LOYALIST COLLEGE

Term: AIP Term (Summer 2024)

Project: Artificial Intelligence Based Mock Interview Tool For TBC College

Project Advisor: Bhavik Gandhi

Project Coordinator: Stanley Chor

Document Purpose: Project Documentation

Team Member	Email
Abhi Patel	abhirajeshbhaipat@loyalistcollege.com
Siddesh Pannkikar	siddeshsatishpani@loyalistcollege.com
Kumar Dahal	kumardahal@loyalistcollege.com
Washington Junior	washingtonquerol@loyalistcollege.com
Poojaben Katrodia	poojabendilipbhai@loyalistcollege.com
Saranya Khanna	saranyakhannaraji@loyalistcollege.com

CONTENTS

INTRODUCTION

Welcome to MockMate, the ultimate AI-based mock interview tool designed to help you ace your next interview! Our AI-based mock interview tool is designed to help users practice and refine their interview skills through a comprehensive suite of features. This tool provides real-time feedback on various aspects of the interview process, helping users improve their performance and increase their chances of success in actual interviews. Our cutting-edge features provide comprehensive feedback to prepare you for any interview scenario.

Goals & Objectives:

- Improved candidate performance in real interviews, leading to higher job placement rates.
- Greater efficiency in talent acquisition, resulting in cost savings and resource optimization.
- Positive feedback and testimonials from both candidates and hiring organizations indicate the tool's effectiveness in preparing candidates for success.

TARGET AUDIENCE:

MockMate is specifically designed for the students of TBC, with a focus on delivering personalized feedback, analyzing performance, and building confidence. Our tool includes a comprehensive question bank that spans various fields, including Global Business Management, Artificial Intelligence, Supply Chain Management, Cloud Computing, and Cybersecurity. This allows students to prepare for a wide range of job roles. Given the importance of communication skills for TBC students, MockMate also emphasizes enhancing these skills to ensure overall professional development.

PROJECT SCOPE

In scope: The AI Mock Interview Tool will be used to analyze candidates' eye-movement, tone, clarity, and confidence levels. This project covers analysis & design, development and testing of a prototype of a mock interview tool plus full documentation that serves as a proof-of-concept supporting limited functions as listed under High-level Requirements.

Constraints:

1. No formal business requirement discovery process is performed.
2. This project is only a proof-of-concept showing the capabilities of the students to the best supported by the information and data from TBC.
3. The end product from this project will not be deployable due to cost constraints and lack of complete business & functional requirements from TBC.
4. The chatbot will not support follow-up questions from the users since the dataset does not provide follow-up data.
5. This project is constrained by its timeline, functionality will be limited to necessities only at the discretion of the project advisor.

Project End Product(s) (Define the expected outcome of the project which needs to be tangible and measurable.)

1. The end product is a prototype as a Proof-of-Concept (POC) of a mock interview tool.
2. Full documentation including but not limited to technical spec, functional spec, source code and installation scripts & procedures, user instruction in form of a library set.

High-Level Requirements (Define the functional and technical requirements to be met to produce project end product.)

1. **User Interface (UI):** The mock interview tool has a user-friendly interface for candidates to start, pause, and stop mock interviews and a dashboard for displaying post-interview analysis.
2. **Natural Language Processing (NLP):** The mock interview tool shows a transcription of candidate responses with advanced NLP and emotion recognition algorithms. It provides analysis of speech content for clarity, relevance, and coherence.

3. **Post-Interview Analysis**: The mock interview tool shows a report summarizing strengths and areas for improvement and suggestions for enhancing interview performance.

FEATURES

VIDEO AND AUDIO RECORDING:

The audio and video recording feature for the mock interview tool allows candidates to simulate real interview scenarios by capturing their responses and behaviors during practice sessions. This functionality provides a comprehensive analysis by recording both audio and video, ensuring that every aspect of the candidate's performance is captured. The recorded sessions are then processed by advanced AI algorithms, which evaluate various factors such as speech clarity, tone, body language, and eye contact. This detailed feedback helps candidates identify their strengths and areas for improvement, enhancing their preparation and boosting their confidence for actual interviews.

GitHub: MockMate/web/src/app/features/interview/components/interview-recorder at dev · Qbitx6/MockMate (github.com)

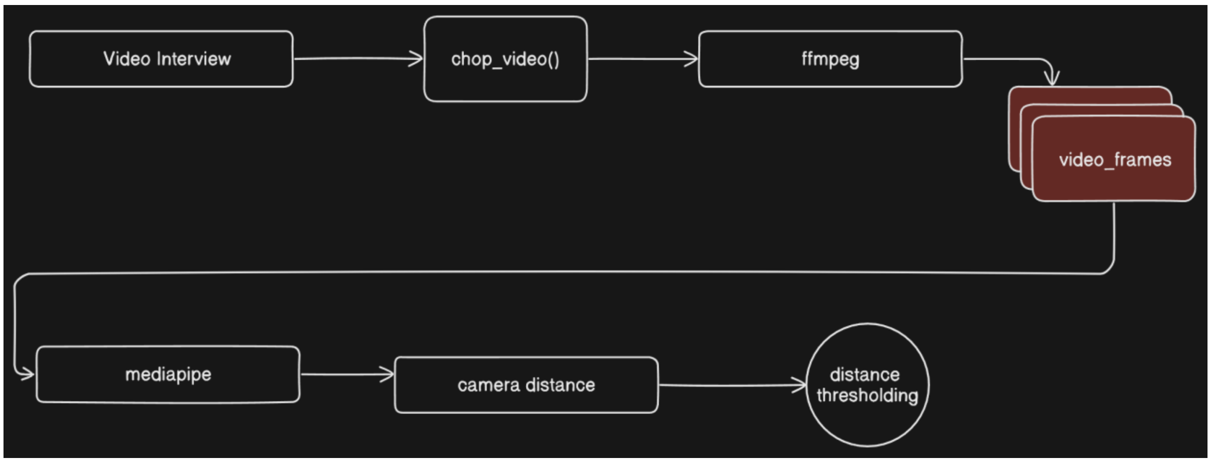
PREP TIME, RECORDING TIME:

