Aim:

To develop a real-time pose estimation application using TensorFlow.js's PoseNet model in a React application.

Objective:

1. Implement a React application that captures video from a webcam.
2. Integrate TensorFlow.js and PoseNet for real-time pose detection.
3. Render detected keypoints and skeleton on the video stream.

Summary:

This project involves building a React application to perform real-time human pose estimation using the PoseNet model from TensorFlow.js. The application captures video from a webcam, processes each frame to detect human poses, and visualizes the detected keypoints and skeleton on a canvas overlay.

Tools and Libraries Used:

- **React**: For building the user interface.
- **TensorFlow.js**: For running the PoseNet model.
- **PoseNet**: A TensorFlow.js model for real-time human pose estimation.
- **Webcam**: React component for accessing the webcam.
- **Canvas**: For drawing keypoints and skeletons on the video stream.

Procedure:

1. **Initialization**:
    - Set up the React application and import necessary dependencies (tensorflow, posenet, react-webcam).
2. **Webcam and Canvas Setup**:
    - Use react-webcam to access the webcam feed.
    - Create canvas elements for drawing the pose keypoints and skeleton.
3. **Load PoseNet Model**:
    - Load the PoseNet model asynchronously using TensorFlow.js.
4. **Pose Detection**:
    - Continuously capture frames from the webcam.
    - Use the PoseNet model to estimate poses from the video frames.
    - Draw keypoints and skeleton on the canvas using the detected pose data.

5. **Drawing Functions**:
    - Define utility functions to draw keypoints, skeletons, and bounding boxes on the canvas.

Highlights:

- **Real-time Pose Estimation**: The application performs pose detection in real-time by processing video frames from the webcam.
- **Overlay Visualization**: Detected keypoints and skeletons are rendered directly on the video feed, providing immediate visual feedback.
- **Modular Code**: Utility functions for drawing and pose estimation are separated, making the code modular and easier to manage.

Conclusion:

The project successfully demonstrates the application of TensorFlow.js and PoseNet for real-time pose estimation within a React application. By capturing video from a webcam and applying the PoseNet model, the application provides real-time visualization of human poses, showcasing the potential of combining modern web technologies with machine learning models for interactive applications.