CHAPTER 4

DESIGN SPECIFICATIONS

Design Specifications form the foundation of any successful project by detailing the essential requirements, features, and standards necessary to achieve the intended objectives. This section provides a clear and structured outline of the technical, functional, and aesthetic aspects that the project must adhere to. It serves as a critical reference point for developers, designers, and stakeholders, ensuring alignment and consistency throughout the development process.

By defining specific parameters such as system architecture, interface design, performance criteria, and constraints, design specifications minimize ambiguities and reduce the risk of errors. They act as a blueprint for efficient execution, guiding the team to deliver a product that meets both user needs and business objectives. Moreover, these specifications provide measurable benchmarks for evaluating progress and performance, ensuring that the final output aligns with the envisioned goals. A well-crafted design specification enhances collaboration, streamlines communication, and drives the project toward success.

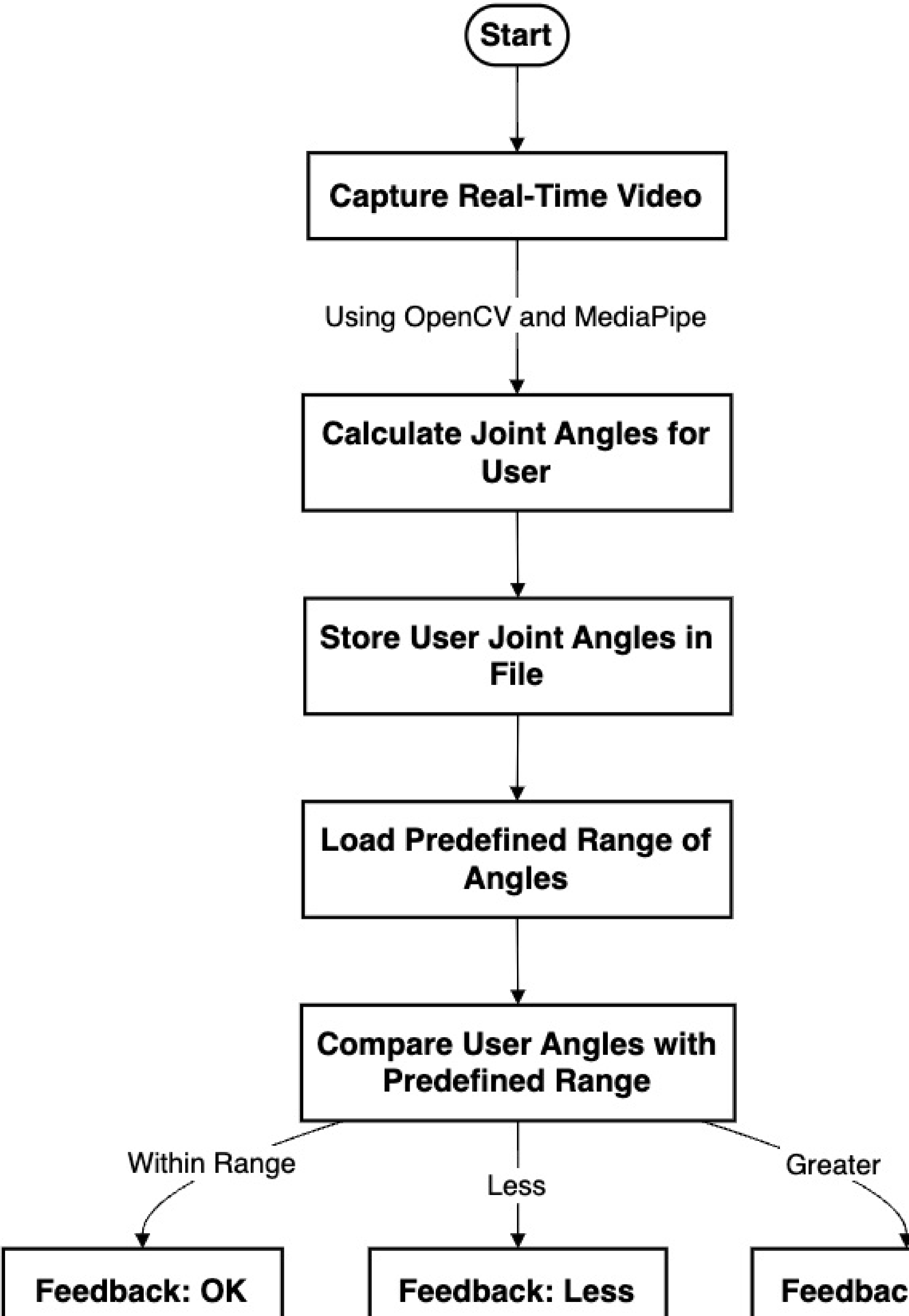
4.1 SYSTEM BEHAVIORAL DIAGRAM

The System Behavioral Diagram focuses on representing the dynamic aspects of a system, illustrating its behavior under various scenarios and interactions. This section includes diagrams such as the Activity Diagram, Use Case Diagram, and State Diagram, which provide a detailed view of the system's processes, user interactions, and state transitions. These diagrams help in understanding the flow of operations, the roles of different users, and how the system responds to specific events or inputs. By analyzing the system's behavior, this section ensures a thorough understanding of its functionality, enabling efficient development and implementation.

