**Metatron AI Tutor UX & System Design Blueprint - Metatron**

**🌐 Purpose**

This document consolidates the full design vision and user experience strategy for the Metatron AI Tutor MVP, modeled after the intuitive structure of ChatGPT but extended to support role-based education programs, knowledge base planning, session coaching, and multi-user management.

**🎯 Design Goals**

* Create a professional, scalable UX for both first-time and returning users
* Follow a ChatGPT-style hybrid layout with persistent sidebar and dynamic main content
* Support team-based learning led by **Team Leaders** (not "teachers")
* Allow each user to:
  + Upload and manage their own Knowledge Bases (KBs)
  + Define personalized Learning Plans
  + See progress and interact via a coaching chat interface
  + View personal dashboards
* Team Leaders should:
  + Monitor learners
  + Manage KB presets and prompt strategies
  + Create and assign AI-based learning programs
  + Manager and configure the AI LLMs for the system
  + I single user setup the user will also be the Team Leader

**🧱 App Architecture & Screens**

**🔲 UI Mode Matrix**

|  |  |  |  |
| --- | --- | --- | --- |
| **Mode** | **Entry Point** | **Main Panel** | **Sidebar Content** |
| Login | First Visit | Login fields | Empty or branding |
| KB Setup | After login or from sidebar ➕ | Upload + plan | AI engine, debug toggle |
| Tutor Session | When plan + KB selected | ChatGPT-style panel | KB switcher, plan summary, tools |
| Settings | From sidebar ⚙️ | API key, default model, support | All user roles |
| Admin Dashboard | If role = team\_leader  or Org manager | Student list, KB distribution, analytics | Extra admin tools |
| Personal Dashboard | Any user | Usage logs, session history | Profile tools |

**✅ Finalized User Flow**

|  |  |  |  |
| --- | --- | --- | --- |
| Step | User Action | System Behavior | Screen / Frame |
| 0️⃣ | Opens app | Loads login/init state | LoginScreen |
| 1️⃣ | Enters email or username | Sets session vars | LoginScreen |
| 3️⃣ | Clicks **Add KB & Plan** | Switches to KB setup screen | KBSetupScreen |
| 2️⃣ | Selects AI engine (Free/OpenAI) | If OpenAI → request key | EngineSelectScreen |
| 4️⃣ | Uploads file(s) | Runs embedding and stores vector DB | KBSetupScreen |
| 5️⃣ | Fills learning plan | Stores to kb\_meta | KBSetupScreen |
| 6️⃣ | Clicks On KB in Sidebar start Tutor Session | Loads chat interface | TutorSessionScreen |
| 7️⃣ | Tutor chat UI active | Uses session KB + plan | TutorSessionScreen |

Returning users see existing KBs listed in the sidebar and can resume or create new ones.

**🧠 Role-Based Experience**

**👤 Learner (default user)**

* Can:
  + Upload KBs
  + Set goals
  + Access tutoring interface
  + View personal dashboard
* Sidebar shows:
  + 📚 My Knowledge Bases
  + ➕ Add New KB
  + 💬 Tutor Chat (per KB)
  + 📊 My Dashboard
  + ⚙️ Settings

**👥 Team Leader (admin or manager role)**

* Has all learner features, plus:
  + Manage student roster
  + Push KBs to users
  + See group analytics
  + Define global prompt strategies (I am not sure about this I think this is our IP and we will do this in the code setup on some kind of super Admin back end config
* Sidebar shows all learner tools **plus**:
  + 🧠 AI Program Settings
  + 📤 Distribute KBs
  + 📈 Team Analytics
  + 👥 User Management
* Single Power User
  + If single user setup the single user will also be a team leader or when the user logs on we may need to ask the user when the register if the are a single suer, a team leader or admin of organisation level

**📐 Visual Layout (inspired by ChatGPT)**

**Sidebar (collapsible )**

* Knowledge Base selector (teach track/Coach
* User options
* New session or KB setup
* Role-aware admin tools (if authorized)

**Main Panel (mode-dependent)**

* Dynamic content:
  + Login
  + Upload
  + Plan form
  + Tutor chat
  + Personal dashboard
  + Admin view (if team\_leader)

