**Hand Ring POC Documentation**

**Overview**

The **Hand Ring POC** is a proof of concept (POC) web application designed to demonstrate a virtual ring try-on experience using modern web technologies. Leveraging the **MediaPipe Hand Landmarker** library, this application detects hand landmarks in real-time via a webcam feed and overlays selected ring images onto the user's ring finger. The project aims to provide an interactive and visually engaging way for users to preview how various ring styles might look on their hands, simulating a real-world try-on experience.

This POC is built using **HTML**, **CSS**, and **JavaScript**, with additional dependencies on the MediaPipe Tasks Vision library for hand detection and Material Components Web for UI elements. It serves as a foundation for potential e-commerce integrations, jewellery retail applications, or augmented reality (AR) enhancements.

**Features**

1. **Real-Time Hand Detection**:
   * Utilizes MediaPipe's Hand Landmarker to detect up to two hands in a video feed.
   * Tracks key landmarks on the ring finger to position the virtual ring accurately.
2. **Ring Selection**:
   * Offers a gallery of ring styles categorized into "Women's Ring Style" and "Men's Ring Style."
   * Users can select from predefined ring designs (e.g., Flower Ring, Ruby Ring, Gold Ring) to overlay on their hand.
3. **Webcam Integration**:
   * Supports enabling/disabling the webcam feed with a toggle button.
   * Allows switching between front and rear cameras on devices with multiple cameras (e.g., mobile devices).
4. **Interactive UI**:
   * Includes a responsive design with collapsible ring galleries and hover/touch interactions.
   * Features a skeleton overlay toggle to visualize hand landmark detection (optional).
5. **Smoothing and Scaling**:
   * Implements exponential moving average smoothing for stable ring positioning.
   * Adjusts ring size dynamically based on hand dimensions and screen size for a realistic fit.
6. **Cross-Device Compatibility**:
   * Optimized for desktop and mobile devices with responsive layouts and touch support.

**Technical Architecture**

**Files and Structure**

* **index.html**:
  + The main HTML file defining the structure of the application.
  + Includes a video feed section, ring galleries, and control buttons.
  + Links to external libraries (MediaPipe, Material Components Web) and local styles/scripts.
* **style.css**:
  + Defines the visual styling and layout of the application.
  + Uses a dark blue theme with light blue accents and responsive design via media queries.
  + Styles the video feed, ring options, and UI controls.
* **script.js**:
  + Contains the core logic for hand detection, ring overlay, and user interactions.
  + Integrates MediaPipe Hand Landmarker for real-time hand tracking.
  + Manages webcam access, ring selection, and canvas rendering.

**Key Technologies**

* **MediaPipe Hand Landmarker**:
  + A machine learning-based solution for detecting 21 hand landmarks per hand.
  + Configured to use GPU acceleration for improved performance.
  + Source: https://cdn.jsdelivr.net/npm/@mediapipe/tasks-vision@0.10.0
* **Material Components Web**:
  + Provides pre-styled UI components (e.g., buttons) for a consistent look and feel.
  + Source: https://unpkg.com/material-components-web@latest
* **HTML5 Canvas**:
  + Used to render the video feed, hand landmarks (optional), and virtual ring overlays.
* **WebRTC**:
  + Enables real-time webcam access via navigator.mediaDevices.getUserMedia.

**Functionality Breakdown**

1. **Initialization**:
   * The application initializes the Hand Landmarker model on page load using a pre-trained model from Google Cloud Storage.
   * Ring images are preloaded to ensure smooth rendering.
2. **Webcam Handling**:
   * The "Enable Webcam" button activates the video feed, defaulting to the front camera on mobile devices or the user-facing camera on desktops.
   * The "Switch Camera" button toggles between available cameras.
3. **Hand Detection and Ring Overlay**:
   * The Hand Landmarker processes each video frame to detect hand landmarks.
   * The ring finger's proximal and middle joints (landmarks 13 and 14) are used to calculate the ring's position and angle.
   * A smoothing algorithm (exponential moving average) stabilizes the ring's placement.
   * The selected ring image is scaled based on the palm width and drawn onto the canvas.
4. **User Interaction**:
   * Clicking or tapping a ring option updates the overlay image.
   * The skeleton checkbox toggles the visibility of hand landmark connectors and points.
   * "View Options" buttons expand/collapse the ring galleries.
5. **Responsive Design**:
   * CSS media queries adjust layouts, font sizes, and ring scales for different screen sizes (e.g., mobile, tablet, desktop).
   * Touch-specific enhancements improve usability on mobile devices.

**Setup Instructions**

**Prerequisites**

* A modern web browser (e.g., Chrome, Firefox, Edge) with WebRTC support.
* A webcam (built-in or external) for desktop or mobile devices.
* An internet connection to load external libraries during the initial run.
* Python 3.x installed (for running a local server).

**Installation**

1. **Clone the Repository**:

git clone https://github.com/saipraneeth77/Hand\_ring\_POC.git

1. **Navigate to the Project Directory**:

cd Hand\_ring\_POC

1. **Start a Local Server**: Use Python's built-in HTTP server to serve the files:

python -m http.server 8000

1. **Access the Application**: Open a browser and navigate to:

http://localhost:8000

**Accessing on Another Device**

To test the POC on a different device (e.g., a mobile phone on the same network):

1. **Expose the Local Server**: Use a tunneling service like **Pinggy**:

ssh -p 443 -R0:127.0.0.1:8000 qr@a.pinggy.io

* + Copy the provided public URL (e.g., http://xyz.pinggy.io).

1. **Visit the URL**: Open the URL on the target device’s browser to access the application.

**Usage**

1. **Launch the Application**:
   * Follow the setup instructions to open the app in a browser.
2. **Enable Webcam**:
   * Click "ENABLE WEBCAM" to start the video feed. Grant camera permissions when prompted.
3. **Select a Ring**:
   * Click "View Options" under "Women's Ring Style" or "Men’s Ring Style" to reveal ring choices.
   * Click a ring (e.g., "Ruby Ring" or "Gold Ring") to select it.
4. **Try On the Ring**:
   * Position your hand in front of the camera. The selected ring will appear on your ring finger.
   * Toggle the "Detection" checkbox to show/hide the hand skeleton.
5. **Switch Cameras (Optional)**:
   * Click "SWITCH CAMERA" to toggle between front and rear cameras (if available).
6. **Disable Webcam**:
   * Click "DISABLE WEBCAM" to stop the feed.

**Limitations and Future Enhancements**

**Current Limitations**

* **Camera Dependency**: Requires a functional webcam and browser permissions.
* **Lighting and Background** **Performance**: May experience lag on low-end devices due to real-time processing.
* **Ring Positioning**: Slight jitter in ring placement under rapid hand movement, despite smoothing.
* **Ring Variety**: Limited to a small set of predefined ring images.

**Potential Enhancements**

* **Custom Ring Upload**: Allow users to upload their own ring designs.
* **3D Rendering**: Replace 2D images with 3D ring models for a more realistic effect.
* **Size Calibration**: Add a calibration step to match ring sizes to the user’s hand.

**Conclusion**

The **Hand Ring POC** successfully demonstrates the feasibility of a virtual ring try-on experience using web-based AR technology. It combines hand tracking, real-time rendering, and an intuitive UI to create an engaging user experience. While currently a proof of concept, it lays the groundwork for further development into a fully-featured application for jewellery retail or personal use.

For questions or contributions, please refer to the GitHub repository: <https://github.com/saipraneeth77/Hand_ring_POC>.