We have chosen these diagrams because they collectively provide a complete understanding of the system's behavior and interactions. The Activity Diagram captures the sequence of processes and decision points in the system. The Use Case Diagram focuses on user interactions, highlighting the roles and functionalities required to meet user needs effectively. The State Diagram showcases the system's dynamic transitions, detailing its responses to various events or conditions. Together, these diagrams complement each other by addressing workflows, user requirements, and behavioral patterns, ensuring clear communication, streamlined development, and robust system design.

4.1.1 Activity Diagram

An activity diagram represents the sequential flow of activities and decision points within a system. The below diagram illustrates the process within the proposed yoga pose correction system. It starts by capturing live video input from the user and processing it to extract joint angles. These angles are compared against predefined ideal ranges stored in the system. Based on the comparison, feedback is generated to guide the user, indicating whether the pose is correct, requires minor adjustments, or needs significant corrections (e.g., raising a specific limb or adjusting alignment). This workflow is depicted in **Fig-4.1**.


Start
Capture Real-Time Video
Using OpenCV and MediaPipe
Calculate Joint Angles for User
Store User Joint Angles in File
Load Predefined Range of Angles
Compare User Angles with Predefined Range
Within Range
Less
Greater
Feedback: OK
Feedback: Less
Feedbac

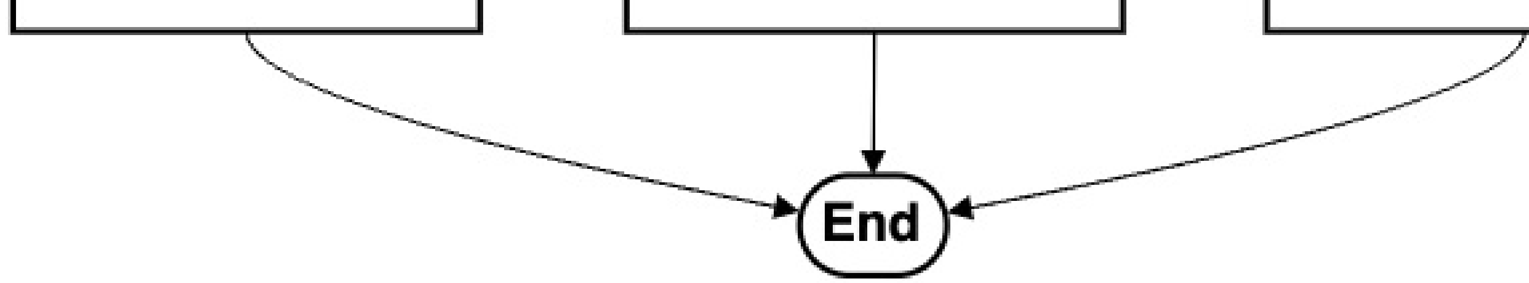
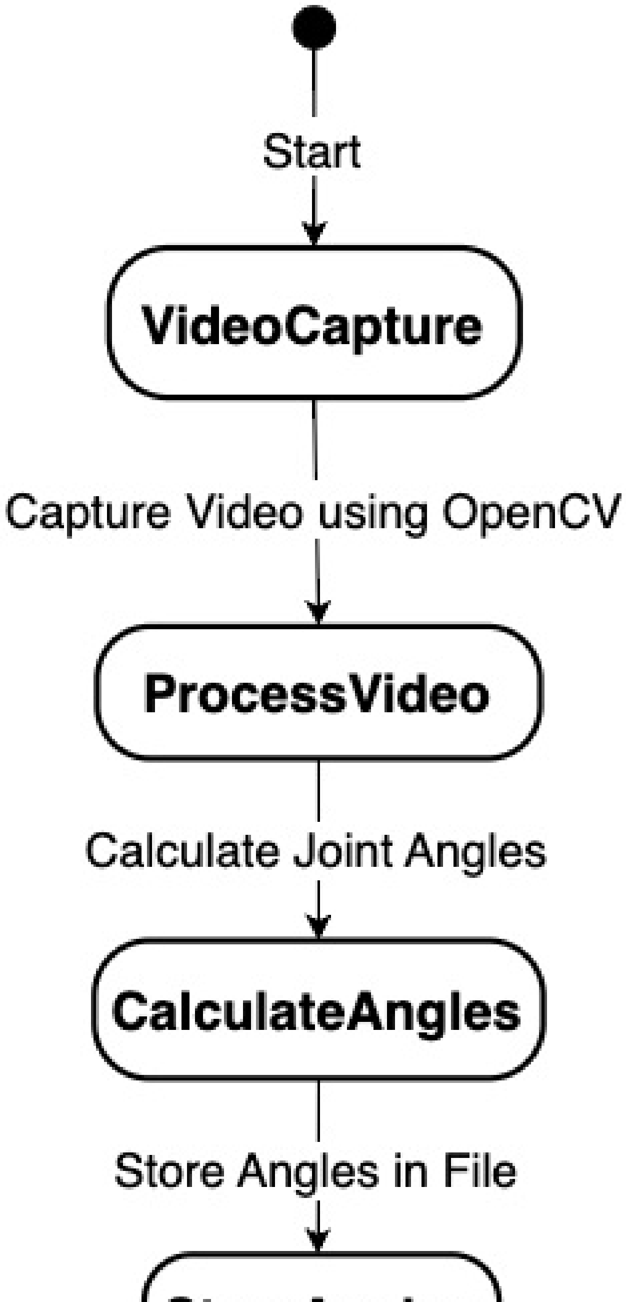


