LifeLens AI Project Idea

Absolutely! Here's a **complete workflow for your PharmaLens MVP**, optimized for **17 days**, featuring AI, multimodal input (image + text), and real-world utility.

🧠 PharmaLens Project Workflow

🎯 *Goal: AI-powered pharmaceutical assistant with pill detection, prescription parsing, interaction checking, and chat support.*

🗂️ 1. Project Structure (Modular & Scalable)

```bash
pharmalens/
│
├── app/                # Frontend (Streamlit or React)
│   ├── main.py         # Main UI file
│   ├── components/     # Modular UI components
│   └── assets/         # CSS, images, etc.
│
├── backend/            # Backend logic (FastAPI optional)
│   ├── api.py          # Routes for model inference, DB, chatbot
│   └── models/         # ML logic (OCR, chatbot, scanner)
│       ├── ocr_utils.py
│       ├── pill_identifier.py
│       ├── chatbot_agent.py
│       └── leaflet_generator.py
│
├── data/               # Drug info, interaction files
│   ├── drug_info.csv
│   └── interaction_rules.json
│
├── notebooks/          # For testing models/OCR separately
│
├── requirements.txt
```

```
├── README.md
└── run.sh / app.py        # Entry point
```

🧩 2. Core Functional Workflow

🔍 1. Pill Identifier

Input: Image of pill or strip
Flow:

1. Frontend uploads image.
2. `ocr_utils.py` runs Tesseract/Google Vision OCR → Extract text (e.g., "Paracetamol 500mg").
3. `pill_identifier.py` → Lookup in `drug_info.csv` or API → Get drug details.

Output: Name, dose, brand, expiry date, common use.

📄 2. Prescription Interpreter

Input: Handwritten or printed prescription (image)
Flow:

1. OCR → Cleaned text → Chunked per line
2. Regex/LLM parses into:
    - Drug name
    - Dosage
    - Duration
    - Frequency

Output: Display structured table in UI
✅ Bonus: Add button to fetch leaflet per medicine

💬 3. Ask a Pharmacist (Chatbot Agent)

Input: User text like "What is Ciprofloxacin used for?"
Flow:

1. Frontend sends query.
2. `chatbot_agent.py` uses:
   - Gemma 3n (local) or OpenRouter (Gemini/GPT)
   - Drug context lookup from CSV
3. Return rich answer: use case, precautions, etc.

Output: Instant drug explanation chatbot

🚫 4. Drug Interaction Checker

Input: List of current medicines (manual entry or OCR list)
Flow:

1. `interaction_rules.json`: Contains risky combinations (e.g., Aspirin + Warfarin)
2. Rule-based or LLM-assisted search
3. Flag any red/yellow alerts with reasons

Output: Alert message with explanation
✅ Bonus: Suggest alternatives

📃 5. Smart Leaflet Generator

Input: Drug name
Flow:

1. Lookup drug in `drug_info.csv`
2. Auto-generate a **plain-language patient info sheet**:
   - What it does

- How to take
- Side effects
- When to avoid

3. Export to **PDF or text**

💡 Bonus Flows (if time permits)

- **Multilingual Mode**: Input in Hindi → Translate → Process → Output back in Hindi
- **Voice Input**: Use Whisper API or Python SpeechRecognition

---

🌐 Optional APIs & Resources

Resource	Use
RxNorm API	Drug identifiers and interactions
openFDA Drug API	Labeling and adverse events
MedlinePlus	Leaflet data
Tesseract OCR	For scanning prescriptions
Hugging Face Transformers	Use `gemma`, `bert`, or `distilGPT2` for chatbot

✅ Final User Flow Summary (from UX)

1. User uploads **prescription or pill photo**
2. App extracts & identifies drug info
3. User gets structured data + can:
    - View usage leaflet
    - Ask chatbot

- Check drug interactions

4. Export PDF / Save to history

Would you like me to:

- Build a sample `main.py` with working Streamlit code?
- Provide dummy data for drug info and interaction examples?
- Help you write the README and pitch lines?

You're building something **seriously industry-grade**. Let's crush it in 17 days.

Export to Google Doc Copy with formatting Select for Multi-select
Export to Google Doc Copy with formatting Select for Multi-select

Perfect! Here's a **parallel 17-day workflow split into two focused tracks**:

🔹 **AI Integration Workflow**
🔹 **Web Development (Frontend + Backend) Workflow**

These run **side-by-side**, so if you have a teammate or you're managing time smartly (e.g., morning AI, evening frontend), this plan **maximizes efficiency and MVP delivery**.