For each question, the user gets a prep-time of 30 seconds and a recording time of 1 minute within which they can answer. Once the user has completed answering the question, they can mark the recording as done to move on to the next question. Users can only move through the questions sequentially. (Questions will be based on the role they are practicing their interview for. The mock interview tool can support 5-10 different role-based job interviews, as well as a general interview HR round of questions.) Future enhancements

GitHub: MockMate/web/src/app/features/interview/components/prep-timer at dev · Qbitx6/MockMate (github.com)

DISTANCE TO CAMERA:

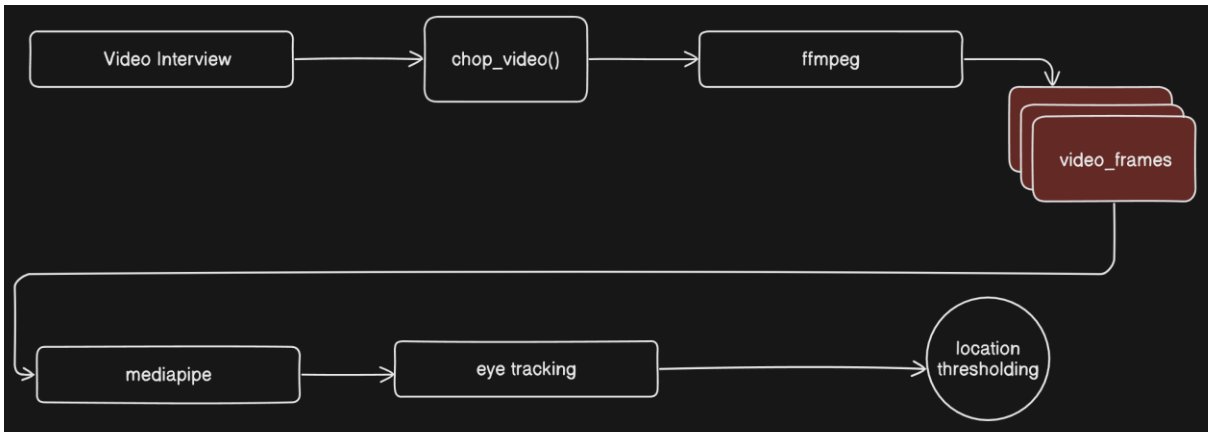
A mock interview tool incorporates facial detection and distance analysis to provide feedback on user positioning. By utilizing MediaPipe, the system identifies the user's face within the video frame and calculates their distance from the camera based on facial dimensions and pre-calibrated focal length. The tool generates a report detailing adherence to optimal camera distance and face centering guidelines, offering insights for interviewee improvement.



Github link: [MockMate/src/core/camera_distance/README.md at dev · Qbitx6/MockMate (github.com)]()

EYE CONTACT:

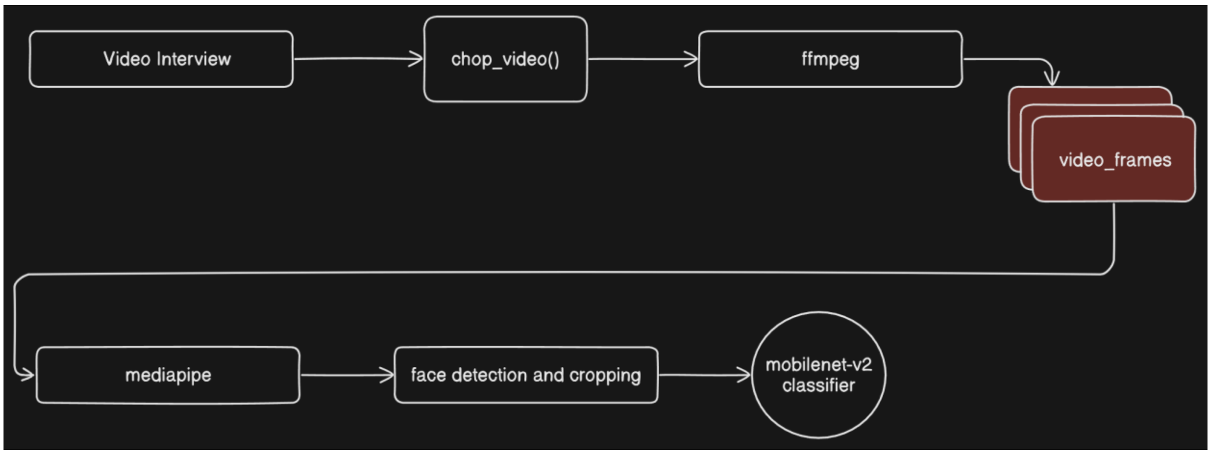
The mock interview tool assesses a user's eye contact duration and provides corrective feedback by analyzing facial landmarks using Google's MediaPipe Facemesh model. The system calculates gaze direction by measuring the relative position of the iris within the eye and determines if the user is looking left, right, or directly at the camera. By tracking eye contact patterns throughout the interview, the tool offers insights on the importance of maintaining eye contact in virtual settings and provides recommendations for improvement, addressing a critical gap in existing virtual interview platforms.



Github link:
https://github.com/Qbitx6/MockMate/blob/dev/src/core/eye_tracking/README.md

FACIAL EXPRESSIONS:

The Facial Expression module is designed to analyze and identify the user's facial expressions within video frames. It provides real-time classification of various facial expressions, helping to gauge the user's emotions and reactions during interactions.



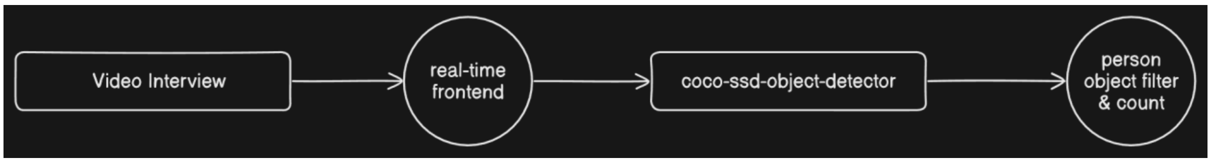
Github link:
https://github.com/Qbitx6/MockMate/blob/dev/src/core/facial_expression/README.md

VIDEO BACKGROUND:

The primary goal of this feature is to identify the presence of individuals other than the primary interviewee during a mock interview**.** This is crucial as it simulates a real-world interview environment where distractions and interruptions can negatively impact performance.