Figure 4.1: Activity diagram depicting the sequential flow of operations in the proposed yoga pose correction system, from video capture to feedback generation based on joint angle analysis.

4.1.2. State Diagram

A state diagram represents the various states of a system and transitions between them based on events or conditions. The below state diagram illustrates the yoga pose analysis system's states, starting from idle to capturing video and processing it. Once the joint angles are extracted, the system compares them with stored ranges. Depending on the results, the system transitions to the feedback state, delivering appropriate guidance to the user.



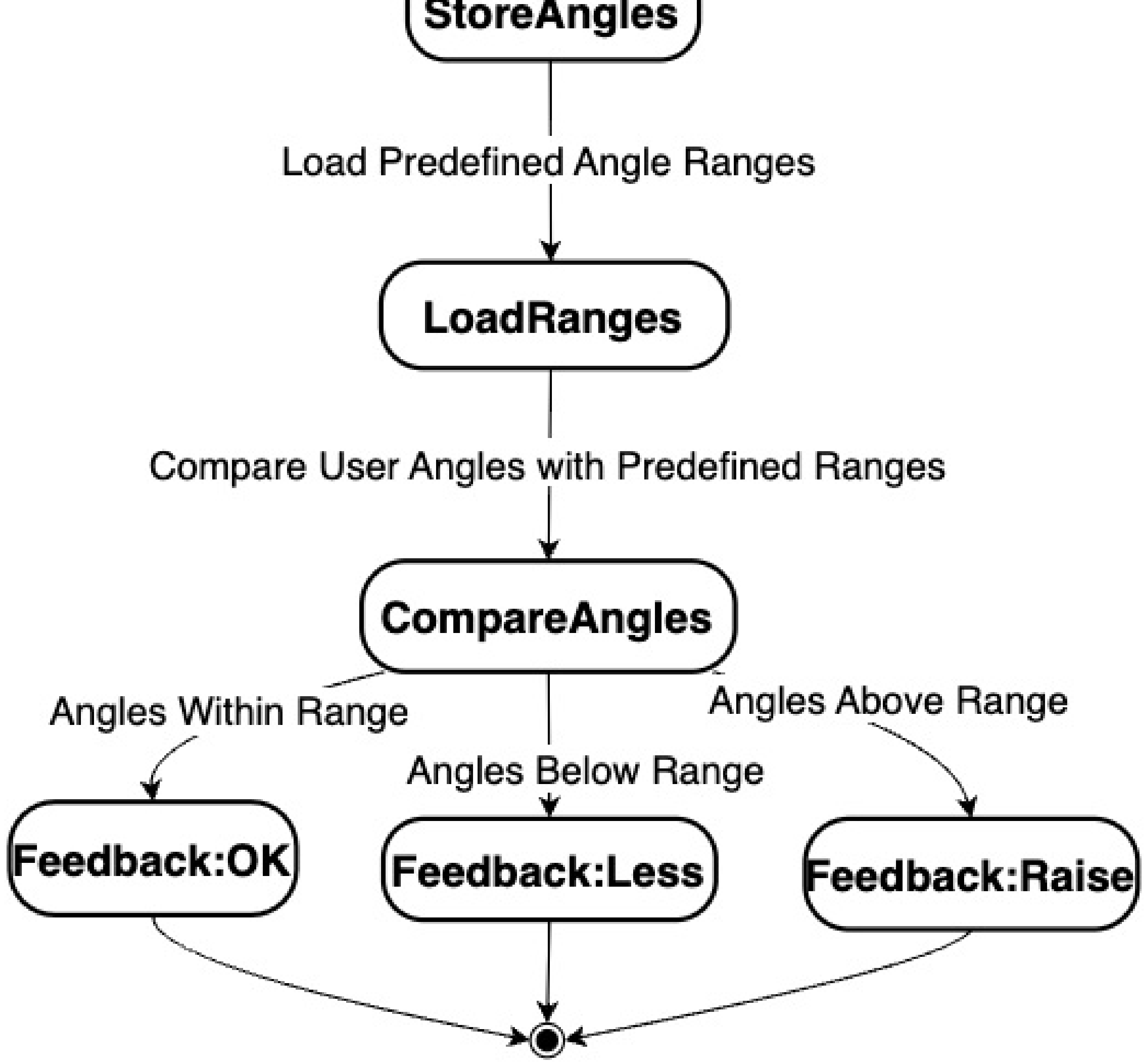


