Awesome — your folder structure is *exactly what a professional backend project should look like*, and you're ready to plug FastAPI into it.

Let me now explain **FastAPI** in the simplest possible way:

🧠 What is FastAPI?

FastAPI is a **Python web framework** used to build APIs (Application Programming Interfaces).
It lets your frontend (like Streamlit or a mobile app) talk to your backend logic and data.

Think of it as:

> 🗣️ Your app's backend **"listener"** that responds to requests like:
>
> - "Hey, I uploaded a PDF"
> - "Hey, give me the extracted tasks"
> - "Hey, answer this question"

🔁 What FastAPI does in your project

Component	What FastAPI helps you do
`src/routes.py`	Defines all the HTTP endpoints like `/upload`
`src/upload.py`	Logic behind what to do when someone uploads a file
`src/query.py`	Logic to process a question and return an answer
`src/tasks.py`	Handles tasks from extracted documents

🛠️ Real-World Analogy

Imagine your project is a restaurant:

- `Streamlit` = the **waiter taking the order**
- `FastAPI` = the **kitchen manager receiving orders**
- `src/` = the **kitchen** where chefs (your logic files) do the real work

So when Streamlit says:

> "POST this new PDF file to `/upload`"

FastAPI routes that request to `upload.py`, which saves the file and triggers parsing/embedding.

🏗️ FastAPI Project Flow (Simplified)

📤 POST /upload
→ saves file
→ calls chunker, embedder
→ returns: "Uploaded and processed."

💬 POST /ask
→ gets user's question
→ calls embedder + FAISS + LLM
→ returns: answer + source chunks

✅ GET /tasks
→ fetches extracted tasks for a project
→ returns: list of tasks

🔧 How FastAPI is structured in your project

File	Role
`src/routes.py`	Registers all the API routes (upload, query, tasks)
`src/upload.py`	Handles file upload & triggers chunk/embed logic
`src/query.py`	Handles question answering (retrieval + LLM)
`src/tasks.py`	Reads extracted tasks from DB and returns them

`src/__init__.py`	Makes `src/` a module

▶️ Example: Basic FastAPI Code (`main.py`)

Here's the **simplest FastAPI server** that connects `upload.py` and `query.py`:

```
from fastapi import FastAPI
from src import upload, query, tasks

app = FastAPI()

# Register routes
app.include_router(upload.router, prefix="/upload")
app.include_router(query.router, prefix="/ask")
app.include_router(tasks.router, prefix="/tasks")

@app.get("/")
def home():
    return {"message": "TenderIQ backend is live ✅"}
```

✅ How to Run It

```
uvicorn main:app --reload
```

- Runs on: `http://localhost:8000`
- You can test `/`, `/upload`, `/ask`, etc.
- Streamlit UI will send requests to these routes

❓ How is it different from Streamlit?

Feature	Streamlit	FastAPI
Used for	UI (chat, upload, forms)	Backend logic & data APIs
Code style	Interactive scripts	HTTP request handlers

Communication	Runs on user browser	Runs on API server
Connects to	User	LLM, database, embedding, etc.
