AiAgent Chat Frontend — Multi-file React + Tailwind App

This canvas contains a ready-to-drop multi-file React + Tailwind frontend for the GPT-style multi-project chat UI you asked for. Put the frontend under your repository root (recommended path shown below) and run `npm install` / `npm run dev`.

Recommended file placement

Place the package at:

```
/home/cloudx/AiCloudxAgent/frontend
```

(If your repo root differs, place it under `<repo-root>/frontend` and adapt the `BACKEND_BASE` in `src/api/client.js`.)

Files included (contents follow below)

- package.json
- tailwind.config.cjs
- postcss.config.cjs
- README.md
- src/main.jsx
- src/index.css
- src/App.jsx
- src/routes.jsx
- src/api/client.js
- src/components/Sidebar.jsx
- src/components/ProjectCard.jsx
- src/components/ChatList.jsx
- src/components/ChatWindow.jsx
- src/components/PersonaSelector.jsx
- src/components/ToolsPanel.jsx
- src/hooks/usePolling.js
- public/index.html
- docs/api.md (small API spec)

> **Note:** The canvas contains the full source code for the above files as code blocks. Copy the files exactly into the `frontend` folder to run.

Quick start

1. From the repository root:

```
cd /home/cloudx/AiCloudxAgent
cd frontend
npm install
npm run dev
```

1. Edit `src/api/client.js` to point `BACKEND_BASE` to your backend (default `http://localhost:3000`).
2. Use the UI to inspect projects, open per-project chats, choose personas, and run tools (installer/repair) via secure backend endpoints.

API & Security (short)

See `docs/api.md` for the small API spec the frontend expects. **Important:** Backend endpoints must authenticate and validate before running any system-level commands.

Included code (copy each file into the `frontend` folder):

package.json

```
{
  "name": "aiagent-frontend",
  "version": "0.1.0",
  "private": true,
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "start": "vite preview"
  },
  "dependencies": {
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "axios": "^1.4.0",
    "clsx": "^1.2.1"
  },
  "devDependencies": {
    "vite": "^5.0.0",
    "tailwindcss": "^3.4.0",
    "postcss": "^8.4.21",
    "autoprefixer": "^10.4.14"
  }
}
```

tailwind.config.cjs

```
module.exports = {
  content: ['./index.html','./src/**/*.{js,jsx}'],
  theme: { extend: {} },
  plugins: []
}
```

postcss.config.cjs

```
module.exports = { plugins: { tailwindcss: {}, autoprefixer: {} } }
```

public/index.html

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>AiAgent Chat UI</title>
  </head>
  <body>
    <div id="root"></div>
    <script type="module" src="/src/main.jsx"></script>
  </body>
</html>
```

src/index.css

```
@tailwind base;
@tailwind components;
@tailwind utilities;

html,body,#root { height: 100%; }
body { @apply bg-slate-50 text-slate-800; }
```

src/main.jsx

```
import React from 'react'
import { createRoot } from 'react-dom/client'
import App from './App'
import './index.css'

createRoot(document.getElementById('root')).render(<App />)
```

src/api/client.js

```
import axios from 'axios'
export const BACKEND_BASE = process.env.REACT_APP_BACKEND_BASE || 'http://
localhost:3000'
const api = axios.create({ baseURL: BACKEND_BASE, timeout: 60000 })
export default api
```

src/App.jsx

```
import React, {useEffect, useState} from 'react'
import Sidebar from './components/Sidebar'
import ChatWindow from './components/ChatWindow'
import ToolsPanel from './components/ToolsPanel'
import api from './api/client'

export default function App(){
  const [projects, setProjects] = useState([])
  const [activeProject, setActiveProject] = useState(null)
  const [activeChat, setActiveChat] = useState(null)

  useEffect(()=>{ api.get('/api/
projects').then(r=>setProjects(r.data)).catch(()=>setProjects([])) },[])

  return (
    <div className="h-screen flex">
      <Sidebar projects={projects} onOpenProject={(p)=>{setActiveProject(p);
setActiveChat(null)}} />
      <div className="flex-1 flex flex-col">
        <div className="flex-1">
          <ChatWindow project={activeProject} chat={activeChat}
onSelectChat={setActiveChat} />
        </div>
        <ToolsPanel project={activeProject} />
      </div>
    </div>
  )
}
```

src/components/Sidebar.jsx

```
import React from 'react'
import ProjectCard from './ProjectCard'

export default function Sidebar({projects,onOpenProject}){
  return (
    <aside className="w-80 bg-white border-r p-4 overflow-auto">
      <h2 className="text-lg font-semibold mb-4">Projects</h2>
```

```
      <div className="space-y-2">
        {projects.length===0 && <div className="text-sm text-slate-500">No
projects found</div>}
        {projects.map(p=> <ProjectCard key={p.id} project={p}
onOpen={()=>onOpenProject(p)} />)}
      </div>
    </aside>
  )
}
```

src/components/ProjectCard.jsx

```
import React from 'react'
export default function ProjectCard({project,onOpen}){
  return (
    <div className="p-2 rounded hover:bg-slate-50 cursor-pointer"
onClick={onOpen}>
      <div className="font-medium">{project.name}</div>
      <div className="text-sm text-slate-500">{project.description || '—'}</
div>
    </div>
  )
}
```

src/components/ChatWindow.jsx

