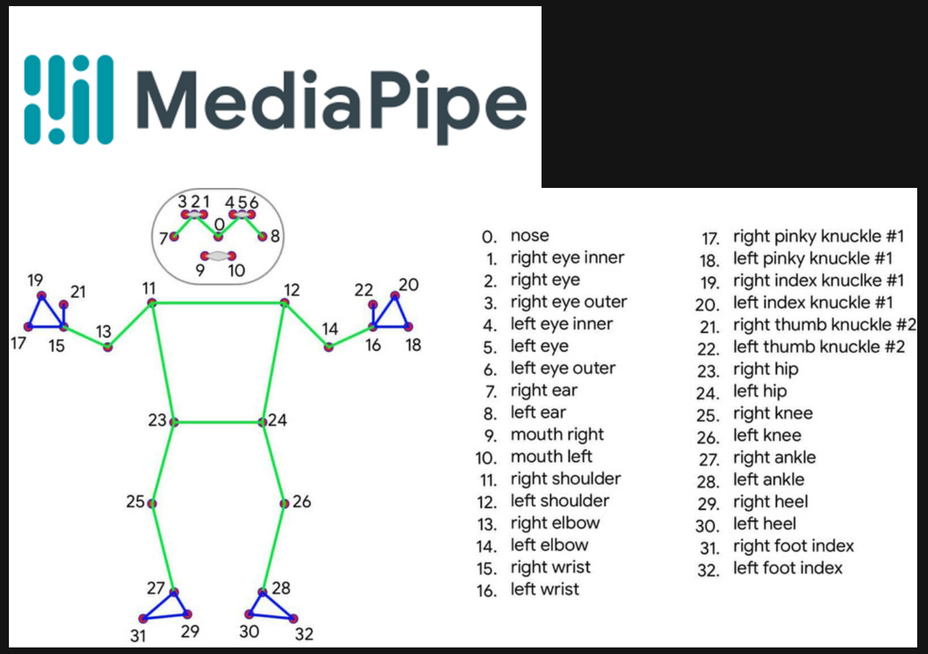
DESIGN AND REQUIREMENTS

Functional Requirements
Personal Data Entry
Automatic Pose Recognition
Real-time Exercise Tracking
Calorie Calculation
Sending Statistics via Email

NonFunctional Requirements
High-quality Camera
Stable Internet Connection
Accessibility to a Web Browser
Ease of Use and Intuitiveness

BACKGROUND

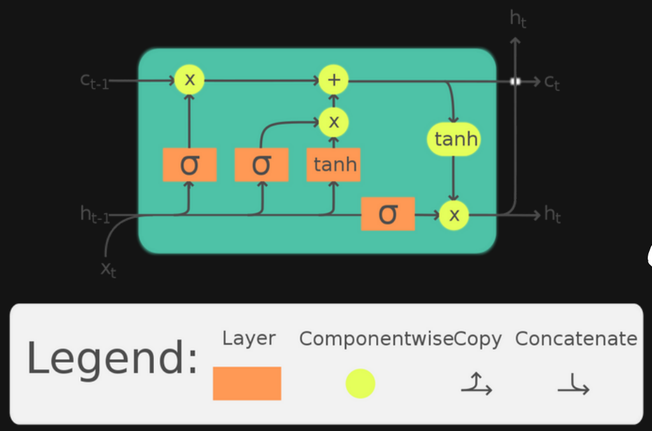
MediaPipe



- Computer vision library to create ML pipelines
- Real-time analysis of video
- For Pose detection MediaPipe Extracts 33 keypoints
- Each keypoint has a specific position in x,y,z space

BACKGROUND (2)

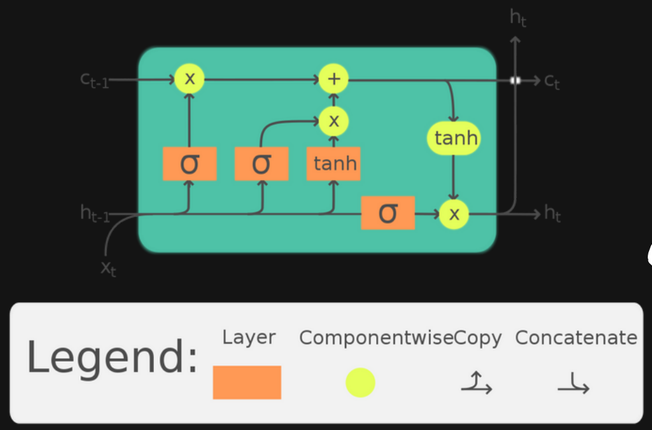
RNN Algoritms (LSTM)