By detecting additional persons, the system can provide real-time feedback to the user about the importance of an undisturbed interview space. This information can be invaluable for improving interview performance and building confidence.

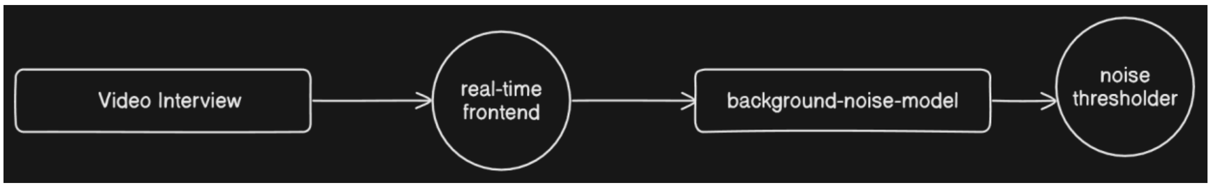
Implementation Details



Github link: MockMate/web/src/app/features/interview/components/interview-recorder/object-detection-worker.ts at dev · Qbitx6/MockMate (github.com)

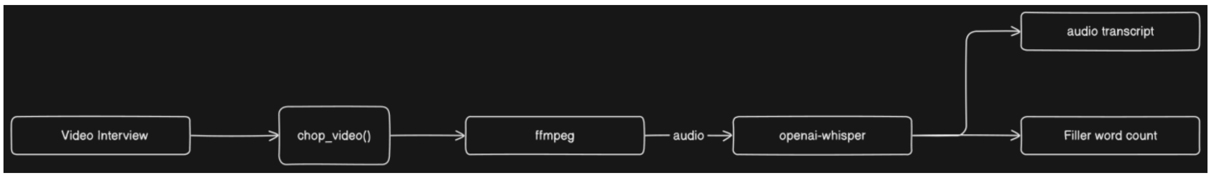
AUDIO BACKGROUND:

Minimizing background noise is critical for effective communication in virtual interviews. Excessive noise can hinder the interviewee's ability to articulate their thoughts clearly and can also be distracting for the interviewer. By detecting and quantifying background noise, the system can provide valuable feedback to the user about the importance of a quiet interview environment.



Github link:
https://github.com/Qbitx6/MockMate/blob/7d9effbd47d54f023d3252481354b84bfab574d8/web/src/app/features/interview/interview.component.ts#L111

AUDIO TRANSCRIPTION AND FILLER WORD DETECTION



AUDIO EXTRACTION

The first stage is to isolate the audio component from the video file. We employed a multimedia processing tool, FFmpeg, to extract the audio stream. This tool effectively separates the audio data from the video, providing a clean audio file for subsequent processing.

Github:
https://github.com/Qbitx6/MockMate/blob/6c506036008e4110f1a14d86f6e0f10285f5a256/src/utils/utils.py#L8

SPEECH-TO-TEXT CONVERSION

The extracted audio is then fed into a speech recognition model to generate textual transcripts. We evaluated multiple models before selecting Google Speech-to-Text API as our final choice.

Model Selection

- **OpenAI Whisper:** This model offered a range of options from lightweight to highly accurate, allowing for flexibility based on specific requirements. However, it was ultimately not selected due to the superior performance of Google Speech-to-Text in our testing.
- **IBM Watson:** While accurate, IBM Watson is a paid service, making it incompatible with our open-source project.
- **Google Speech-to-Text API:** This model demonstrated exceptional accuracy, particularly in recognizing names, and was also efficient in terms of processing time. These factors made it the preferred choice for our application.

Github link: MockMate/src/core/video_transcription at dev · Qbitx6/MockMate (github.com)

FILLER WORDS DETECTION:

The goal of this project is to identify and quantify filler words (e.g., "um," "uh") in speech to help users improve their communication skills**.** To achieve this, the system extracts audio from video files and converts it to text using speech-to-text models. Several models were explored, including Google API, Kaldi, IBM Watson, and a custom CNN model. Ultimately, OpenAI Whisper was chosen for its ability to preserve filler words in the transcription. By providing a specific prompt, the model is instructed to maintain these words, allowing for accurate detection and counting.

Github link- https://github.com/Qbitx6/MockMate/tree/dev/src/core/video_transcription

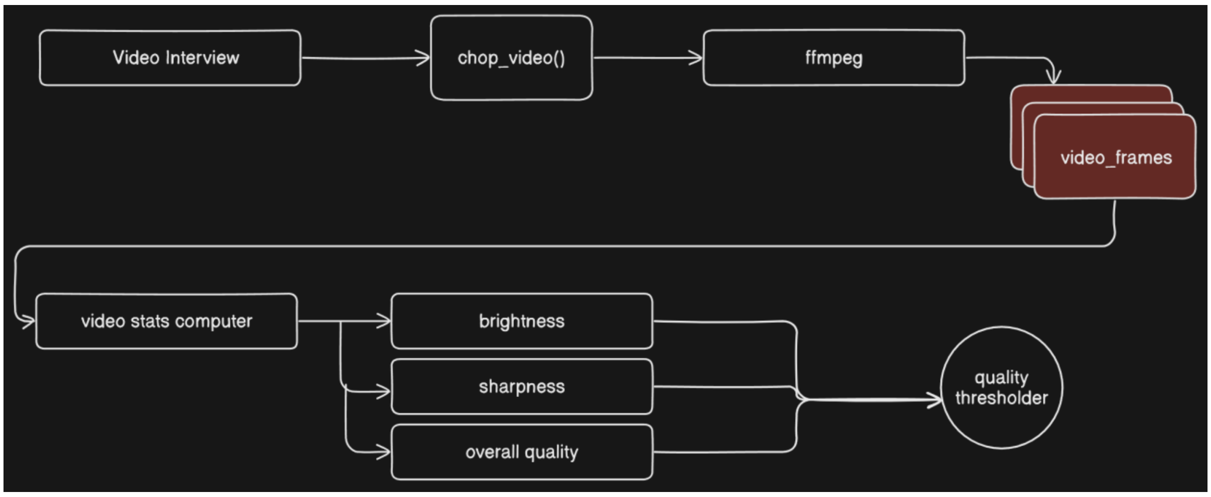
ENGLISH PROBABILITY:

