# Chatbot Project — DOCLING

Generated: October 22, 2025

## Title

Chatbot (React frontend + FastAPI backend) — HR / Legal / L1 / L2 domains

## Project overview

This project is a starter chatbot application that demonstrates a React (Vite) frontend connected to a FastAPI backend and a pluggable LLM client (ChatGROQ) with a mock fallback. The system supports domain routing (HR, Legal, L1, L2), conversational sessions, and a simple UI for development.

Purpose: provide a local-ready scaffold to build an internal chatbot for HR, Legal, or Support workflows.

## Architecture

- Frontend: React (Vite). Provides a dark-themed chat UI with a left-hand sidebar domain selector. Proxies `/api` → backend in dev.

- Backend: FastAPI. Exposes `/api/chat` and `/health`. Contains a ChatGROQ client wrapper (mock when API key is missing).

- LLM: pluggable client `ChatGROQClient` in `backend/app/llm/chatgroq\_client.py`. When `CHATGROQ\_API\_KEY` is present the client calls the remote API; otherwise returns deterministic mock replies for UI development.

## Quick start (PowerShell)

1) Backend

cd C:\Users\dhaya\Desktop\chatbot\backend

python -m venv .venv

# if policies allow

.\.venv\Scripts\Activate.ps1

pip install -r requirements.txt

# create .env from .env.example and set CHATGROQ\_API\_KEY (optional)

python -m uvicorn app.main:app --port 8000

If PowerShell prevents activation because of execution policies, run pip with the interpreter in the venv directly:

.\.venv\Scripts\python.exe -m pip install -r requirements.txt

.\.venv\Scripts\python.exe -m uvicorn app.main:app --port 8000

2) Frontend

cd C:\Users\dhaya\Desktop\chatbot\frontend

npm install --legacy-peer-deps

npm run dev

Open the Vite URL (default http://localhost:5173) to use the UI.

## Environment variables

- `CHATGROQ\_API\_KEY` — (optional) API key to enable real LLM calls

- `CHATGROQ\_BASE\_URL` — base URL for the ChatGROQ API (default placeholder in `.env.example`)

- `BACKEND\_PORT` — port for the FastAPI server (default 8000)

## API contract

- POST `/api/chat`

- Request JSON:

- `domain?: "auto"|"hr"|"legal"|"l1"|"l2"`

- `session\_id?: string`

- `messages: [{role: "user"|"assistant", content: string}]`

- Response JSON:

- `reply: string`

- `domain: string`

- `session\_id: string`

### Example request

{

"domain":"auto",

"messages":[{"role":"user","content":"How do I apply for leave?"}]

}

### Example response (mock)

{

"reply":"(mock-hr) I received your message: 'How do I apply for leave?'. This is a starter reply.",

"domain":"hr",

"session\_id":"..."

}

## Frontend usage

- Domain selector moved to the left sidebar (Auto, HR, Legal, L1, L2).

- Composer: press Enter to send (Shift+Enter for newline).

- Session ID: returned by backend and displayed in the sidebar. Re-send the `session\_id` in requests to continue context.

## Troubleshooting

- `Activate.ps1` blocked: adjust execution policy or use the venv Python directly.

- Vite peer dependency errors: use `npm install --legacy-peer-deps`.

- If the LLM API is unreachable, the backend returns a 500 with details; when no key is set the backend runs in mock mode.

## Next steps / roadmap

- Persist conversations (Redis or DB) to survive restarts.

- Replace naive domain routing with intent classifier or LLM-based routing.

- Add CI (GitHub Actions) for tests and linters.

- Improve frontend UX: streaming responses, avatars, timestamps.

## Important files

- `backend/app/main.py` — FastAPI entry

- `backend/app/routers/chat.py` — chat endpoint, domain routing, session handling

- `backend/app/llm/chatgroq\_client.py` — LLM client wrapper (mock fallback)

- `frontend/src/App.jsx` — main React app (dark theme, sidebar)

---

Generated by assistant — will also create `DOCLING.docx` in the repository.