AI Chatbot Template - File Structure Guide

Overview

This AI chatbot template uses a modern full-stack architecture with a Node.js backend and React frontend, designed for real-time chat functionality with document processing capabilities.

Technology Stack

- **Backend**: Node.js + Express.js + Socket.IO
- **Database**: MongoDB (primary) + Redis (caching/real-time)
- **Frontend**: React 18 + Vite + Tailwind CSS
- **Document Processing**: pdf-parse, mammoth, natural
- **Deployment**: Render.com + GitHub Actions

Root Directory Structure

```
ai-chatbot-template/
├── .github/                    # GitHub Actions CI/CD workflows
├── .gitignore                  # Git ignore patterns
├── README.md                   # Project documentation
├── package.json               # Workspace root dependencies
└── apps/                      # Application modules
```

Root Files

File	Purpose
`.gitignore`	Excludes node_modules, .env files, and build artifacts from version control
`README.md`	Project documentation and setup instructions
`package.json`	Workspace configuration for managing multiple apps (backend/frontend)

CI/CD Pipeline (`.github/workflows/`)

```
.github/
└── workflows/
    ├── deploy-backend.yml      # Backend deployment to Render
    └── deploy-frontend.yml     # Frontend deployment automation
```

Workflow Files

File	Purpose
`deploy-backend.yml`	Automates backend deployment to Render when code is pushed to main branch
`deploy-frontend.yml`	Automates frontend build and deployment process

Backend Application (`apps/backend/`)

```
apps/backend/
├── src/                    # Source code
├── knowledge/              # Document storage
├── tests/                  # Test files
├── .env.example           # Environment variables template
├── .env                   # Environment variables (local)
├── package.json           # Backend dependencies
└── server.js              # Application entry point
```

Backend Root Files

File	Purpose
`server.js`	Application entry point, starts Express server and Socket.IO
`package.json`	Backend dependencies (Express, Socket.IO, MongoDB, Redis, etc.)
`.env.example`	Template for environment variables
`.env`	Local environment variables (API keys, database URLs)

Backend Source Code (`apps/backend/src/`)

Configuration (`src/config/`)

```
src/config/
├── database.js            # MongoDB connection setup
├── redis.js               # Redis client configuration
├── openrouter.js         # OpenRouter API configuration
└── index.js              # Configuration aggregator
```

File	Purpose
`database.js`	MongoDB connection, error handling, and connection pooling
`redis.js`	Redis client setup for caching and real-time features
`openrouter.js`	OpenRouter API client configuration for AI responses
`index.js`	Exports all configuration modules

Controllers (`src/controllers/`)

```
src/controllers/
├── chat.controller.js       # Chat message handling
├── knowledge.controller.js  # Document management
└── health.controller.js     # Health check endpoints
```

File	Purpose
`chat.controller.js`	Handles chat requests, AI responses, conversation management
`knowledge.controller.js`	Document upload, processing, and retrieval
`health.controller.js`	API health checks and system status

Middleware (`src/middleware/`)

```
src/middleware/
├── auth.middleware.js        # Authentication logic
├── error.middleware.js       # Global error handling
├── rateLimit.middleware.js   # Rate limiting protection
└── cors.middleware.js        # CORS configuration
```

File	Purpose
`auth.middleware.js`	User authentication and session validation
`error.middleware.js`	Centralized error handling and logging
`rateLimit.middleware.js`	Prevents API abuse with request limiting
`cors.middleware.js`	Cross-origin resource sharing configuration

Models (`src/models/`)

```
src/models/
├── Conversation.model.js     # Chat conversation schema
├── Message.model.js          # Individual message schema
├── Document.model.js         # Uploaded document metadata
└── Chunk.model.js            # Document text chunks for search
```

File	Purpose
`Conversation.model.js`	MongoDB schema for chat conversations
`Message.model.js`	MongoDB schema for individual messages
`Document.model.js`	Schema for uploaded document metadata
`Chunk.model.js`	Schema for processed document chunks (for RAG)

Routes (`src/routes/`)

```
src/routes/
├── chat.routes.js            # Chat API endpoints
├── knowledge.routes.js       # Document API endpoints
└── index.js                  # Route aggregator
```

File	Purpose
`chat.routes.js`	Defines chat-related API endpoints (/api/chat/*)
`knowledge.routes.js`	Defines document management endpoints (/api/knowledge/*)
`index.js`	Combines all routes and exports main router

Services (`src/services/`)

```
src/services/
├── openrouter.service.js     # AI API integration
├── embedding.service.js      # Text embedding generation
├── vector.service.js         # Vector similarity search
├── chunking.service.js       # Document text chunking
├── knowledge.service.js      # Knowledge base management
└── cache.service.js          # Redis caching operations
```

File	Purpose
`openrouter.service.js`	Handles AI model requests and responses
`embedding.service.js`	Generates text embeddings for semantic search
`vector.service.js`	Performs vector similarity searches
`chunking.service.js`	Splits documents into searchable chunks
`knowledge.service.js`	Manages document knowledge base
`cache.service.js`	Redis operations for caching and real-time data

Socket Handlers (`src/sockets/`)

```
src/sockets/
├── handlers/
```