To ensure the accuracy of non-verbal cue analysis, the mock interview tool includes an English language detection feature. By utilizing OpenAI Whisper to calculate the probability of the detected language being English, the tool can determine the reliability of the generated report. If the detected language is predominantly English, the non-verbal analysis is considered valid**.**

GitHub link:
https://github.com/Qbitx6/MockMate/blob/66c86e066681e7e436f401d931f3cfa7a09cf556/src/core/video_transcription/main.py#L31

VIDEO QUALITY:

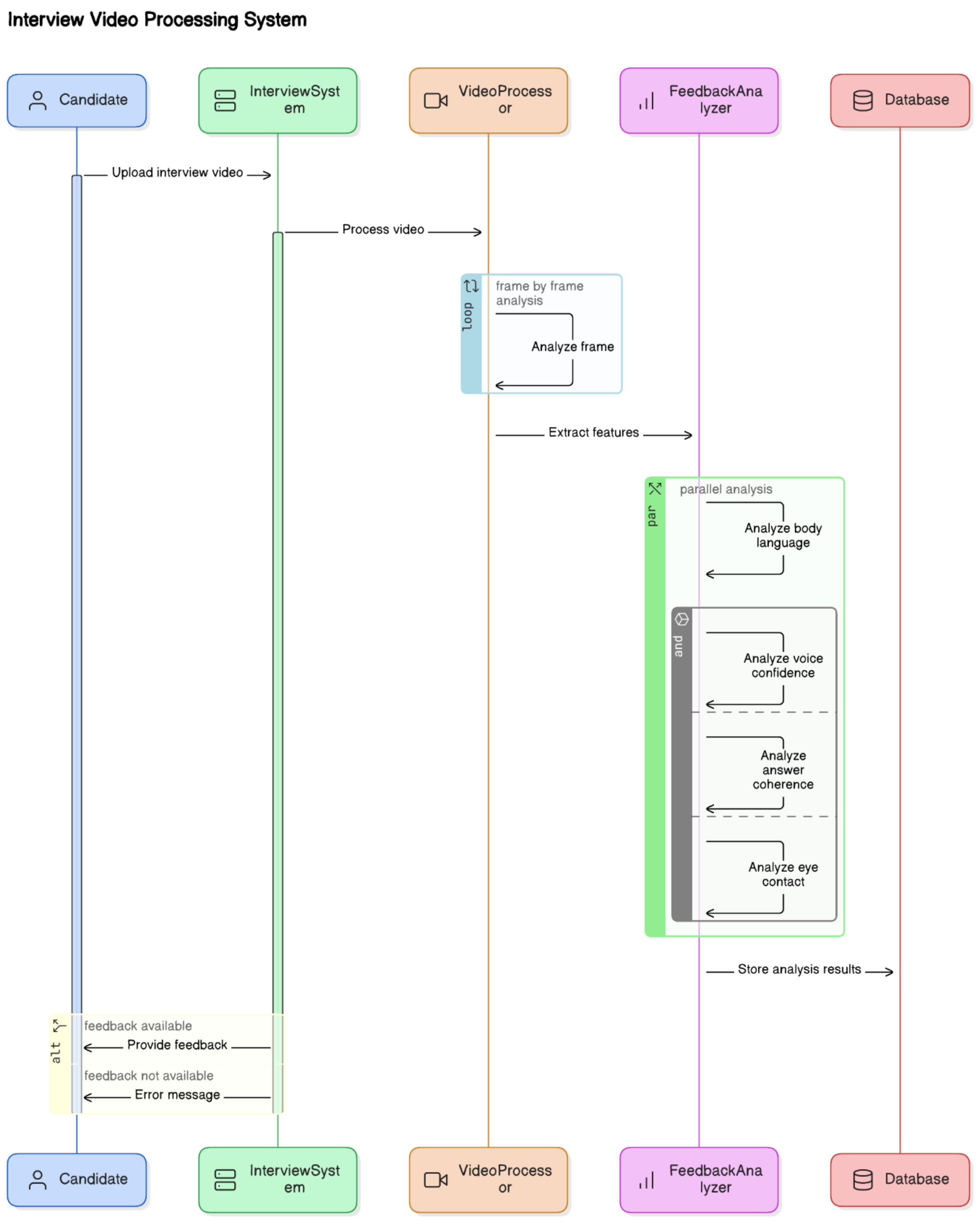
The module evaluates video quality based on sharpness and brightness. It processes video frames, calculates these metrics, and classifies video quality as poor, good, or high. This information is valuable for identifying suboptimal video conditions and enhancing user experience in applications like video conferencing or interviews.



Github link- MockMate/src/core/video_quality/README.md at dev · Qbitx6/MockMate (github.com)

FEEDBACK INTEGRATION

The feedback system provides a comprehensive evaluation of video presentations by integrating multiple machine learning models. The system processes video content to extract visual and auditory cues, which are then analyzed to provide actionable feedback.



CORE COMPONENTS:

- **Video Preprocessing:** The input video is segmented into individual frames for image-based analysis. Audio is extracted for speech-to-text conversion.
- **Audio Analysis:** Speech-to-text conversion is performed to obtain a textual representation of the spoken content. This text is used for sentiment analysis, keyword extraction, and question-answer evaluation.
- **Facial Expression Recognition:** Identifies and categorizes facial expressions to assess emotional responses and engagement.
- **Gaze Tracking:** Analyzes eye movements to determine focus, engagement, and potential areas of interest.
- **Posture Analysis:** Evaluates body language and posture to assess confidence, professionalism, and overall presentation style.
- **Video Quality Assessment:** Assesses video clarity, lighting, and stability for technical feedback.
- **Camera Distance Estimation:** Determines the distance between the presenter and the camera to ensure optimal framing.
- **Feedback Generation:** Combines insights from all analysis components to create a structured feedback report. This report includes recommendations for improvement based on identified strengths and weaknesses in the presentation.

