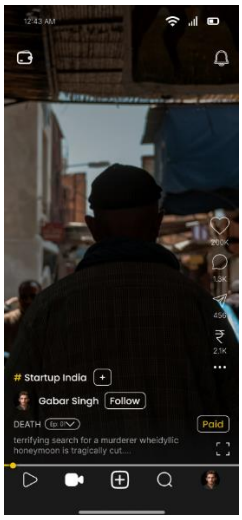
STRMLY Frontend Developer Technical Challenge
Assignment Type: Web + Mobile
Deadline: 2 Days from receiving the assignment
Submission: Via a Google Form : https://forms.gle/Ns8yqJrLTMY9Ftv98
(Check out this form once and read all the details)

Overview
Welcome to the **STRMLY Frontend Developer Technical Challenge**. Your task is to replicate a **vertical video feed layout** based on the provided screenshot/image (resembling platforms like YouTube Shorts, TikTok, or Reels). This assignment evaluates your ability to:

- Build scalable, modern UI using React or your preferred frontend tech
- Work with video playback, scroll mechanics, mock data, and interactivity
- Match visual fidelity with precision

This is a **challenge**, expected to be completed in **48 hours**. Only submit if you're confident in React.js or React Native and can execute end-to-end features independently.



Objective
Build two separate apps:

- A **web application** using **React.js** (or alternative web framework)
- A **mobile application** using **React Native (Expo)** or another mobile framework

You **must replicate the layout and interactivity exactly as shown in the provided image**, including all visual elements, placements, and behaviors.

Tech Stack Flexibility
You are free to use any frontend stack as long as:

- The **layout, styling, and interactions match the given UI reference**
- Code quality, structure, and responsiveness are maintained

Recommended stacks:

- **Web:** React.js + Tailwind CSS or CSS Modules
- **Mobile:** React Native (Expo)
- **Other Options (optional):** Next.js, Flutter, Svelte, SolidJS (if confident)

Part 1: Web Application (React.js)

Features & UI Requirements

- A vertical full-screen scrollable video feed (one video per viewport)
- HTML5 <video> tag for playback:
    - Auto-play when in view
    - Muted by default
    - Tap/click to toggle mute and play/pause
- Overlays per video (left + right UI):
    - Hashtag (e.g. #StartupIndia)
    - Creator name (e.g. Gabar Singh) + Follow button with state toggle
    - Title + Episode tag
    - Description (3-line clamp)
    - Right-aligned icons:
        - Like count (e.g. 200K)
        - Comment count (e.g. 1.3K)
        - Share (e.g. 456)
        - Tip/earnings (e.g. ₹ 2.1K)
        - Three-dot menu
- Bottom navigation bar (Home, Shorts, Add, Search, Profile — dummy only)
- Responsive layout for mobile, tablet, and desktop

Data Handling

- Use mock data (array or JSON file):
    - Fields: id, videoUrl, title, description, userName, userImage, likes, comments, shares, earnings, isPaid
- Simulate API fetch with timeout
- Show a loading screen and handle fetch failure gracefully

Part 2: Mobile Application (React Native)

Features & UI Requirements

- Replicate same vertical video feed layout for mobile
- Use react-native-video for video playback:
    - Auto-play on scroll into view
    - Tap to play/pause and mute/unmute
- Same UI overlays and elements as web
- Bottom navigation bar with dummy icons
- Tap on creator name/image opens a dummy profile page

- Use FlatList or ScrollView for feed
- Support both iOS and Android
- Use mock data from Part 1

General Requirements (Both Web and Mobile)

Code Quality

- Functional components with React Hooks only
- Use clean and scalable folder structure (/components, /screens, /services, etc.)
- Add comments and maintain readability
- Use TypeScript or PropTypes (optional but recommended)

Performance & UX

- Use React.memo, useCallback, and useMemo for optimization
- Ensure smooth transitions and animations (e.g. follow button state)
- Lazy load or optimize video rendering where needed

Responsiveness

- Web: Mobile-first responsive layout
- Mobile: Support different screen sizes, respect touch UX standards

---

Bonus Tasks (for standout candidates)

These are optional but highly encouraged:

1. **Optimistic UI Updates:**
    - Like button should instantly update UI, then simulate an API call.
    - Revert if "API" fails.
2. **Infinite Scroll / Pagination:**
    - Load more video cards when reaching bottom.
3. **Dummy Login Flow:**
    - Simple login page that stores a userID in localStorage/AsyncStorage.
    - Block video feed unless logged in.
4. **Global State Management:**
    - Use Context API or Zustand for managing global state.
5. **Unit Tests:**
    - Write simple unit tests with Jest or React Testing Library.

STRMLY Technologies Private Limited
(Building the world's largest decentralized entertainment ecosystem**)**
Website: http://strmly.com
Email: team@strmly.com
Instagram: instagram.com/strmly_
LinkedIn: linkedin.com/company/strmly