✅ Part 1: AI Integration – 17-Day Workflow

Goal: Integrate OCR, pill detection, prescription parsing, chatbot, interaction checker, leaflet generation.

Day	Task	Tools/Models
Day 1	Set up `models/` folder, create dummy inputs & outputs for pill and prescription OCR	Pillow, Tesseract, Python
Day 2	Build **OCR pipeline** for pill images → test with `tesseract`	pytesseract
Day 3	Create pill name matcher using fuzzy string search or cosine sim from embeddings	`fuzzywuzzy`, `sentence-transformers`

Day	Task	Tools/Models
Day 4	Integrate **drug info lookup** from CSV or scraped data (RxNorm / FDA)	pandas, API
Day 5	Build **prescription parser** from OCR text → extract [name, dose, duration]	Regex + manual rules
Day 6	Test prescription parser with multiple image types (handwritten + printed)	pytesseract
Day 7	Build **basic chatbot backend** → "Ask a Pharmacist" with static drug data	`transformers`, `Gemma 2b/7b` or OpenRouter
Day 8	Fine-tune or prompt-tune chatbot on pharmacy Q&A (optional)	LangChain, LlamaIndex
Day 9	Build **Drug Interaction Checker** logic (rule-based + optional LLM fallback)	Rule-based + OpenFDA
Day 10	Create **Smart Leaflet Generator**: Given a drug, generate plain-language text	Templates + LLM
Day 11	Format leaflet → export as PDF	`pdfkit`, `reportlab`, `markdown2pdf`
Day 12	Test all modules together with sample inputs	End-to-end test
Day 13	Optimize responses for chatbot + leaflet	Prompt design
Day 14	Add multilingual support in chatbot (Gemma or via translation pipeline)	MarianMT or Gemini
Day 15	Add confidence levels for pill detection & chatbot answers	Probability thresholding
Day 16	Save all model outputs + logs + sample predictions	Logging system
Day 17	Final polish, refactor, document `models/` folder	README + usage guide

🌐 Part 2: Web Development (Frontend + Backend) – 17-Day Workflow

Goal: Create a beautiful, functional, modular web app using Streamlit or React + FastAPI.

Day	Task	Tools/Frameworks
Day 1	Set up GitHub repo, virtual environment, basic folder structure	Git, venv
Day 2	Set up **Streamlit or React app layout**: Sidebar, pages	Streamlit / React + Tailwind
Day 3	Design Home Page + About Page + Upload Page	UI components
Day 4	Build File Upload UI for: prescription image & pill image	Streamlit FileUploader / HTML input
Day 5	Connect uploaded file to **OCR preview output** on UI	FastAPI or direct call
Day 6	Build **Drug Info UI** → show matched name, brand, dose, use	Streamlit components
Day 7	Build **Prescription Table UI** from parsed result	`st.table`, DataGrid
Day 8	Build **Ask a Pharmacist chatbot UI**	Chatbox UI / LLM connector
Day 9	Connect chatbot backend → live responses on UI	REST API or local run
Day 10	Design **Drug Interaction Checker UI**	Input pills list + display alerts
Day 11	Connect interaction checker output → style with colors	Markdown / HTML
Day 12	Build **Smart Leaflet PDF viewer and download** button	`st.download_button`
Day 13	Create **history view / logs** page (bonus)	SQLite / session cache
Day 14	Add **loading spinners + error handling**	UX polish

Day	Task	Tools/Frameworks
Day 15	Add **dark mode, mobile responsive UI**	Tailwind/Bootstrap (React)
Day 16	Deploy to **Streamlit Cloud / Hugging Face Spaces / Vercel**	CI/CD
Day 17	Final polish + record demo video + UI README	Loom, Markdown

---

🎁 Deliverables at the End of 17 Days:

- ✅ Functional **AI-based PharmaLens MVP**
- ✅ Working UI with 5 flagship features
- ✅ GitHub repo with clean code, README, demo video
- ✅ Deployed app (Streamlit Cloud / Hugging Face)
- ✅ PDF report / submission doc (if for a hackathon)

🧩 Bonus: Want Ready-Made Templates?

I can give you:

- A ready-to-use `main.py` Streamlit layout
- Sample OCR + drug interaction dataset
- Chatbot prompt template for "Ask a Pharmacist"
- Leaflet markdown → PDF export pipeline

Would you like any of these now to kickstart implementation?

Export to Google Doc Copy with formatting Select for Multi-select
Export to Google Doc Copy with formatting Select for Multi-select