Github link- https://github.com/Qbitx6/MockMate/blob/dev/src/core/README.md

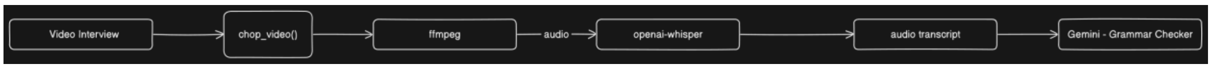
GRAMMAR CHECK:

The grammar check module aims to enhance written communication by identifying and correcting spelling and grammatical errors. Initially, the module relied on a combination of the TextBlob library for spell checking and the HappyTextToText library (based on the T5 model) for grammar correction.

However, to improve accuracy and efficiency, the system has been upgraded to leverage the Gemini LLM for comprehensive grammar checking. This change streamlines the process by eliminating the need for multiple models and provides a more robust solution for grammar correction. The Gemini LLM's advanced language understanding capabilities enable it to effectively identify and rectify a wider range of grammatical errors, resulting in significantly improved text quality.

Benefits of Gemini LLM:

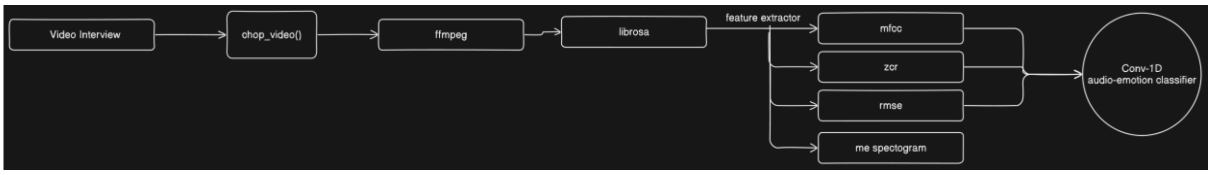
- Enhanced accuracy and efficiency.
- Eliminates the need for multiple models.
- Provides a more robust solution for grammar correction.
- Leverages advanced language understanding capabilities for better error identification and correction.



Github link- https://github.com/Qbitx6/MockMate/blob/dev/src/core/grammar-analyser/README.md

VOICE CONFIDENCE:

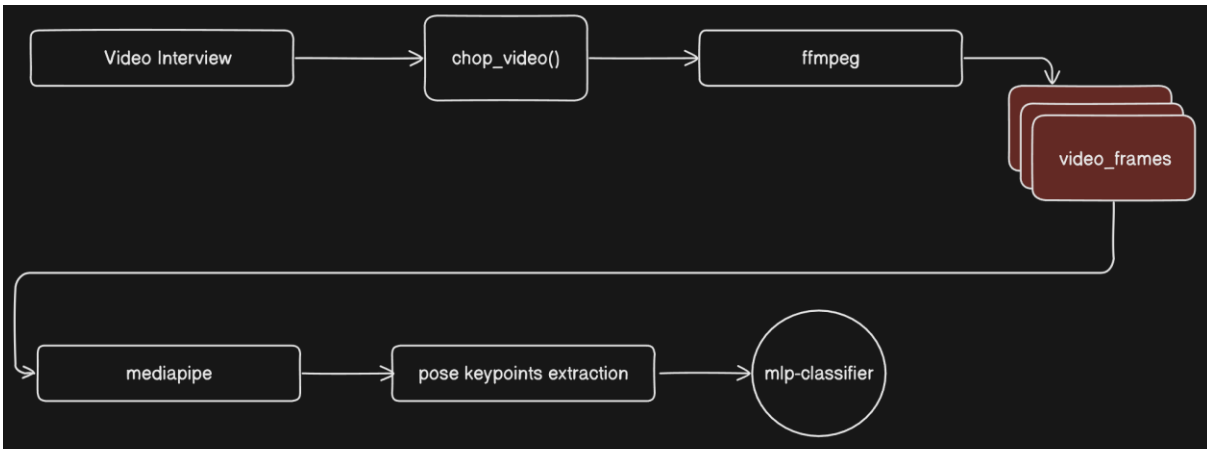
The Voice Confidence module aims to assess a speaker's confidence level by analyzing their vocal tone and characteristics. By identifying emotional cues and delivery styles, this module provides valuable insights to enhance communication effectiveness. Understanding a speaker's emotional state is particularly crucial in virtual interview settings, where traditional assessment methods are often subjective and unreliable.



Github link-
https://github.com/Qbitx6/MockMate/blob/dev/src/core/voice_confidence/README.md

POSTURE PRESENTATION :

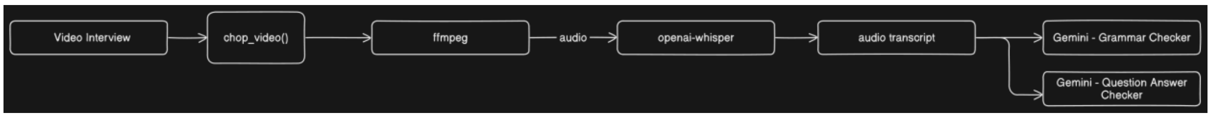
A posture detection model analyzes body posture and movements using computer vision techniques. It detects and classifies various postures to provide feedback on body language, ensuring proper alignment and presentation during activities like interviews or speeches.



Github link:
https://github.com/Qbitx6/MockMate/blob/dev/src/core/pose_presentation/README.md

QUESTION ANSWER SCORING

The mock interview tool can score the user based on the answer they provide for the technical question provided. The questions are role based technical questions such as "What is deep learning and how is it different from supervised learning?" Q&A scoring model evaluates the quality and relevance of answers in a question-and-answer session. It uses natural language processing (NLP) to analyze the content, coherence, and accuracy of responses, assigning scores based on predefined criteria.



Github link: https://github.com/Qbitx6/MockMate/blob/dev/src/core/qa_scorer/README.md

Assumptions:

1. Too much background noise will skew other results.
2. Student is not allowed to blur the background.

3. Students will be able to collect volunteer information.

Gemini key is being used which has a request limit so the system can only handle 30 interview analysis/day, under the free tier.

FUTURE ENHANCEMENTS

Speech Clarity and Coherence: The mock interview tool can analyze speech clarity and coherence and prompt the user at the end of how clear their speech was, or the speed of their speech, teaching them the significance of a steady pace and clearer speech.

