System Design Document – Insyd Notification POC App

1. Introduction

The **Insyd Notification Proof of Concept (POC) System** is designed to provide a real-time notification infrastructure for Insyd, a social platform for the architecture industry. Its purpose is to allow users to receive event-based notifications (likes, comments, mentions, follows, etc.) in a scalable, reliable, and secure way.

This system is built with:

- **Backend**: Node.js + Express.js
- **Database**: SQLite (for POC, extendable to PostgreSQL/MySQL in production)
- **Frontend**: React (for displaying and managing notifications)

2. System Overview

At a high level, the system enables **users to be notified** of important events in the Insyd platform.

- **Goals**
    - Deliver real-time, reliable notifications.
    - Support multiple notification types (social, system, transactional).
    - Allow extensibility for push/email/SMS in the future.
    - Ensure notifications are idempotent and consistent.
- **Non-Goals**
    - Full production-scale message queues (POC uses in-memory queue).
    - Advanced analytics/dashboards.
    - Integration with external push/SMS/email providers (to be added later).

3. Architecture

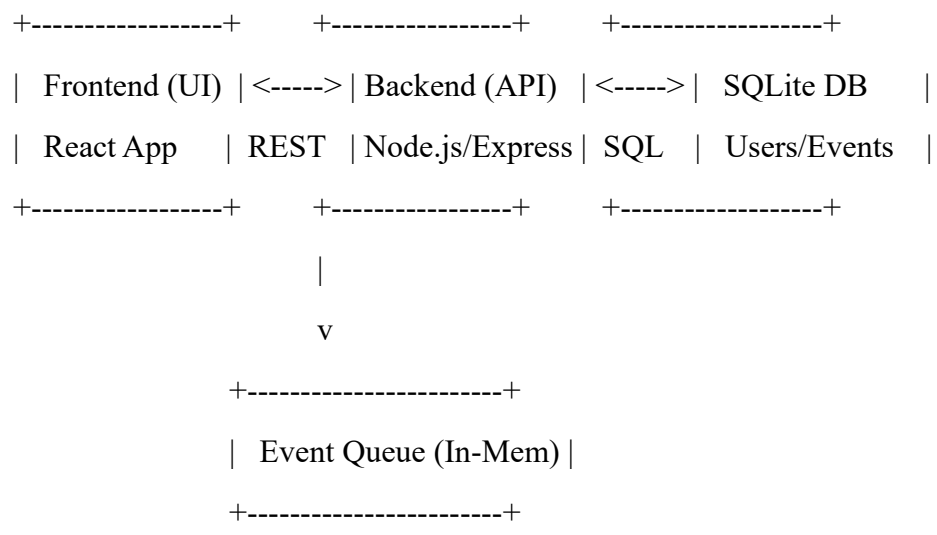
The system has **three layers**:

1. **Frontend (React)**
    - Displays notifications to users.
    - Provides APIs to fetch and mark notifications as read.
2. **Backend (Express.js)**

- Exposes REST APIs for notifications.
- Manages in-memory event queue and processes new events.
- Writes notifications to SQLite DB.

3. **Database (SQLite)**
   - Stores users, notifications, and events.
   - Provides durability and query support.

Architecture Diagram



4. Data Design

Database Schema (SQLite)

- **Users**
  - id (TEXT, PK)
  - name (TEXT)
  - email (TEXT)
- **Notifications**
  - id (TEXT, PK)
  - user_id (TEXT, FK → Users.id)
  - message (TEXT)
  - type (TEXT: "like", "comment", "follow", etc.)
  - status (TEXT: "unread" | "read")

    - created_at (DATETIME)
- **Events**
    - id (TEXT, PK)
    - event_type (TEXT: "like_post", "comment_post", etc.)
    - user_id (TEXT, FK → Users.id)
    - metadata (JSON)
    - created_at (DATETIME)

5. Scalability and Performance

- Current POC uses **SQLite + in-memory queue** (sufficient for demo).
- For **100–1M DAUs**, scale plan:
    - Replace in-memory queue → **Kafka/RabbitMQ**.
    - Replace SQLite → **PostgreSQL/MySQL (sharded)**.
    - Use **Redis** for caching notifications.
    - Deploy backend on **Kubernetes / Render / AWS ECS**.
    - Serve frontend via **CDN**.

6. Limitations

- In-memory queue loses events if server crashes.
- SQLite is not suitable for millions of concurrent users.
- No push/email integration yet.
- No analytics/metrics dashboard yet.

7. Conclusion

The **Insyd Notification POC App** provides a strong foundation for delivering notifications within the Insyd ecosystem. While the POC uses lightweight tools (SQLite, in-memory queue), the design ensures **scalability, modularity, and extensibility**, making it suitable to evolve into a production-grade system with minimal redesign.