💬 DM (Direct Messages) Testing Guide

This guide explains how to test the Instagram-style Direct Messaging system in the Kalpix Frontend Tester.

📚 Key Concepts

What is a Channel?

A **Channel** is a conversation container that holds messages between participants. Think of it as a chat room or thread.

Channel Type	Description	Use Case
Direct	1-on-1 conversation between two users	Private DMs like Instagram
Group	Multi-user chat with 2+ participants	Group conversations
Bot	Chat with an automated bot	Customer support, games

Inbox vs Message Requests (Instagram-style)

When someone who is **NOT your friend** (not a mutual follower) sends you a DM:

- The message appears in your **"Requests"** tab (not your inbox)
- You can **Accept** (moves to inbox) or **Decline** (blocks conversation)
- The sender sees the message in their inbox but doesn't know if you've seen it

Friends (mutual followers) can DM each other directly without requests.

🧪 Testing Scenarios

Prerequisites

1. **Start the backend**: `cd kalpix-backend && docker-compose up`
2. **Start the frontend tester**: `cd kalpix-frontend-tester && npm start`
3. **Open two browser windows** (or one regular + one incognito)

Scenario 1: DM Between Friends (No Request Needed)

1. **Browser A**: Login with User A, go to **Social** tab, search for User B
2. **Browser A**: Send follow request to User B
3. **Browser B**: Login with User B, go to **Social** tab → **Follow Requests**
4. **Browser B**: Accept User A's follow request
5. **Browser B**: Send follow request back to User A
6. **Browser A**: Accept User B's follow request
7. ✅ **Now they are friends (mutual followers)**
8. **Browser A**: Go to **Chat** tab → **Create** → Select "Direct Message" → Enter User B's ID
9. **Browser A**: Send a message

10. **Browser B**: Go to **Chat** tab → **Inbox** → Should see the message directly (no request!)

Scenario 2: DM from Non-Friend (Message Request Flow)

1. **Browser A**: Login with User A (new account, follows nobody)
2. **Browser B**: Login with User B (does NOT follow User A)
3. **Browser A**: Go to **Chat** tab → **Create** → Select "Direct Message" → Enter User B's ID
4. **Browser A**: Send a message ("Hey!")
5. **Browser B**: Go to **Chat** tab → Click **Requests** filter
6. **Browser B**: Should see the message request with Accept/Decline buttons
7. **Browser B**: Click **Accept**
8. ✅ **Message now moves to inbox, you can chat normally**

Scenario 3: Group Chat

1. **Browser A**: Go to **Chat** tab → **Create**
2. Select **"Group Chat"**
3. Enter channel name: "Test Group"
4. Enter participant IDs: `user_id_1, user_id_2, user_id_3` (comma-separated)
5. Click **Create Channel**
6. All participants can see and send messages in the group

Scenario 4: Typing Indicators & Read Receipts

1. Open the same channel in **two browsers** (two different users)
2. **User A**: Start typing in the message input
3. **User B**: Should see "User A is typing..." below the messages
4. **User A**: Send the message
5. **User B**: The message appears; when User B views it, User A sees "Seen" indicator

---

🔑 Where to Find User IDs

1. **From Profile Page**: Login → Go to **Profile** tab → Your User ID is displayed
2. **From Console**: Login → Open browser DevTools → Console → Look for session info
3. **From Nakama Console**: Go to `http://localhost:7351` → Login (admin/loginkro) → Users

---

📝 Channel View Filters

Filter	Shows
All	Every channel you're part of
Inbox	Normal chats + outgoing requests + accepted requests
Requests	Incoming pending DM requests from non-friends

🤖 Bot Chat (Advanced)

Bot chats are for automated conversations. To test:

1. Create a channel with type **"Bot"**
2. Enter a bot user ID (e.g., `bot_1`, `bot_customer_support`)
3. Note: Bots need to be implemented on the backend to respond

---

🔧 Troubleshooting

Issue	Solution
WebSocket not connected	Go to Home page first, then return to Chat
Can't see typing indicators	Ensure both users have WebSocket connected
Messages not appearing	Click 🔄 Refresh or reload the page
S3 upload errors	Check AWS credentials in `.env` file

📊 Real-time Features

Feature	How It Works
Typing Indicators	Broadcasts via WebSocket stream when user types
Read Receipts	Marked when recipient views message, broadcasts "Seen"
New Messages	Push to channel stream, auto-refresh on receive

🎯 Quick Start Checklist

- [ ] Backend running (`docker-compose up`)
- [ ] Frontend running (`npm start`)
- [ ] Created at least 2 test accounts
- [ ] Found user IDs from profile page
- [ ] Tested DM creation
- [ ] Tested message sending/receiving
- [ ] Tested typing indicators (need WebSocket connection)
- [ ] Tested read receipts