- AI Models designed to process sequential data
- RNNs can be used to classify the action in a video by processing the image frames as a sequence and using the hidden state
- LSTM works well to predict poses using the recurrent neural learning

BACKGROUND (3)

Frontend, Backend Technologies

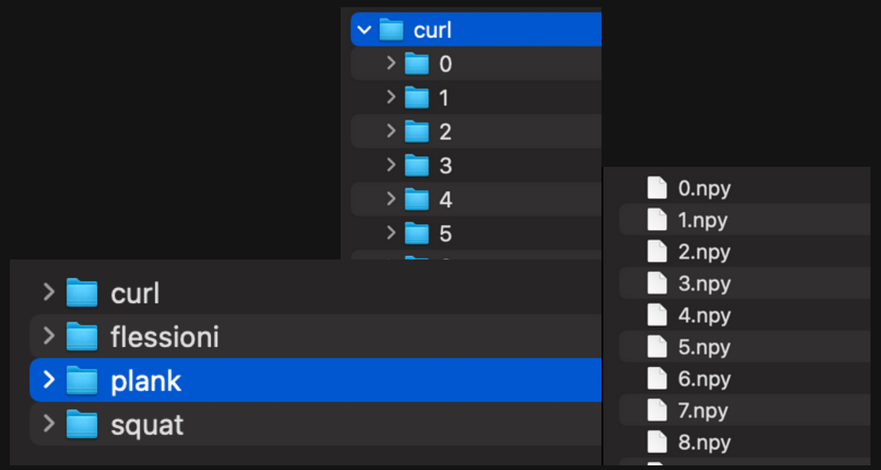


- AI Models designed to process sequential data
- RNNs can be used to classify the action in a video by processing the image frames as a sequence and using the hidden state
- LSTM works well to predict poses using the recurrent neural learning

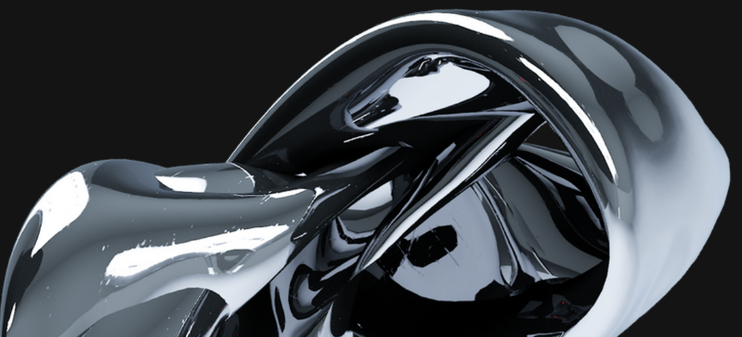
IMPLEMENTATION: data collection

- Autonomous dataset creation of poses
- 4 different pose : **plank**, **squat**, **push-up**, **curl**
- We recorded 60 videos for each pose.
- Create a data for action velocity

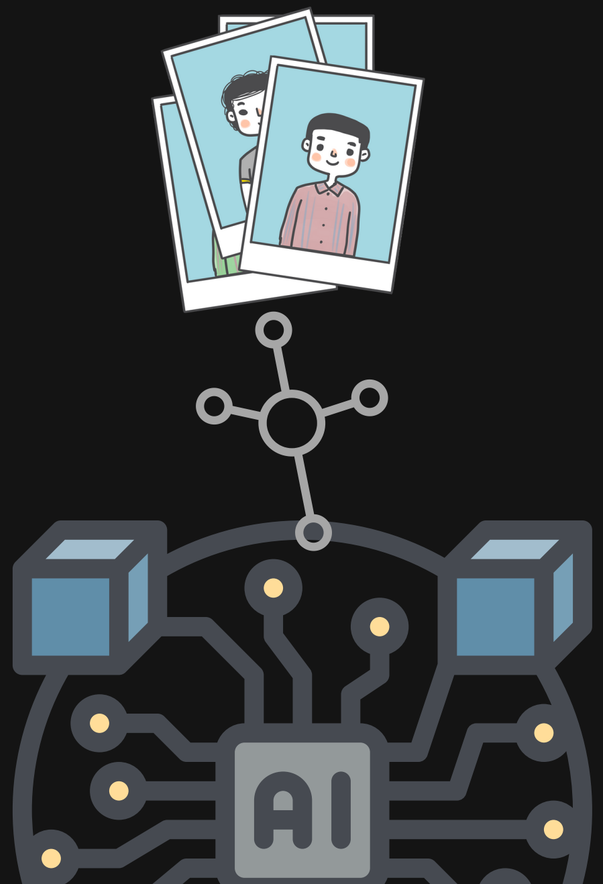
IMPLEMENTATION: data collection (2)