Figure 4.2: State diagram illustrating the transitions between various system states in the proposed yoga pose correction system, from video capture to feedback generation based on joint angle evaluation.

4.1.3. Use Case Diagram

A use case diagram visualizes the interactions between a user and the system, describing the functional requirements. The below use case diagram illustrates the user's involvement in the yoga pose analysis system. The user captures real-time video, which is processed by the system using OpenCV and MediaPipe. The system extracts joint angles, compares them with predefined ranges, and provides actionable feedback to the user about their pose accuracy.

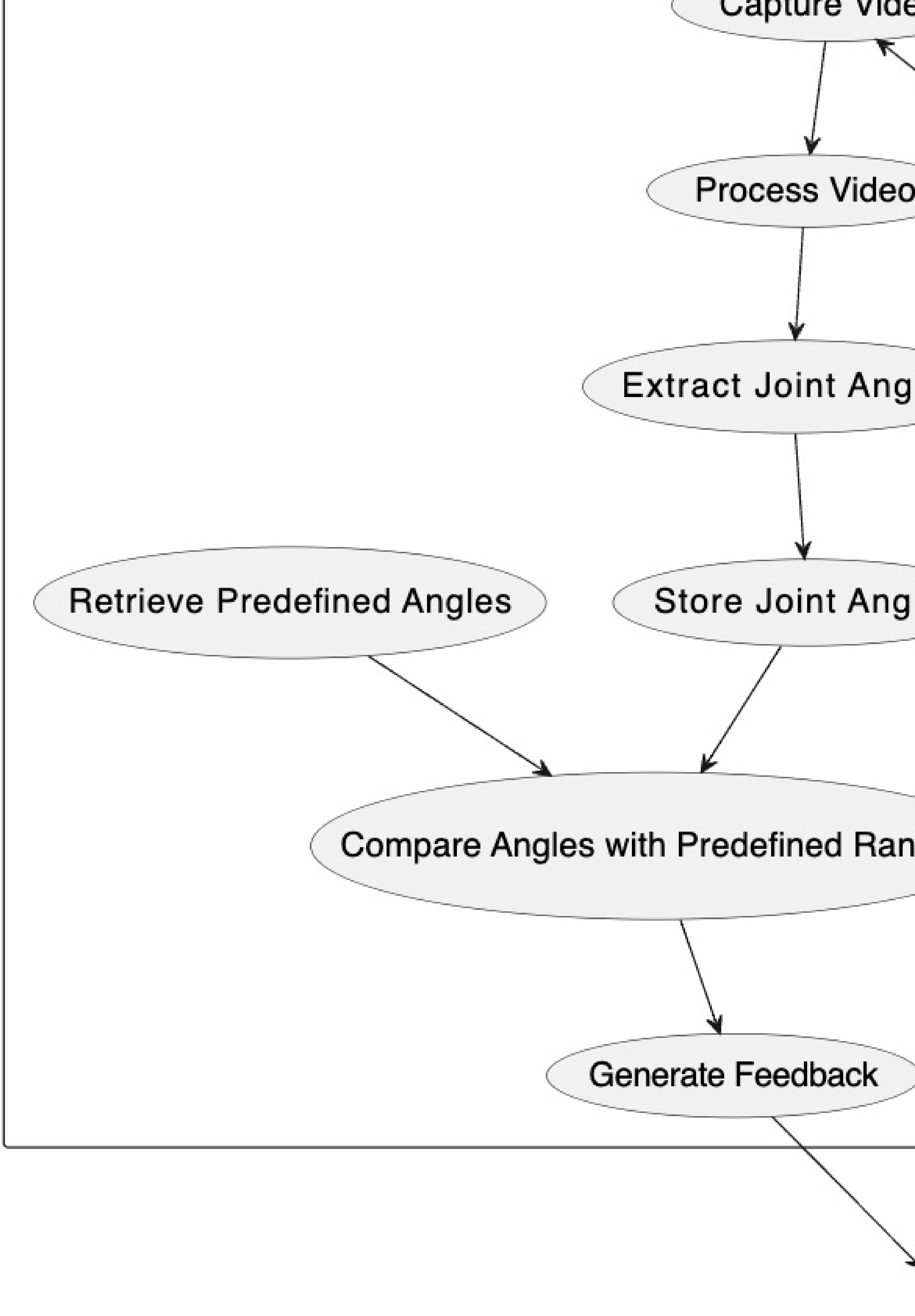

Capture Vide
Process Video
Extract Joint Ang
Retrieve Predefined Angles
Store Joint Ang
Compare Angles with Predefined Ran
Generate Feedback

