**Insyd Notification System (POC)**

**1. Introduction**

The **Insyd Notification System** is designed to provide real-time notifications for users when important events occur, such as:

* A followed user creates a new post
* A user follows another user
* Other system events (future extensions)

The system ensures that notifications are delivered both in real-time and can be retrieved via REST API endpoints for persistence.

**2. Goals and Objectives**

* Allow users to follow each other.
* Create and store notifications for relevant users.
* Deliver notifications in real-time using WebSockets.
* Enable frontend UI to display notifications clearly.
* Persist notifications in a MongoDB Atlas database.

**3. System Components**

**3.1 Backend (Node.js + Express + MongoDB)**

* **API Endpoints:**
  1. POST /api/users → Create new user.
  2. POST /api/users/:followerId/follow/:followeeId → Follow a user.
  3. POST /api/posts → Create a post and generate notifications.
  4. GET /api/notifications/:userId → Retrieve notifications for a user.
  5. POST /api/notifications/:notifId/read → Mark notification as read.
* **Models (Mongoose):**
  1. **User:** \_id, name, followers (array of User IDs)
  2. **Post:** \_id, author (User ID), content, createdAt
  3. **Notification:** \_id, recipient (User ID), sender (User ID), type, title, message, post (Post ID), isRead, createdAt
* **Real-time Notifications:**
  1. Implemented using **Socket.IO**.
  2. Each user subscribes to a room named by their \_id.
  3. Notifications are emitted to the room of relevant users.

**3.2 Frontend (React + Axios + React Router)**

* **Pages:**
  + Home.jsx → Dashboard, basic welcome, link to notifications.
  + Notifications.jsx → Displays list of notifications for the logged-in user.
  + NotFound.jsx → 404 page.
* **Components:**
  + NotificationList.jsx → Fetches notifications via API and displays them.
  + NotificationBell.jsx → Optional real-time notification indicator.
* **State Management:**
  + Local useState and useEffect hooks.
  + Socket.IO listener updates notification state in real-time.

**4. Data Flow**

1. **User Registration**
   * User (Alice/Bob) created via POST /api/users.
   * Stored in MongoDB Atlas with empty followers array.
2. **Follow Operation**
   * Bob follows Alice via POST /api/users/:followerId/follow/:followeeId.
   * Alice’s followers array is updated to include Bob.
3. **Post Creation**
   * Alice creates a post via POST /api/posts.
   * Backend creates notifications for each follower (Bob).
   * Notification includes title and message fields.
   * Notification stored in MongoDB Atlas and emitted to Socket.IO room.
4. **Notification Retrieval**
   * Bob opens /notifications in UI.
   * React component calls GET /api/notifications/:userId.
   * Notifications are displayed sorted by createdAt.

**5. Database Schema**

**User Collection:**

{

"\_id": "ObjectId",

"name": "String",

"followers": ["ObjectId"]

}

**Post Collection:**

{

"\_id": "ObjectId",

"author": "ObjectId",

"content": "String",

"createdAt": "Date"

}

**Notification Collection:**

{

"\_id": "ObjectId",

"recipient": "ObjectId",

"sender": "ObjectId",

"type": "String",

"title": "String",

"message": "String",

"post": "ObjectId",

"isRead": "Boolean",

"createdAt": "Date"

}

**6. Real-time Notification Flow**

Alice creates post → Backend creates post → Backend creates notifications for followers → Socket.IO emits notifications to followers → Bob's UI receives notification in real-time

**7. API Endpoints Overview**

| **Method** | **Endpoint** | **Description** |
| --- | --- | --- |
| POST | /api/users | Create a new user |
| POST | /api/users/:followerId/follow/:followeeId | Follow a user |
| POST | /api/posts | Create a post and notifications |
| GET | /api/notifications/:userId | Get notifications for a user |
| POST | /api/notifications/:notifId/read | Mark a notification as read |

**8. Future Enhancements**

* Add notification types: likes, comments, mentions.
* Add pagination for notifications.
* Integrate authentication for user sessions.
* Allow filtering unread/read notifications.
* Push notifications via email or mobile.

**9. Deployment**

* Backend hosted on **localhost** (development) or cloud server.
* MongoDB Atlas used for persistent storage.
* Frontend React app hosted on **localhost:5173** or Netlify.
* Socket.IO handles real-time communication.