- During data collection we store actions in different folders
- Each folders have inside the videos
- Each video folder have numpy arrays of body position corresponding to each frame

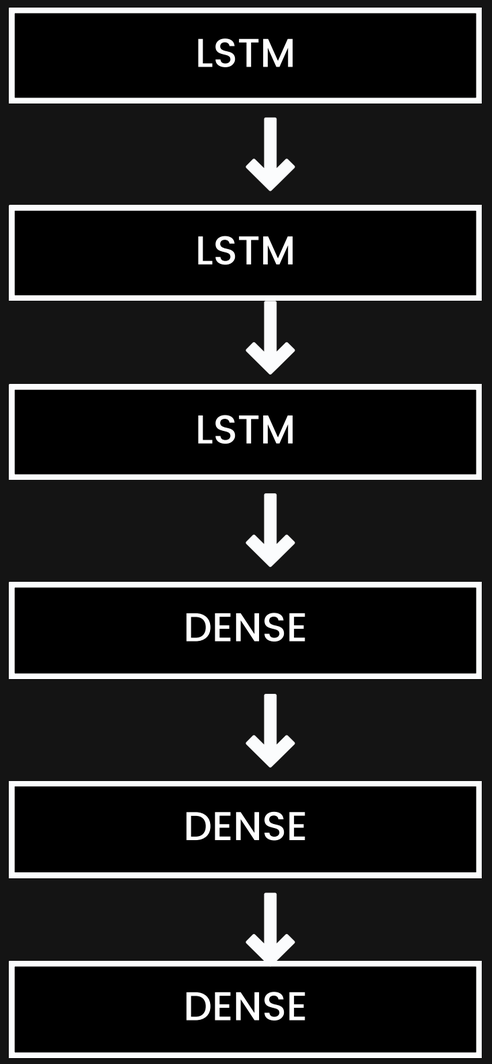


IMPLEMENTATION: model training



- We need multiple video or foto of the same pose.
- Mediapipe extract keypoints for each frame of the video or image and store them in an array
- We Train an LSTM model with the array made of keypoints

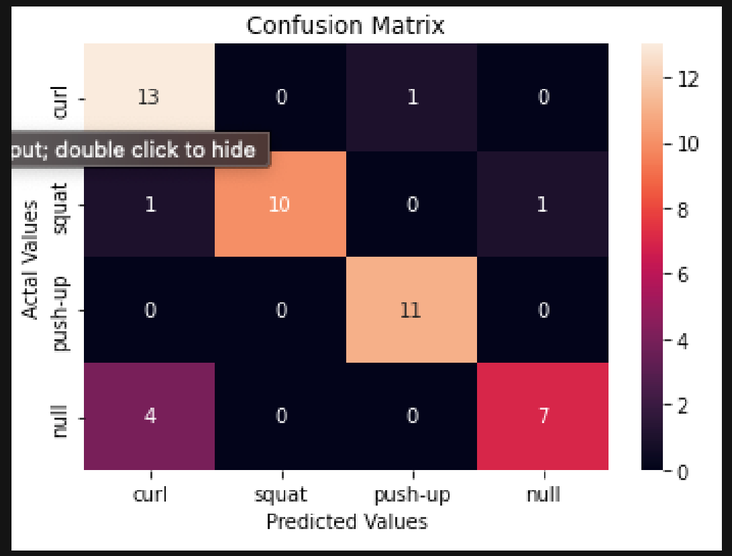
IMPLEMENTATION: model training (2)