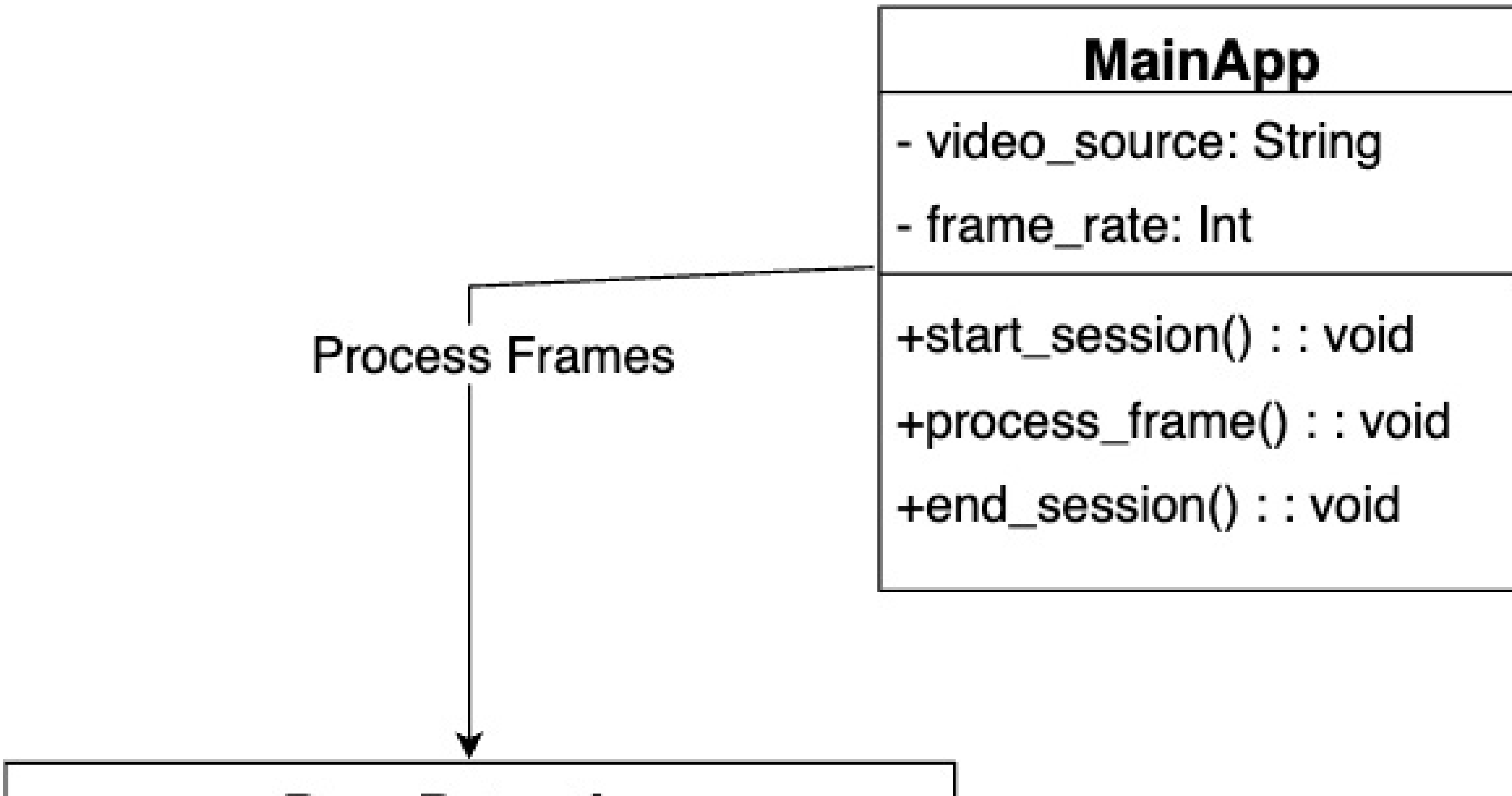
Figure 4.3: Use case diagram showcasing the user's interaction with the proposed yoga pose correction system, detailing activities such as video capture, angle processing, and feedback reception.

4.2 STRUCTURE MODEL

The Structure Model provides a comprehensive representation of the system's architecture, focusing on the organization of its components and the flow of information. This section includes essential diagrams such as the Class Diagram and Data Flow Diagram to visually depict the relationships between system entities and the movement of data across processes. By illustrating how various elements interact within the system, the structure model serves as a blueprint for developers and stakeholders to understand the system's functionality and dependencies. These visualizations ensure clarity, foster better communication, and lay the groundwork for an efficient and scalable system design.

