Internship selection assignment task:

Real-time Video Streaming Application using Django and Websockets

Deadline for submission:
4 Days after receiving the task. Early submissions will get brownie points.

The top 3 submissions will receive Amazon gift vouchers up to INR 1K.

Instructions:

Please develop a real-time video streaming application using Django and Websockets that allows users to stream their webcam video to the backend. It will then be sent back to the front end and displayed on the same page. This task will test your skills in Django, Websockets, and handling real-time multimedia content. You may use any online resources or ChatGPT, but please refrain from seeking assistance from others. Upon completion, email your solution to **hirings@mpyg.in** with the subject line "Backend Developer Intern Real-time Video Streaming Task."

Requirements:

1. Create a Django app with a single page containing:
   - A button to start/stop the webcam stream.
   - A video element to display the user's webcam video.
   - A video element to display the received video from the backend.

2. Request user permissions and access their webcam using JavaScript and the WebRTC API:
   - When the start button is clicked, request access to the user's webcam.
   - Display the webcam video in the first video element on the page.

3. Set up WebSockets using Django Channels or any other suitable library:
   - Create a consumer to handle WebSocket connections.
   - Establish a WebSocket connection between the frontend and backend when the webcam stream starts.

4. Capture the webcam video frames and send them to the backend via the established WebSocket connection:
   - Convert the video frames to a suitable format (e.g., base64 encoded JPEG images).
   - Send the frames to the backend at a reasonable frame rate (e.g., 15-30 FPS).

5. Implement the backend consumer to receive and process the video frames:
   - Receive the video frames from the frontend through the WebSocket connection.
   - (Optional) Perform any desired processing on the video frames.
   - Send the processed video frames back to the frontend through the same WebSocket connection.

6. Display the received video frames in the second video element on the frontend:
   - Update the video element's source with the received frames in real time.

7. Close the WebSocket connection and stop the webcam stream when the stop button is clicked.

Please provide your solution in a zipped folder containing your Django project, along with a README file containing instructions on how to set up and run the application.

Good luck! We look forward to reviewing your submission.

For any feedback and queries, write us at hirings@mpyg.in or ask a question via the chat feature of the hiring platform.