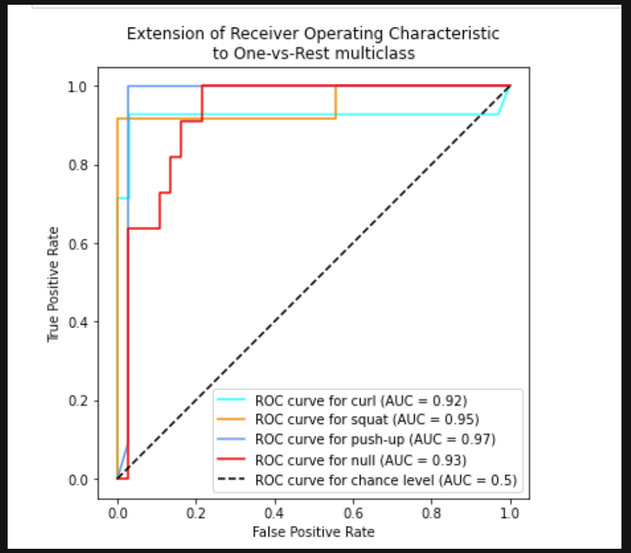

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_3 (LSTM)               (None, 30, 64)            50432

 lstm_4 (LSTM)               (None, 30, 128)           98816

 lstm_5 (LSTM)               (None, 64)                49408

 dense_3 (Dense)             (None, 64)                4160

 dense_4 (Dense)             (None, 32)                2080

 dense_5 (Dense)             (None, 4)                 132

=================================================================
Total params: 205,028
Trainable params: 205,028
Non-trainable params: 0
```

- We split train and test data : 60% 40,%
- Initially Fit the model on 300 epochs
- Tune model to find best params

IMPLEMENTATION: evaluation