**🧪 UX Rules**

* ✅ Tutor UI never shown unless plan + KB are both set
* ✅ Prompt box hidden until session starts
* ✅ Org Admin & Team Leader tools gated by role
* ✅ Re-entry logic based on stored KBs and last session state (the prompts will need to consider the full history of the user or student, the plan and progress made and the goals of the user to decide how the tutor proceeds from there, unless the user instructs otherwise, using default quick prompt suggestions hand around the prompt tutor
* Engine selection must occur before uploading KB
* Prompt tutor chat should reflect the selected model (GPT-4 or Mistral)

**📌 Example Use Case Scenarios**

**📚 First-Time Student**

1. Logs in
2. Uploads textbook PDF
3. Sets “Exam Prep - June 2025” plan
4. Starts tutor session
5. Gets guided chat-based study support
6. Later visits dashboard to review chat, progress, reminders

**👥 Team Leader Coaching Session (advanced features, not for MVP)**

1. Logs in with Team Leader role
2. Views class roster and their KBs
3. Pushes a shared KB to 10 users or pushes a subset of the knowledge base for tutoring
4. Adds a standard goal prompt for them (this could dovetail with the above point)
5. Monitors who started, who’s idle
6. Sends nudges or edits goal plans live

**👥 Org Administrator assigns Team Leaders (advanced features, not for MVP)**

1. TBC
2. TBC
3. TBC

**🧱 Data Structure Notes**

* Vector DB stores per user+kb: org\_user\_kbname
* Plans stored in SQLite: kb\_meta table
* Session logs stored per user, timestamped
* Admin views query grouped logs per KB / user

**✅ You’re Refining the MVP to Match a Real UX System**

**✔ Key Enhancements You Made (and I support them):**

| **Original Step** | **Your Update** | **Why It’s Smart** |
| --- | --- | --- |
| **“Clicks Start Setup”** | **“Clicks Add KB & Plan”** | **💡 Clear, action-based label. Matches sidebar UX.** |
| **“Tutor chat auto-appears”** | **“Clicks On KB in Sidebar to start Tutor Session”** | **✅ Aligns with ChatGPT — user chooses what to open.** |
| **Unclear setup boundaries** | **You separated setup (upload + plan) from chat** | **✅ Prevents premature tutor prompts. Matches real-world flow.** |
| **No clarity on default options** | **You added “returning users see KBs listed in sidebar”** | **✅ Makes re-entry logic obvious. Improves onboarding.** |

**🧠 You’re Shifting the MVP into Platform Thinking**

**Yes, absolutely:**

* **You're now thinking in terms of a persistent UI framework (sidebar, content area)**
* **You want to support multiple KBs, roles, dashboards — without breaking linear flow**
* **You are designing for scalability (one user today, full org later)**

**And that’s exactly right. This is how modern learning systems evolve.**

**✅ Summary: Yes, I Understand and Agree With You**

* **The sidebar must always be there**
* **The user should see and choose KBs (like selecting a chat in ChatGPT)**
* **The chat interface only appears when a KB is active and ready**
* **Team Leaders should have access to “program management” views**
* **Every user (even a solo one) needs a path to view their own history, goals, progress**
* **The role = team\_leader or org\_admin unlocks coaching tools**
* **The prompt area must stay hidden until all setup steps are complete**

Okay I need to capture some new content based on the upgrade

C:\MyDevProjects\Metatron

├── venv/

├── backend/ ← Python logic, auth, embeddings, etc.

├── frontend/ ← React + Tailwind UI

├── sync\_to\_gdrive.py

├── run\_sync.bat

├── app.py ← legacy or placeholder

├── requirements.txt

└── last\_sync.log

We are dropping the streamlit platform and design.

C:\MyDevProjects\Metatron\backend

├── main.py

├── auth.py

├── chat\_engine.py

├── embedder.py

├── file\_handler.py

├── logger.py

C:\MyDevProjects\Metatron ← Project root

├── backend ← All FastAPI Python backend logic

│ ├── main.py ✅ FastAPI entry point

│ ├── auth.py

│ ├── chat\_engine.py

│ ├── embedder.py

│ ├── file\_handler.py

│ └── logger.py

├── frontend ← React + Vite + Tailwind UI

│ └── src/

├── venv ← Python virtual environment

├── requirements.txt

└── README.md (optional)