1. Class Diagram

A class diagram depicts the static structure of a system, showing classes, attributes, methods, and relationships. The below class diagram represents the core components of the yoga pose analysis system. The PoseDetection class handles video processing and keypoint extraction. The PoseClassification class predicts and classifies poses based on the extracted keypoints. The AngleCalculation class calculates and compares joint angles, while the PoseCorrection class generates and displays feedback. The MainApp class integrates all functionalities, ensuring a seamless flow from video capture to user feedback.



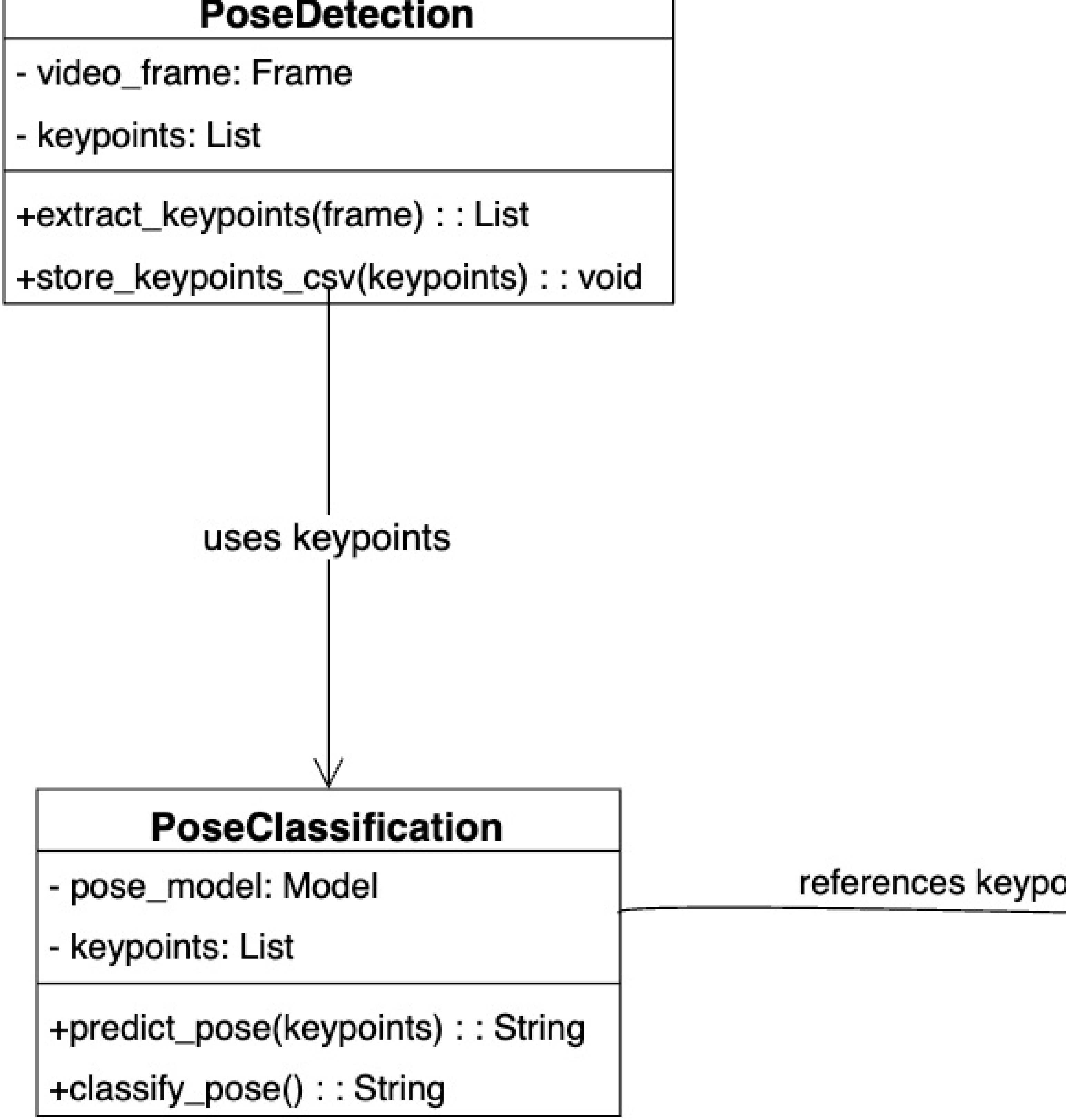
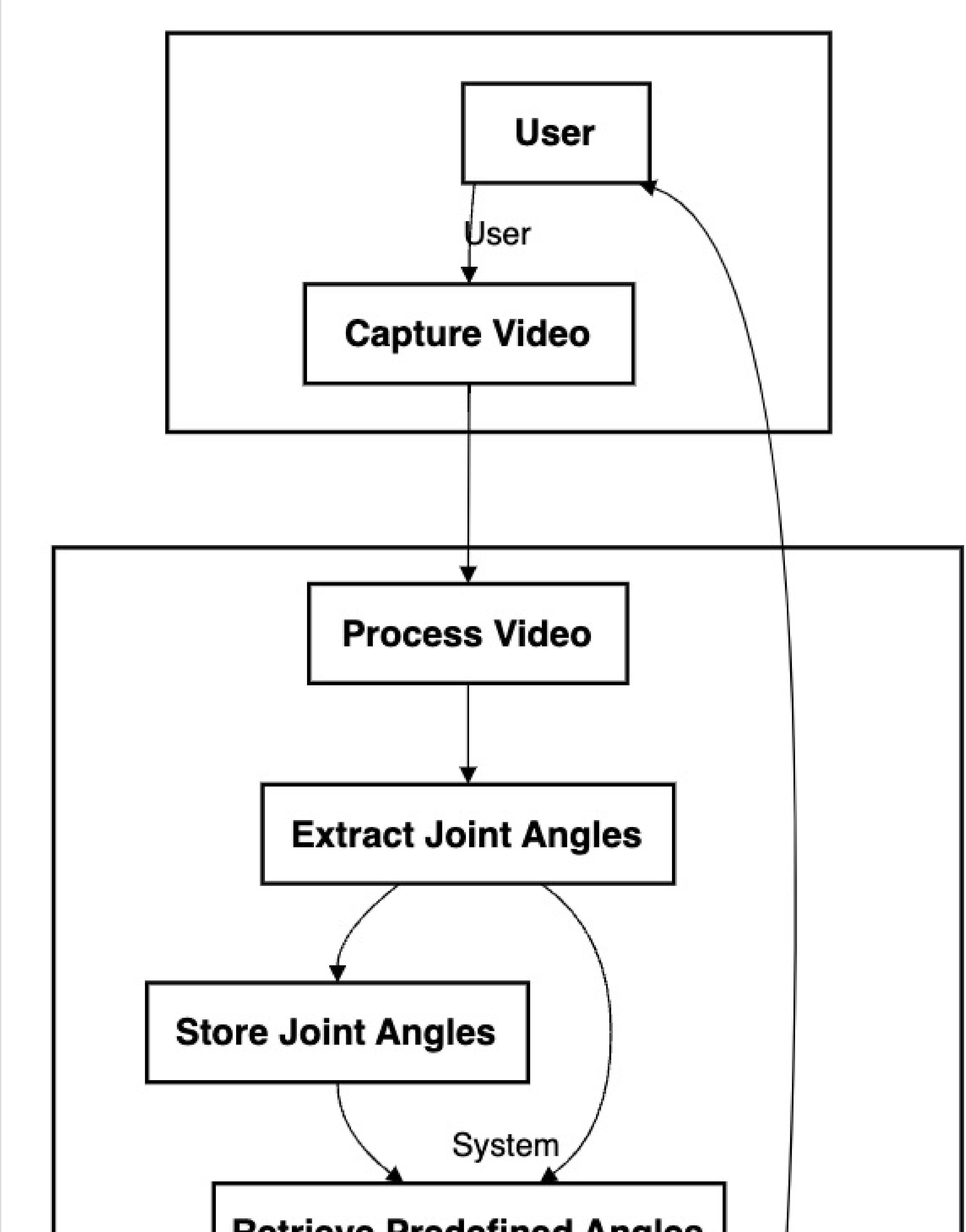