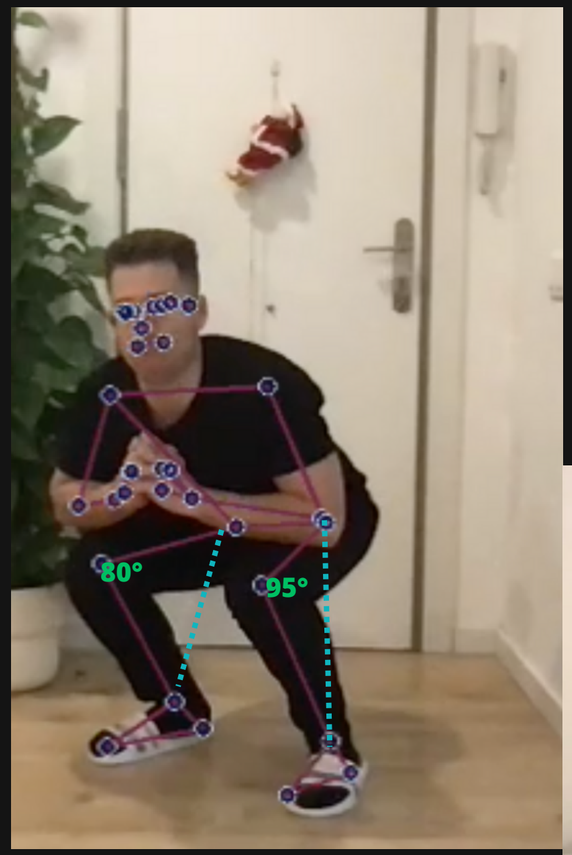


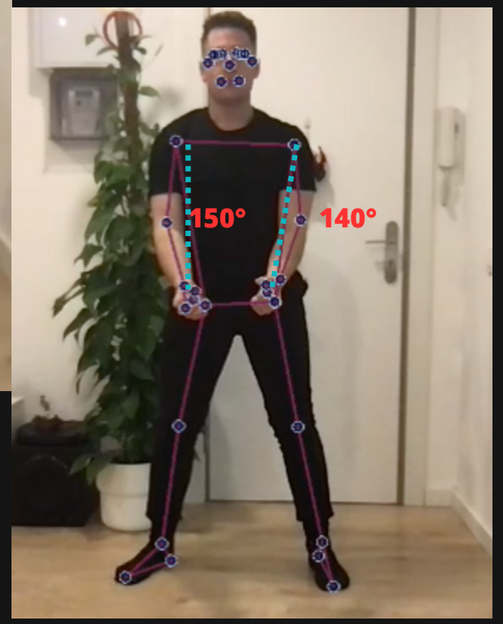
	precision	recall	f1-score	support
curl	0.72	0.93	0.81	14
squat	1.00	0.83	0.91	12
push-up	0.92	1.00	0.96	11
null	0.88	0.64	0.74	11
accuracy			0.85	48
macro avg	0.88	0.85	0.85	48
weighted avg	0.87	0.85	0.85	48



- In most cases the submitted instances were classified correctly
- Accuracy around 85%
- the model has high sensitivity and high specificity

IMPLEMENTATION: compute angle



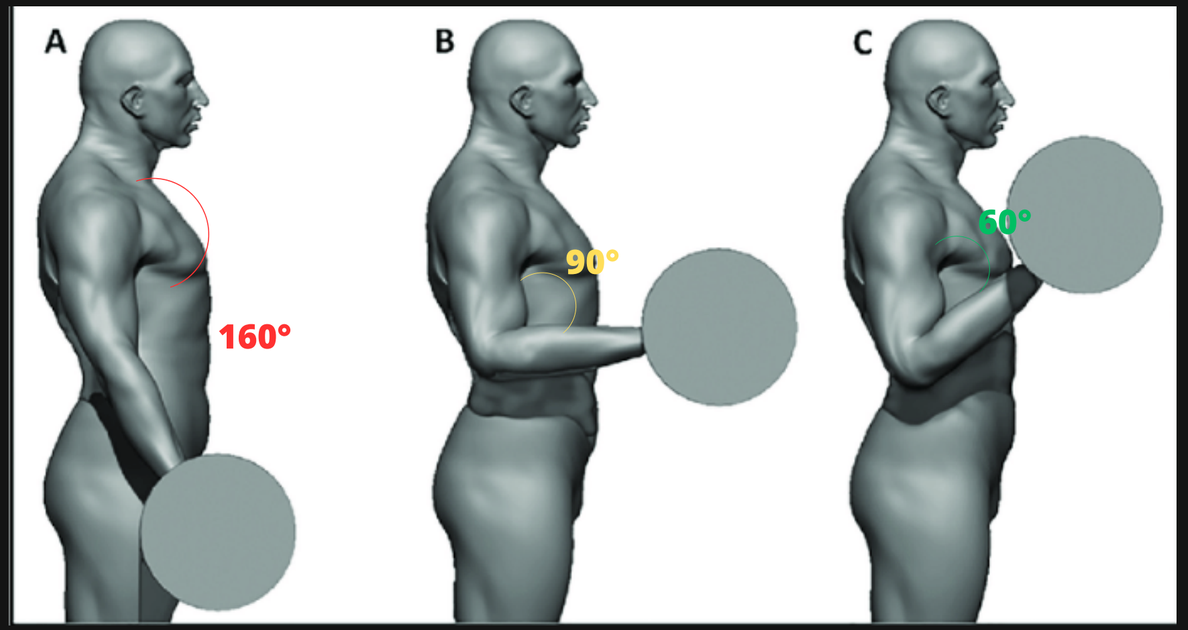


- Compute the angle between 3 landmarks points
- In order to count an exercise I set two threshold-angle one for stage down one for stage up.
- When both stage down and up are done we achieve one repetition

General formula

