

Two-Person Chat App Backend Architecture

PROJECT STRUCTURE

server/

```
  └── src/
    ├── server.ts          # App entry point
    ├── config/db.ts       # MongoDB connection
    ├── models/             # Database schemas
    ├── routes/             # API endpoints
    ├── controllers/        # Route logic
    ├── services/            # Business logic
    ├── websocket/          # Realtime messaging
    ├── middleware/         # Auth / validation
    └── types/               # Shared types
```

ARCHITECTURE OVERVIEW

REST API:

- Create users
- Send chat request
- Accept chat
- Load message history

WEBSOCKET:

- Send realtime messages
- Track connected users
- Broadcast messages

FILE RESPONSIBILITIES

server.ts:

Bootstraps Express server, connects database, and initializes WebSocket.

models:

Define MongoDB schemas like User, Chat, and Message.

routes:

Declare endpoints like POST /chat/request.

Routes only map endpoints to controllers.

controllers:

Handle request logic and call services.

services:

Contain business logic and database interaction.

Reusable by both REST controllers and WebSocket handlers.

websocket:

Handles realtime events such as MESSAGE_SEND.

Maintains connected client map.

FLOW EXAMPLE

1. User sends invite → POST /chat/request
2. Controller calls chat.service.ts
3. Chat stored with status 'pending'
4. When accepted → status becomes 'accepted'
5. WebSocket allows messaging