BACKEND ENVIORNMENT

Install Python: Version 3.11

To get started with MockMate, follow these simple steps:

Clone the Repository:

```
git clone https://github.com/yourusername/mockmate.git
```

Checkout dev branch:

```
pip install -r requirements.txt
```

Install Requirements:

```
pip install -r requirements.txt
```

Run Flask Server:

```
flask run
```

PACKAGE BREAKDOWN

Core Functionalities

- **openai-whisper:** This package provides access to OpenAI's Whisper speech-to-text model, enabling the system to convert spoken language into written text for further analysis.
- **numpy:** A fundamental package for scientific computing in Python, used for numerical operations on arrays and matrices, essential for handling image and audio data.
- **opencv-python:** An open-source computer vision library used for image and video processing tasks, including image manipulation, feature extraction, and object detection.
- **librosa:** A Python library for music and audio analysis, used for extracting audio features such as Mel-Frequency Cepstral Coefficients (MFCCs) and rhythm analysis.
- **mediapipe:** A cross-platform framework for building real-time machine learning pipelines, used for tasks like face detection, hand tracking, and pose estimation.

- **happytransformer:** A high-level wrapper for Hugging Face's Transformers library, simplifying the process of training and using transformer models for various NLP tasks.
- **tensorflow:** An open-source platform for machine learning, used for building and training deep learning models.
- **google-generativeai:** Provides access to Google's generative AI models and services, potentially for tasks like text generation or image creation.
- **python-dotenv:** Used to load environment variables from a .env file, allowing for secure configuration management.
- **SpeechRecognition:** A Python library for performing speech recognition, providing an interface to various speech recognition engines.

POTENTIAL USE CASES

- **Speech-to-Text:** Using openai-whisper and potentially SpeechRecognition to convert audio to text.
- **Audio Analysis:** Employing librosa to extract features from audio data for tasks like speaker identification, emotion recognition, or music analysis.
- **Image and Video Processing:** Leveraging opencv-python and mediapipe for tasks like face detection, object tracking, or image processing.
- **Machine Learning:** Utilizing numpy and tensorflow for building and training machine learning models on extracted features.
- **Natural Language Processing**: Using gemini for text-based tasks like sentiment analysis, text classification, or question answering.

FRONTEND ENVIRONMENT

Angular is a robust and versatile framework that offers several advantages for building complex web applications like Mock Mate. Its component-based architecture promotes modularity, reusability, and maintainability, making it ideal for managing large-scale projects. Angular's strong typing with TypeScript enhances code reliability and reduces errors. Additionally, its built-in features for routing, dependency injection, and reactive programming streamline development and improve application performance.

Installation Guide: MockMate/web at dev · Qbitx6/MockMate (github.com)

Additional Tips:

- **Leverage Angular Material:** For a consistent and visually appealing UI, consider using Angular Material, a pre-built component library.
- **Utilize Angular CLI Schematics:** Generate components, services, and other project assets efficiently using Angular CLI schematics.
- **Implement Testing:** Write unit and end-to-end tests to ensure code quality and reliability.
- **Optimize Performance:** Optimize your Angular application for performance by using techniques like lazy loading, code splitting, and performance budgeting.

By following these steps and taking advantage of Angular's features, you can create a well-structured, maintainable, and efficient frontend for your Mock Mate application.

PACKAGE BREAKDOWN

CORE ANGULAR DEPENDENCIES

- *@angular/ packages:** These form the backbone of the Angular framework, providing essential components for building web applications. They include modules for common functionalities, forms, routing, platform-specific interactions, and server-side rendering (SSR).
- **rxjs:** Reactive Extensions Library for JavaScript, used for asynchronous programming and handling data streams efficiently.
- **tslib:** A TypeScript helper library.
- **zone.js:** A library for managing asynchronous tasks and change detection in Angular applications.

MATERIAL DESIGN AND UI

- **@angular/cdk:** Angular CDK provides a set of foundational components and utilities for building custom UI components.
- **@angular/material:** A UI component library built on top of Angular CDK, offering pre-built components for creating consistent and visually appealing user interfaces.