```
result = atan2(P3.y - P1.y, P3.x - P1.x) -
         atan2(P2.y - P1.y, P2.x - P1.x);
```

IMPLEMENTATION: real-time algorithm



- Determine the execution time of the exercise easily by calculating the difference between the two timestamps of the two ordered stages
- Evaluate each execution of repetition through a time threshold based on real action dataset (perfect, good, too quick)

Stage DOWN

Trashold = 150°

Start timer

Stage UP

Trashold = 70°

End timer

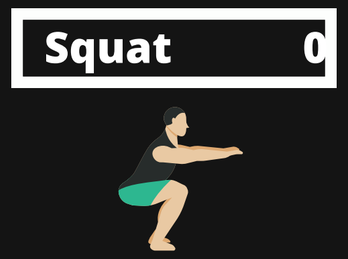
=

1REPS

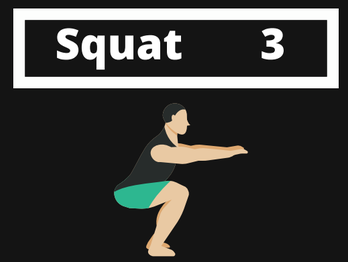
3,5s

PERFECT!

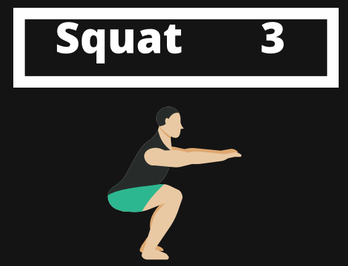
