Real■Time Chat App (Typing Indicator + Online Status)

Overview

This Django-based project is a real■time chat platform that supports one■to■one and group messaging. Using WebSockets (Django Channels), users see live message updates, typing indicators, and online presence without refreshing the page.

Key Features

- Instant messaging powered by WebSockets.
- Typing indicator to show when someone is composing a message.
- Online/offline status with automatic updates.
- Private and group chat rooms.
- User authentication and profiles.
- Message history stored in the database.

Project Structure

```
project_root/
  manage.py
  chat/
    models.py
    views.py
    consumers.py
    routing.py
  templates/
  static/
```

Installation & Setup

1. Install Python 3.10+ and Node (optional for builds).
2. Create a virtual environment: python -m venv venv
3. Activate it: (Windows) venv\Scripts\activate | (Mac/Linux) source venv/bin/activate
4. Install dependencies: pip install -r requirements.txt
5. Apply migrations: python manage.py migrate
6. Create an admin user: python manage.py createsuperuser
7. Run Redis (required for WebSockets) — e.g., redis-server (local install or Docker).
8. Start the server: python manage.py runserver
9. Open http://127.0.0.1:8000 to access the app.

How Real■Time Messaging Works

The application uses Django Channels to create persistent WebSocket connections. When a user sends a message, the message is broadcast to all connected clients in that chat room. Presence and typing states are updated instantly through channel events.

Admin Panel

Visit */admin* using your admin credentials to manage users, chat rooms, and stored messages. Admins can moderate or delete inappropriate conversations if necessary.

Example Use Case

- Two users log in and join the same chat room.
- User A starts typing — User B immediately sees a typing indicator.
- User A sends a message which appears instantly on both screens.
- If one user leaves, their status switches to offline automatically.

Screenshots (Mock Layout Preview)

[Chat List + Online Users Preview]
[Conversation Window with Typing Indicator]
[Group Chat Room]
[Admin Panel Messages View]

What This Project Teaches

- Using Django Channels to build WebSocket-based apps.
- Managing real■time events such as typing and presence.
- Designing database models for conversations and messages.
- Handling authentication and permissions in chat systems.
- Structuring scalable real■time backends with Redis.

Possible Enhancements

- Add read receipts and message reactions.
- Implement file and image sharing.
- Add push notifications for new messages.
- Deploy on a production server using Daphne or Uvicorn + Nginx.