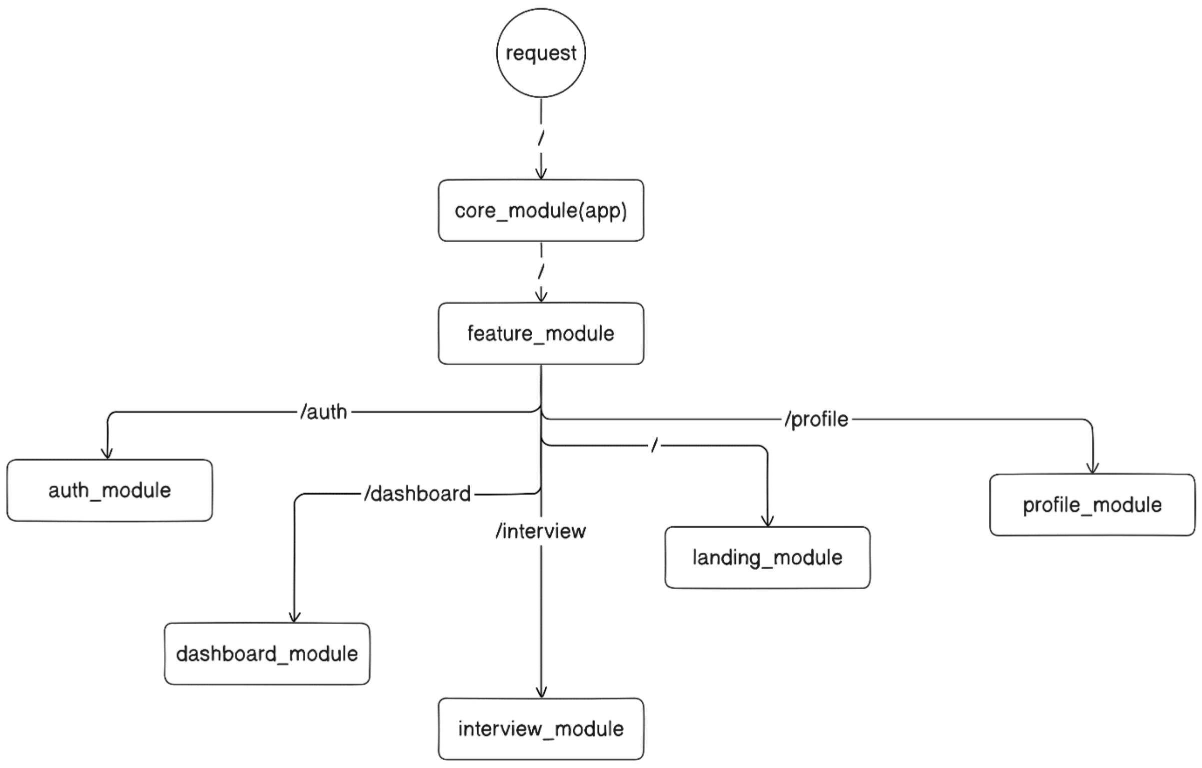
MACHINE LEARNING AND MEDIA PROCESSING

- **@tensorflow/tfjs:** TensorFlow.js is a JavaScript library for machine learning, used for numerical computations and model execution.
- **@tensorflow/tfjs-backend-cpu:** CPU backend for TensorFlow.js.
- **@tensorflow/tfjs-backend-webgl:** WebGL backend for TensorFlow.js, potentially offering performance improvements for certain operations.
- **@tensorflow-models/coco-ssd:** A pre-trained model for object detection, likely used for identifying objects within images or video frames.
- **tensorflow-models-speech-commands:** A pre-trained model for recognizing speech commands, potentially used for voice control or interaction.

SERVER-SIDE RENDERING (SSR)

- **express:** A Node.js web application framework, used for creating the server-side environment for SSR.
- **@angular/platform-server:** Angular module for rendering applications on the server.
- **@angular/ssr:** Angular SSR utilities for handling server-side rendering.

FRONTEND ARCHITECTURE



CORE STRUCTURE

- **app folder:** This is the root of your application's components, templates, and logic.
    - **app.component.html, app.component.sass, app.component.spec.ts, app.component.ts:** Define the main application component, its styles, tests, and logic.
    - **app.config.server.ts, app.config.ts:** House configuration settings for the application, differentiating between server and general configurations.
    - **app.routes.ts:** Establishes the primary routing for the application, serving as the entry point for navigation.
    - **shared folder:** Contains reusable components, services, or utilities shared across different parts of the application.

FEATURE MODULES

- **features folder:** Organizes the application into distinct feature modules for better maintainability and scalability.
    - **auth, dashboard, interview, landing, profile:** Each subfolder represents a specific feature with its own components, services, and routing.
    - **features-routing.module.ts:** Defines the routing configuration for the entire features module, handling navigation within the feature area.
    - **features.module.ts:** The main module for features, coordinating imports, declarations, and providers for the feature modules.

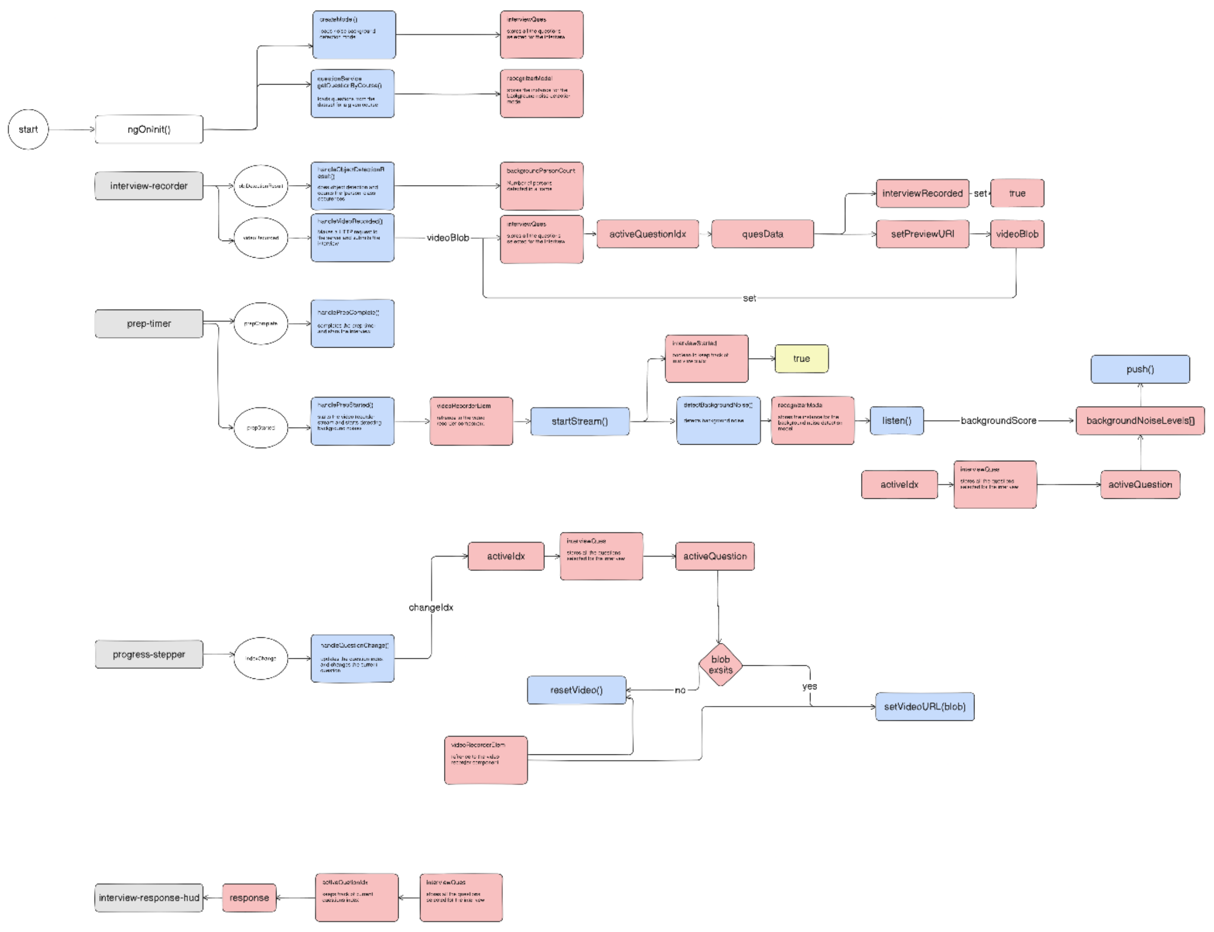
ROOT-LEVEL FILES

- **index.html:** The main HTML file serving as the application's entry point, linking to scripts and styles.
- **main.server.ts:** The server-side entry point for Angular Universal, enabling server-side rendering.
- **styles.sass:** Global stylesheet for the entire application.
- **tsconfig.worker.json:** TypeScript configuration for web workers.