The complete system workflow



- The system recognize a pose

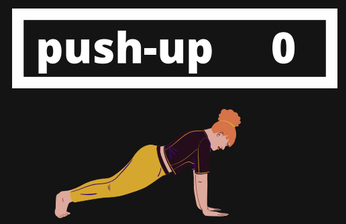


- Now can count its the repetitions and check the correctness of the exercise

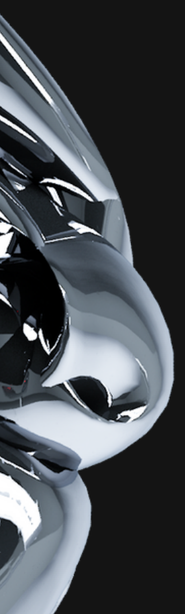


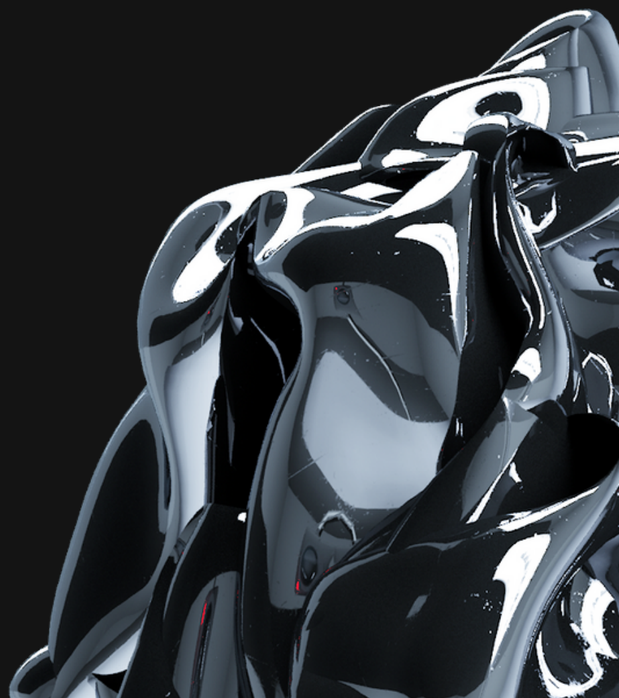
Perfect!

- If the exercise is correctly executed the App advise the user with a feedback



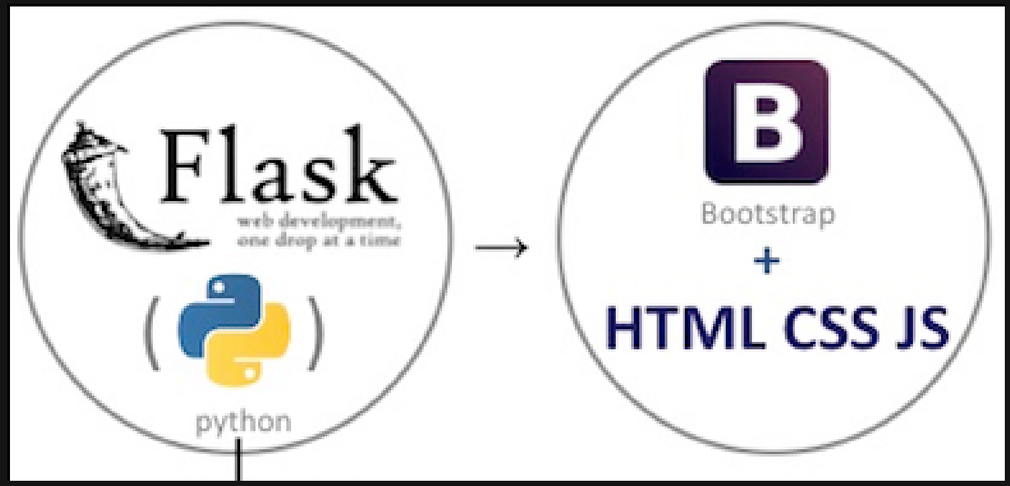
- If the system detect a different pose it reset the counter and restart to count



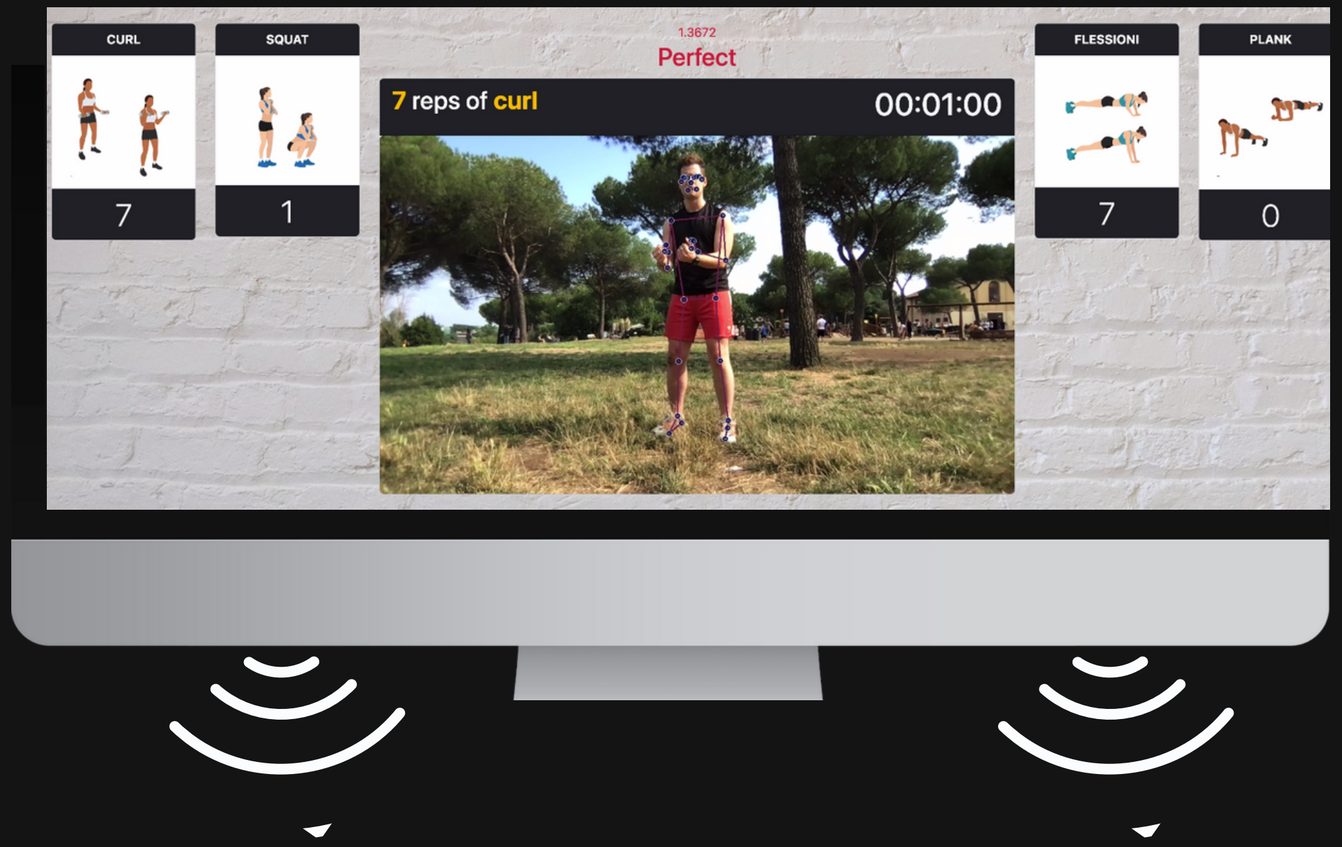


Web Application

- Flask framework (Python), HTML, and Bootstrap
