iOS Engineering Challenge: Video Feed + Inline Messaging

Overview
Create a proof-of-concept iOS application that demonstrates **two** primary capabilities:
1) a vertically scrollable, infinite short-video feed (TikTok/Reels-style), and
2) an inline **Text Input & Reactions** bar that feels native, ergonomic, and performant.

This challenge assesses your ability to design robust mobile architectures, manage resources for smooth video playback, and craft responsive UI/UX (state management, animations, and keyboard ergonomics).

Core Requirements

A) Infinite Video Feed
Functionality
- **Infinite Scrolling:** Users can continuously swipe up to see new videos and swipe down to revisit previous items.
- **Auto-Play Current:** The video that becomes current **auto-plays**.
- **Seamless Looping:** Each video loops without gaps or flashes.
- **Smooth Transitions:** Gesture-velocity tuning for natural feel. No stutters, black frames, or harsh jumps while changing items.
- **Playback Readiness:** Do **not** advance to the next item until it is ready for immediate playback.

Technical Constraints
- **Memory Efficiency:** Handle arbitrarily large catalogs without unbounded memory growth.
- **Network Efficiency:** Prefetch judiciously (HLS segments/bitrates) without downloading everything.
- **Video Format:** HLS streams.
- **iOS Native:** SwiftUI, UIKit, or a hybrid.

B) Inline Text Input & Reactions
Implement a bottom **Text Input bar** with messaging and reaction affordances as a first-class part of the experience.

Input Behavior
- Shows placeholder (“Send message”) when empty; becomes editable when tapped.
- Expands vertically as the user types (up to **5 lines**); after that, the input scrolls internally.
- While typing, **pause** video playback and **disable feed scrolling**; swipes should only scroll the text when focused.

- **Send** clears the field.
- *See provided video for desired behavior.*

Focus & Keyboard Ergonomics
- **Not Focused (default):**
  - Collapsed field shows placeholder.
  - To the right, show **reaction** buttons (❤️ and ✈️).
  - The feed is fully scrollable.
- **Focused (after tapping the field):**
  - Reaction buttons **disappear**.
  - A **Send** button (paper plane ✈️) **appears only when there is text** (hidden otherwise).
  - Feed scrolling is **disabled** while typing.
  - When the keyboard appears, the Input bar **pins above the keyboard**; the feed video remains stationary and does **not** resize.

Polish & Accessibility
- Smooth fade/scale animations for button appearance/disappearance.
- Input text in **white** with a lighter-opacity placeholder.
- Prefer VoiceOver-friendly labels and appropriate content types (e.g., `.textContentType(.none)` if appropriate), and ensure controls are reachable via accessibility focus.

Provided Resources

We will provide:
- **HLS Manifest URL**: `https://cdn.dev.airxp.app/AgentVideos-HLS-Progressive/manifest.json`
- **Video Access**: All videos in the manifest are accessible from our CDN.
- **Manifest Format:**
```json
{
  "videos": [

"https://cdn.dev.airxp.app/AgentVideos-HLS-Progressive/000298e8-08bc-4d79-adfc-459d7b18edad/master.m3u8",

"https://cdn.dev.airxp.app/AgentVideos-HLS-Progressive/0042b074-1533-47f8-b370-5713d536f09b/master.m3u8"
    // ... additional HLS stream URLs
  ]
}
```

Deliverables

1. **Working Application** (Xcode project) targeting iOS device hardware.
2. **Architecture Document** (1–2 pages) that explains:
   - Overall approach and key trade-offs.
   - Memory strategy (feed reuse, player lifecycle).
   - Network/prefetch strategy for HLS.
   - **Messaging UI architecture** (state management, keyboard handling, focus rules, animation model).
   - Transition/scrolling strategy and how you maintain responsiveness.
3. **Implementation** that:
   - Fetches the manifest and plays HLS items in an infinite feed.
   - Implements the **Inline Text Input & Reactions** core behaviors described above.

Evaluation Criteria

- **Architecture Quality:** Separation of concerns; appropriate patterns for feed, playback, and messaging state.
- **Performance:** Smooth scrolling, efficient memory usage, appropriate buffering.
- **Messaging UX Quality:** Keyboard ergonomics, focus behavior, send/clear logic, reaction affordances, and animation polish.
- **Code Quality:** Readable, maintainable, testable structure.
- **Problem Solving:** Clear rationale for decisions, thoughtful trade-offs.
- **User Experience:** Native feel, responsiveness, and resilience to slow networks.

Submission

Provide a GitHub repository or compressed archive with source code and build instructions.
Collapse