PoseDetection
- video_frame: Frame
- keypoints: List
+extract_keypoints(frame) : : List
+store_keypoints_csv(keypoints) : : void
uses keypoints
PoseClassification
- pose_model: Model
- keypoints: List
+predict_pose(keypoints) : : String
+classify_pose() : : String
references keypo

Figure 4.4: Class diagram illustrating the structural relationships and responsibilities of various components within the proposed yoga pose correction system, focusing on pose analysis and feedback generation.

2.Data flow diagram

A data flow diagram (DFD) illustrates the flow of data through a system, highlighting inputs, processes, and outputs. The below data flow diagram showcases how the yoga pose analysis system processes data. It starts with the user capturing video, which is processed to extract joint angles. These angles are stored and retrieved for comparison with predefined ranges. The comparison results flow into the feedback generation process, ultimately guiding the user.



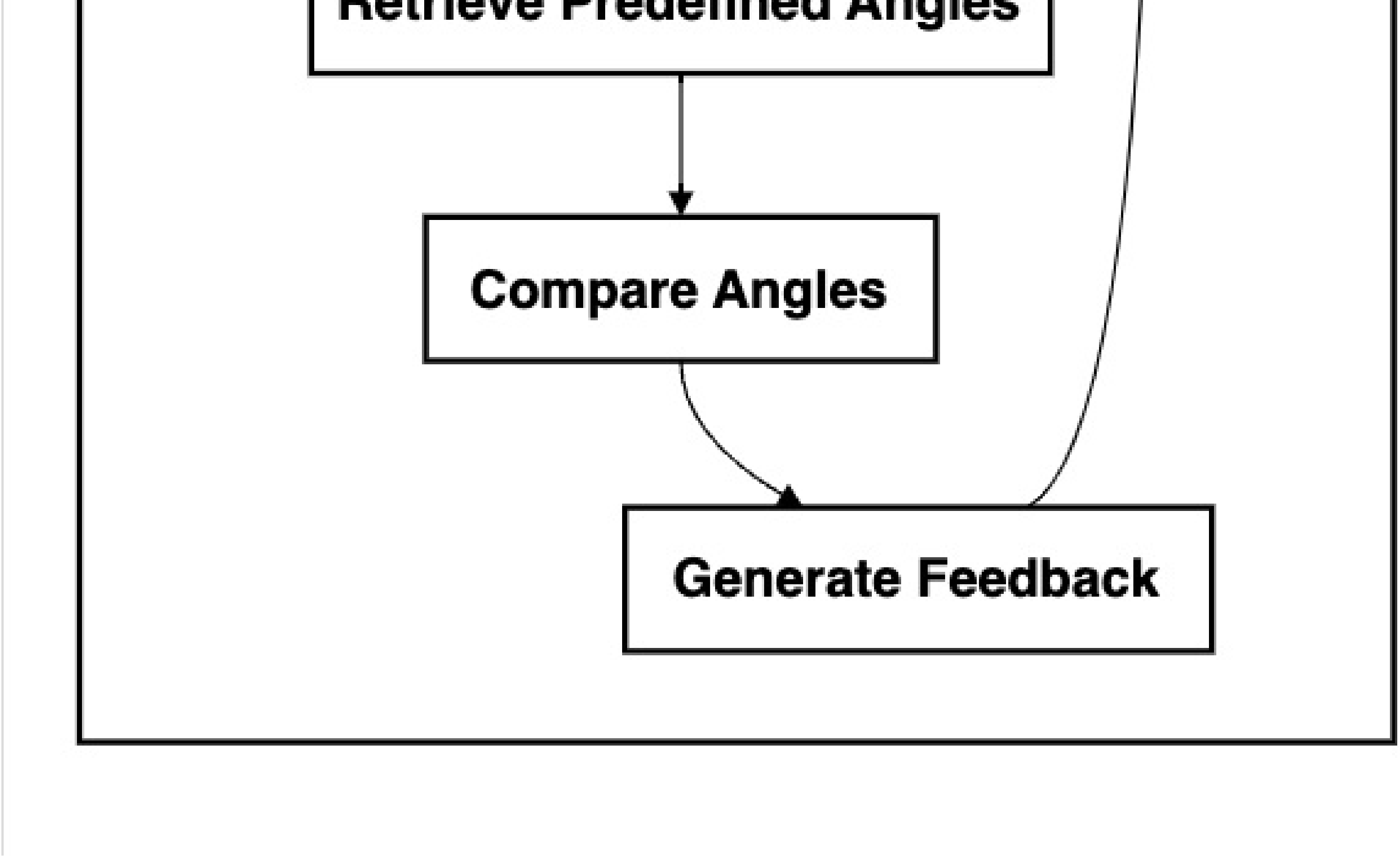


Figure 4.4: The above Data Flow Diagram (DFD) represents the flow of data within the proposed system, illustrating how inputs like user video are processed to extract joint angles, compare them against predefined thresholds, and generate corrective feedback.

4.3 Summary

The **Design Specification** section provides a detailed and structured outline of the yoga pose analysis system, combining workflow descriptions, component interactions, and architectural design. This section ensures a clear understanding of how the system operates, from capturing real-time video to processing joint angles and generating feedback for users. The **Activity Diagram** highlights the sequential steps involved in the process, while the **Use Case Diagram** focuses on the interactions between the user and the system. The **State Diagram** explains the transitions the system undergoes during various stages of execution.The **Class Diagram** highlights the static structure of the system, detailing key classes, their attributes, and the relationships between them. It provides a foundational understanding of how the system's components are organized and interact with each other. Additionally, the **Data Flow Diagram** illustrates the movement of data between various components of the system, offering valuable insights into how data is handled and processed throughout the workflow. Together, these diagrams provide a comprehensive view of the system's architecture and functionality.

Together, these diagrams form the foundation for understanding the system's functionality and ensure the design aligns with its goal of providing accurate, real-time feedback. The inclusion of multiple perspectives, from workflow to data management and structure, ensures that the system is well-optimized and easy to implement, offering a clear roadmap for its development and deployment. This holistic design approach not only enhances the system’s reliability but also provides a robust framework for future scalability and improvements.