- Core of the application: "app.py" file handling backend
- Homepage
- Exercise page with countdown, webcam output, exercise cards, and real-time feedback
- Final overview page



Web Application (2)

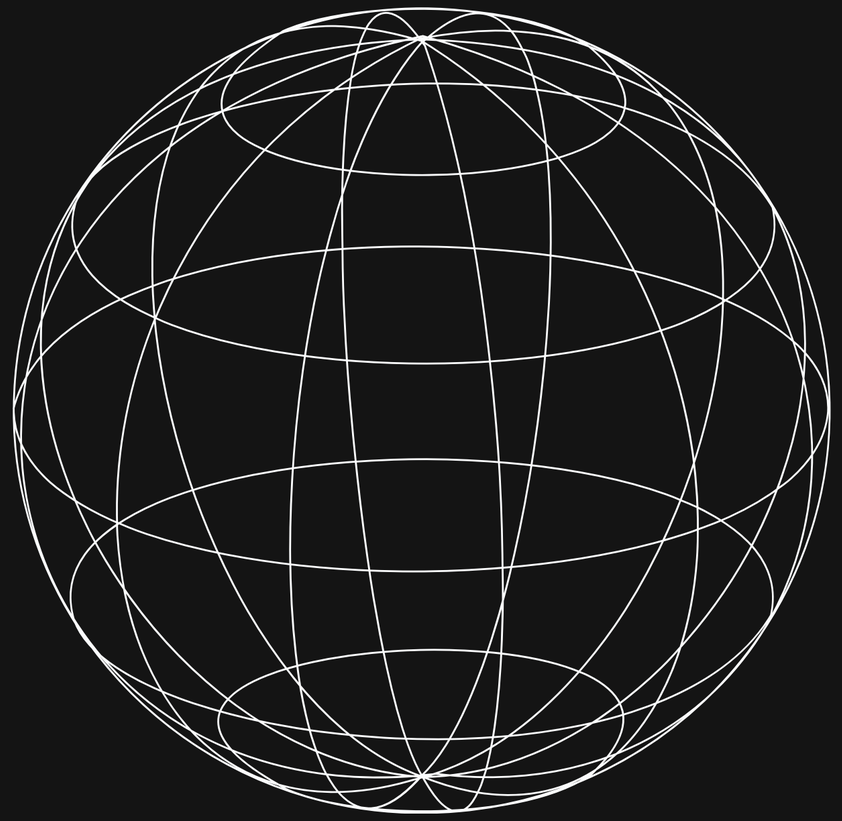


- Initial countdown to be ready
- Real-time body landmarks
  - Error alert if principals are not clearly visible (<30%)
- Classify actions in {"perfect", "good", "too quick"} based on speed
- Timer and cards for exercises

FUTURE DEVELOPMENT

Add new actions

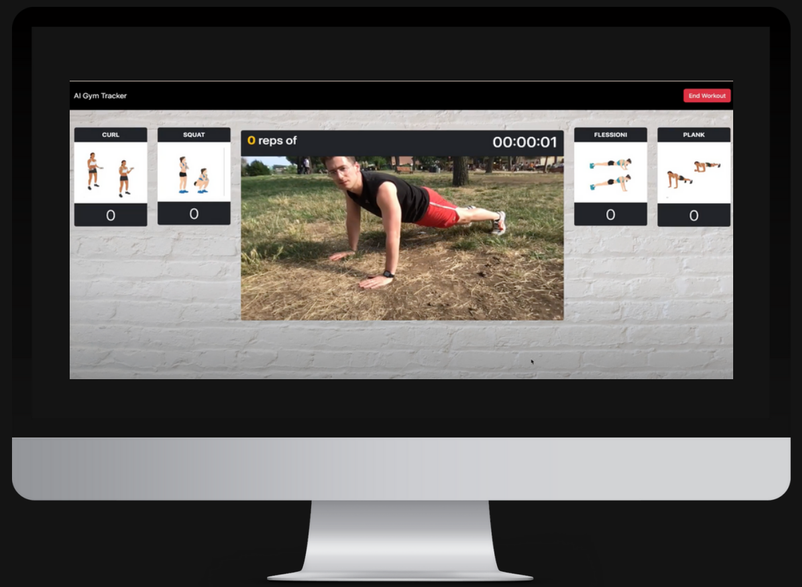
Automatic email



DB authentications

Deploy to server

LIVE DEMO



Click Here!