🧪 React Intern Assignment – Real-Time Gallery Interaction Task (Round 1)

Internship Type: In-office
Location: Pune – Baner
Duration: 6 Months
Opportunity: Full-time role based on performance

🎯 Objective

Build a **multi-user real-time image interaction web application**.

The app should allow users to view a gallery of images, interact with them using emojis and comments, and see **all interactions update instantly** for other users—both at the **image level** and in a **global activity feed**.

This task evaluates:

- Ability to read and apply documentation
- Real-time state reasoning
- React fundamentals
- UX & UI decision-making
- Problem-solving approach

📌 Application Structure

The application must have **two primary sections**:

1️⃣ Gallery Section

Displays images and allows real-time interactions on each image.

2️⃣ Feed Section (Real-Time)

Displays a live feed of all interactions happening across images.

🔌 Images

- Use Unsplash API to render images (https://unsplash.com/developers)

🖼️ Gallery Section – Requirements

- Render images in a **scrollable grid**
- Support pagination or infinite scroll
- Clicking an image opens a **focused image view**
- Users should be able to interact:
    - From the grid (likes / emojis)
    - From the image view (emojis & comments)

😊 Image Interactions (Real-Time – Mandatory)

Each image must support:

Emoji Reactions

- Users can add emoji reactions to an image
- Emoji updates must sync **instantly across all users**
- Emoji count or placement should feel intuitive

Comments

- Users can add comments on an image
- Comments must appear in real time for others
- Display comments in a clear, readable UI

Use **InstantDB** as the real-time data layer:
👉 https://www.instantdb.com/

🔄 Real-Time Synchronization Rules (Very Important)

Image-Level Sync

- If **multiple users are viewing the same image at the same time**:
    - Emojis added by one user must appear instantly for others
    - Comments added by one user must appear instantly for others
    - Interaction state must remain consistent across users

Feed vs Image View

- **Feed Section**
    - Shows all interactions across images (global activity stream)
    - Updates in real time
- **Image View / Gallery**
    - Shows only interactions related to that image
    - Updates in real time for users on the same image

👉 Both must update instantly and independently.

📰 Feed Section – Requirements

- Display a real-time feed of interactions such as:
    - “User reacted ❤️ to Image”
    - “New comment added on Image”
- Feed updates immediately when any interaction occurs
- (Optional) Clicking a feed item may focus the related image

---

⚙️ Technical Requirements

Mandatory Stack

- React (functional components only)
- Tailwind CSS
- Any modern component library
- **InstantDB** for real-time sync
- Modern async patterns (`async/await`)
- Deployed application (Vercel / Netlify / Cloudflare Pages)

React Concepts We Expect to See

- Clean component decomposition
- Proper separation of Gallery, Feed, and Interaction logic
- Correct usage of:
    - `useEffect` (with correct dependencies)
    - `useMemo` / `useCallback` where appropriate
    - `React Query` for api handling
    - `Zustand` for context management
- Controlled inputs for comments
- Immutable state updates
- Safe handling of async + real-time updates

🎨 UI / UX Expectations

- Clean, minimal, readable UI
- Clear separation between:
  - Gallery
  - Feed
  - Image interactions
- Smooth real-time updates (no flicker)
- Desktop-first design is fine; responsiveness is a bonus

We value **clarity and usability over visual complexity**.

⭐ Bonus (Optional – Strong Signal)

- Emoji picker instead of hardcoded emojis
- Basic user identity (random username or color)
- Ability to delete your own emoji/comment
- Subtle animation for new feed items
- Basic conflict handling (simultaneous interactions)

📦 Deliverables (Compulsory)

1️⃣ Live Deployed Application

- Publicly accessible URL
- Real-time behaviour testable using multiple tabs/devices

2️⃣ GitHub Repository

Include a **detailed README** covering:

- Setup instructions
- API handling strategy
- InstantDB schema & usage
- Key React decisions
- Challenges faced and how you solved them
- What you would improve with more time

🚫 Disallowed

- Class components
- Copy-pasted full solutions
- Ignoring error/loading states
- UI frameworks that hide core logic

🧠 Evaluation Criteria

- Documentation reading & application
- Real-time state understanding
- React fundamentals
- UX decision-making
- Code clarity & structure
- Problem-solving mindset (README matters)

📤 Submission Method

Send the following to **careers@fotoowl.ai** or anushka.alandkar@fotoowl.ai

1. Deployed app link
2. GitHub repository link

💡 Final Note

This task is intentionally open-ended.
We are not testing perfection — we are testing **how you think, learn, and build**.