```
import React, {useEffect,useState} from 'react'
import ChatList from './ChatList'
import PersonaSelector from './PersonaSelector'
import api from '../api/client'

export default function ChatWindow({project, chat, onSelectChat}){
  const [chats, setChats] = useState([])
  const [messages, setMessages] = useState([])
  const [input, setInput] = useState('')
  const [persona, setPersona] = useState('assistant')

  useEffect(()=>{ if(project) { api.get(`/api/projects/${project.id}/
chats`).then(r=>setChats(r.data)).catch(()=>setChats([])) } },[project])

  useEffect(()=>{ if(chat){ api.get(`/api/projects/${project.id}/chats/$
{chat.id}/
messages`).then(r=>setMessages(r.data)).catch(()=>setMessages([])) } },
[chat,project])

  const send = async ()=>{
    if(!project) return
    const payload = { text: input, persona }
    setInput('')
```

```
    const res = await api.post(`/api/projects/${project.id}/chats/${chat.id}/
messages`, payload).catch(()=>null)
    if(res) setMessages(res.data)
  }

  return (
    <div className="h-full flex">
      <div className="w-72 border-r p-3 overflow-auto">
        <ChatList chats={chats} onSelect={onSelectChat} />
      </div>
      <div className="flex-1 p-4 flex flex-col">
        <div className="flex items-center justify-between mb-2">
          <h3 className="text-xl font-semibold">{project ? project.name :
'Select a project'}</h3>
          <PersonaSelector persona={persona} onChange={setPersona} />
        </div>
        <div className="flex-1 bg-white rounded p-3 overflow-auto border">
          {messages.map((m,i)=>(<div key={i} className={`mb-3 $
{m.role==='user'?'text-right':''}`}><div className="inline-block p-2 rounded
bg-slate-100">{m.text}</div></div>))}
        </div>
        <div className="mt-3 flex gap-2">
          <input value={input} onChange={e=>setInput(e.target.value)}
className="flex-1 border rounded p-2" placeholder="Message" />
          <button onClick={send} className="px-4 py-2 bg-sky-600 text-white
rounded">Send</button>
        </div>
      </div>
    </div>
  )
}
```

src/components/ChatList.jsx

```
import React from 'react'
export default function ChatList({chats,onSelect}){
  return (
    <div className="space-y-2">
      {chats.map(c=> (
        <div key={c.id} className="p-2 rounded hover:bg-slate-100 cursor-
pointer" onClick={()=>onSelect(c)}>
          <div className="font-medium">{c.title || 'Chat'}</div>
          <div className="text-xs text-slate-500">{c.last_activity || ''}</
div>
        </div>
      ))}
    </div>
  )
}
```

src/components/PersonaSelector.jsx

```
import React from 'react'
export default function PersonaSelector({persona,onChange}){
  const opts=[{id:'assistant',label:'Assistant'},{id:'dev',label:'Dev Bot'},
{id:'commands',label:'Cmds'}]
  return (
    <select className="border rounded p-1" value={persona}
onChange={e=>onChange(e.target.value)}>
      {opts.map(o=> <option key={o.id} value={o.id}>{o.label}</option>)}
    </select>
  )
}
```

src/components/ToolsPanel.jsx

```
import React, {useState} from 'react'
import api from '../api/client'

export default function ToolsPanel({project}){
  const [log, setLog] = useState('')
  const runTool = async (tool)=>{
    if(!project) return alert('Select project')
    setLog('Running...')
    try{
      const r = await api.post(`/api/projects/${project.id}/tools/run`, {
tool })
      setLog(JSON.stringify(r.data, null, 2))
    }catch(e){ setLog('Error or timeout') }
  }

  return (
    <div className="border-t p-3 bg-white">
      <div className="flex gap-2">
        <button onClick={()=>runTool('doctor')} className="px-3 py-1 bg-
yellow-500 rounded">Run Doctor</button>
        <button onClick={()=>runTool('install')} className="px-3 py-1 bg-
green-600 text-white rounded">Install</button>
        <button onClick={()=>runTool('repair')} className="px-3 py-1 bg-
red-600 text-white rounded">Repair</button>
      </div>
      <pre className="mt-3 max-h-40 overflow-auto text-xs bg-slate-900 text-
white p-2 rounded">{log}</pre>
    </div>
  )
}
```

src/hooks/usePolling.js

```
import { useEffect } from 'react'
export default function usePolling(fn, ms=3000){
  useEffect(()=>{ let alive=true; const tick=async()=>{ if(!alive) return;
await fn(); setTimeout(tick,ms) }; tick(); return ()=>{ alive=false } },
[fn,ms])
}
```

docs/api.md

```
# Minimal backend API spec (frontend expectations)

## Authentication
Frontend expects a secure token-based auth or session. **Never** expose
endpoints that execute shell commands without auth & RBAC.

## Endpoints

- GET /api/projects
  - returns: [{id,name,description}]

- GET /api/projects/:pid/chats
  - returns chats list: [{id,title,last_activity}]

- GET /api/projects/:pid/chats/:cid/messages
  - returns messages array [{role:'user'|'assistant',text}]

- POST /api/projects/:pid/chats/:cid/messages
  - body: { text, persona }
  - returns new messages array

- POST /api/projects/:pid/tools/run
  - body: { tool: 'install'|'repair'|'doctor' }
  - returns: { status, output }

- GET /api/status
  - returns global status (dpkg, services, ports)

Security: require admin token and validate request origin.
```

If you want I will:

1. Produce a small Express backend stub (secure endpoints + mock behavior) so you can wire the UI locally.
2. Split the React files into separate components and produce a zip ready to drop into your repo.
3. Add unit tests (Vitest/Jest) and basic Cypress smoke tests for the UI.

Which next step do you want?