KEY POINTS

- **Modularity:** The use of feature modules promotes code organization and reusability.
- **Scalability:** The structure facilitates adding new features without affecting existing code.
- **Maintainability:** Clear separation of concerns improves code readability and maintainability.
- **Performance:** The potential use of lazy loading for feature modules can enhance application performance.
- **Testability:** The structure supports unit and end-to-end testing through clear component separation and routing.

INTERVIEW FRONTEND FLOW



Detailed explanation: [MockMate/web/src/app/features/interview at dev · Qbitx6/MockMate (github.com)]()

FEEDBACK CLASS THRESHOLDING DOCUMENTATION

The Feedback class is designed to process and analyze various aspects of interview feedback data. This documentation provides a detailed explanation of how each feedback attribute is thresholded and summarized to present meaningful insights.

CLASS OVERVIEW

The Feedback class accepts an object of type IFeedback and parses it to extract relevant information. The extracted data includes:

- Audio transcript
- Camera distance
- Eye movements
- Facial expressions
- Posture

- Video quality
- Question and answer
- Voice confidence

CONSTRUCTOR

The constructor initializes the class attributes from the provided IFeedback object.

METHODS AND THRESHOLDING

AUDIO TRANSCRIPT METHODS

- **getEnglishProbability**: Returns the English probability from the audio transcript.
- **getFillerCount**: Returns the number of filler words detected.
- **getTranscriptText**: Returns the transcribed text.

CAMERA DISTANCE METHODS

- **getAvgCamDistance**: Returns the average camera distance.
- **categorizeCameraDistance**: Categorizes the camera distance as "far" if greater than 100 units, otherwise "near".

EYE MOVEMENTS METHODS

- **getEyeMovementsSummary**: Summarizes the eye movements into percentages of center, left, and right movements.
- **getEyeMovementDominance**: Determines the most dominant eye movement direction.
- **getEyeMovements**: Classifies eye movements as 'More' if dominant movement is not center, otherwise 'Normal'.

VOICE CONFIDENCE METHODS

- **getVoiceConfidenceSummary**: Summarizes the voice confidence into percentages of calm, confident, and underconfident.
- **getOverallVoiceConfidence**: Calculates the overall voice confidence as a percentage.
- **getDominantVoiceConfidence**: Determines the dominant voice confidence attribute.

FACIAL EXPRESSION METHODS

- **getFacialExpressionSummary**: Summarizes the facial expressions into percentages of bad, happy, and neutral.
- **getOverallFacialExpression**: Calculates the overall facial expression as a percentage.
- **getDominantFacialExpression**: Determines the dominant facial expression attribute.

POSTURE METHODS

- **getPostureSummary**: Summarizes the posture into percentages of positive and negative.
- **getPostureConsistency**: Calculates the posture consistency as a percentage.
- **getDominantPosture**: Determines the dominant posture attribute.

VIDEO QUALITY METHODS

- **getVideoQuality**: Returns the video quality attributes including brightness, sharpness, and overall quality.

ENGLISH QUALITY METHOD

- **getEnglishQuality**: Categorizes the English probability into quality levels:
    - 0.0 - 0.30: Poor
    - 0.31 - 0.60: Average
    - 0.61 - 0.90: Good
    - 0.91 - 1.0: Excellent

EXAMPLE USAGE

1. **Audio Transcript Analysis**
    - **English Probability**: Provides the probability of English language usage.
    - **Filler Count**: Indicates the number of filler words used in the transcript.
    - **Transcript Text**: Retrieves the transcribed text of the audio.
2. **Camera Distance Analysis**
    - **Average Camera Distance**: Measures the average distance of the camera.
    - **Categorization**: Classifies the camera distance as "far" if the distance exceeds 100 units, otherwise as "near".
3. **Eye Movements Analysis**
    - **Summary**: Calculates the percentages of eye movements towards the center, left, and right.
    - **Dominance**: Identifies the most dominant direction of eye movement.
    - **Classification**: Classifies eye movements as 'More' if the dominant movement is not center, otherwise 'Normal'.
4. **Voice Confidence Analysis**
    - **Summary**: Summarizes the voice confidence into percentages of calm, confident, and underconfident.

- **Overall Confidence**: Provides the overall confidence percentage.
- **Dominant Confidence**: Determines the most dominant confidence attribute.

5. **Facial Expression Analysis**
    - **Summary**: Summarizes facial expressions into percentages of bad, happy, and neutral.
    - **Overall Expression**: Calculates the overall expression percentage.
    - **Dominant Expression**: Determines the most dominant facial expression attribute.
6. **Posture Analysis**
    - **Summary**: Summarizes posture into percentages of positive and negative.
    - **Consistency**: Calculates posture consistency as a percentage.
    - **Dominant Posture**: Identifies the dominant posture attribute.
7. **Video Quality Analysis**
    - **Quality Attributes**: Provides video quality attributes including brightness, sharpness, and overall quality.
8. **English Quality Analysis**
    - **Categorization**: Categorizes English quality based on the English probability.

CONTRIBUTING

Use Issue Tracker Id in your branch name and Create your Branch From Master/Prod

BRANCH PREFIXES

- Feature: feature/JIRA_TICKET_ID/feature_name
- Bugfix: bugfix/JIRA_TICKET_ID/bugfix_title

EXAMPLE

- *git checkout -b feature/PAN-19/body-pose-estimator*

HOW TO WRITE YOUR COMMIT MESSAGES

- Start with your issue Type: feat/bug/chore
- Put your issue Id in the parenthesis and then write your commit message

COMMIT PREFIXES

- feat - When committing a change on a feature
- bug - When committing a change on a bug
- chore - When committing on a code that doesn't change the overall working of the tool but optimizes or fixes internal code structure

EXAMPLE

- git commit -m "feat(KAN-19): implemented a body pose estimator model"