FULL-STACK DEVELOPMENT INTERNSHIP REPORT – YouTube Clone

Name: Vashu Singh

# Introduction

This project involved enhancing a YouTube-inspired web application by adding a quality-switching video player, a download feature with Razorpay-based premium upgrade, and a VoIP feature with screen sharing and recording. The goal was to build a feature-rich media platform similar to YouTube while exploring full-stack technologies.

# Background

The web application was designed to offer a smooth user experience for video streaming, with added features such as quality selection for video playback, a download option with user restrictions and payment integration, and real-time video calls with screen sharing and recording. These enhancements are in line with modern content delivery and user personalization needs.

# Learning Objectives

- Learn to implement multi-quality video playback using HLS.js or similar libraries

- Set up and handle Razorpay payment gateway for premium access

- Integrate VoIP with screen sharing and recording using WebRTC and MediaRecorder

- Deploy a full-stack application using Vercel/Netlify and backend hosting services

# Activities and Tasks

## 1. Video Player with Quality Options

Task: Integrate a video player with options to play at multiple resolutions (320p, 480p, 720p, 1080p).

Activity: Used FFmpeg to encode videos into multiple resolutions (e.g., 320p, 480p, 720p, 1080p) to support adaptive playback. Implemented a custom video player with a quality switcher dropdown allowing users to select their preferred resolution. All encoded video files were hosted and integrated into the player. The implementation was tested for responsiveness across various devices.

## 2. Download Feature with Razorpay Integration

Task: Enable video downloading with a one-download-per-day limit. Allow unlimited downloads for premium users.

Activity: A download button was added below each video. The app tracks the daily limit and prompts the user to upgrade to premium. Razorpay test payment was integrated, and on success, the user could access unlimited downloads. Downloads are shown in a user’s profile page.

## 3. VoIP with Screen Sharing & Recording

Task: Add video calling, screen sharing of YouTube, and recording functionality.

Activity: WebRTC was used for peer-to-peer calls. Users can share screens using getDisplayMedia(). MediaRecorder API was used to capture and locally download the session. The UI includes buttons to start/end calls, record, and save the video.

# Skills and Competencies

- ReactJS, Node.js, Express

- Video streaming (HLS.js)

- Payment integration (Razorpay)

- WebRTC, MediaRecorder

- Hosting with Vercel + Render

- Responsive Design with TailwindCSS

# Challenges and Solutions

## Challenge 1: Razorpay Integration

Challenge: Razorpay not triggering success handler.

Solution: Used Razorpay test keys and validated with webhook.

## Challenge 2: Daily Download Limit Logic

Challenge: Limit download logic was bypassable.

Solution: Moved logic to backend and added date-based restriction.

## Challenge 3: Recording Not Capturing Shared Screen

Challenge: Recording was only capturing camera streams.

Solution: Used getDisplayMedia() and ensured proper MIME types.

# Outcomes and Impact

The internship project was completed successfully with all core features implemented: multi-quality video player, limited/free downloads, Razorpay premium upgrade, and a fully functioning VoIP system with screen sharing and recording. This improved the usability and scalability of the YouTube clone.

# Conclusion

The internship was a valuable learning experience in full-stack development. The tasks allowed me to dive deep into advanced browser APIs, payment gateways, and media streaming concepts, while also strengthening my frontend/backend deployment skills.