Tamiti Sprint 5 - Chatroom & Real-Time Messaging

Objective:

Build a scalable, Slack-like messaging system with support for:

- Channels (public/private/direct)
- One-on-one Direct Messaging
- File attachments
- Real-time WebSocket integration with Django Channels
- Integration with Virtual Assistants (VAs)

Models Implemented:

1. Channel - group/private/direct conversation container
2. ChannelMember - tracks user-channel membership
3. ChannelMessage - stores messages in a channel
4. MessageFileUpload - handles attachments
5. DirectThread - 1-on-1 chat session
6. DirectMessage - messages in a direct thread
7. DirectMessageFile - attachments for DMs

Enums:

- ChannelType: public, private, direct

Admin Panel Features:

- Inline members & file attachments
- Filters for channel type, creator, and participants
- Search by name and user

API Views (DRF ViewSets):

- /api/chat/channels/
- /api/chat/channel-messages/
- /api/chat/direct-threads/
- /api/chat/direct-messages/

All views enforce permissions to scope data to the current user.

Factories & Tests:

- Provided factories for all models using FactoryBoy
- Tests validate:
  - Message creation
  - Membership enforcement
  - Direct thread interaction

Run with:

```
pytest -v chatroom/tests/
```

Setup Commands:

pip install channels channels_redis

python manage.py makemigrations chatroom

python manage.py migrate

Prepare ASGI & redis config for real-time in next sprint.

Integration Plans:

- VAs will respond in channels via process_va_chat

- Each assistant is a StaffRole-backed bot identity