Awesome! Since you're aiming to implement everything in **1 week**, here's a **7-day high-efficiency plan** tailored for your stack, goal, and schedule 💪

**⚡ 7-Day Plan to Build Your AI-Powered Auto Scheduler (With Cloudinary, EDA, Java + Python + Angular)**

**🔥 Day 1 – Project Setup & Planning**

**Tasks:**

* Create project structure:
  + /frontend-angular
  + /backend-java
  + /backend-python
* Setup GitHub repo
* Plan API routes and frontend flow (on paper or draw.io)

**Learn:**

* Microservices basics (short YouTube or blog)
* Angular, Java, Python folder structure

✅ Output:

* Project folders ready, API flow mapped

**🔥 Day 2 – Java + Cloudinary File Upload**

**Tasks:**

* Setup Spring Boot project
* Add file upload endpoint with MultipartFile
* Integrate Cloudinary Java SDK
* Upload file → Get Cloudinary URL → Return in response

**Learn:**

* Spring Boot REST API
* Cloudinary Java SDK

✅ Output:

* API that takes a file, uploads to Cloudinary, and returns URL

**🔥 Day 3 – Python EDA Microservice**

**Tasks:**

* Setup Flask or FastAPI backend
* Accept Cloudinary URL via POST request
* Fetch file using Python (requests.get)
* Perform basic EDA using pandas, matplotlib, seaborn

**Learn:**

* Flask or FastAPI routing
* How to fetch and analyze remote file (Cloudinary)

✅ Output:

* Python API that takes URL and returns EDA JSON + charts (as base64 or links)

**🔥 Day 4 – Frontend File Upload Integration**

**Tasks:**

* Setup Angular frontend with file upload form
* Call Java backend to upload file → show Cloudinary URL
* Show “EDA in progress” loader

**Learn:**

* Angular file upload + HttpClient
* Connecting frontend to backend

✅ Output:

* Working Angular frontend to upload file and display Cloudinary link

**🔥 Day 5 – Java-to-Python Communication & Display EDA**

**Tasks:**

* After getting Cloudinary URL in Java backend, call Python API
* Receive EDA results
* Return them to Angular frontend
* Display graphs + summary on frontend

**Learn:**

* Java HTTP client to call Python REST API
* Parsing JSON & returning to Angular
* Angular chart display using ng2-charts / Chart.js

✅ Output:

* Full flow working: Upload → EDA → Results shown

**🔥 Day 6 – Task Management Features**

**Tasks:**

* Implement task CRUD (Angular + Java)
* Simple UI to add/view/edit/delete tasks
* Store data in MySQL or MongoDB

**Learn:**

* Angular forms + services
* Spring Boot CRUD APIs
* Connecting DB to backend

✅ Output:

* Basic task management ready

**🔥 Day 7 – Auto-Scheduler Logic + UI Polish + Final Testing**

**Tasks:**

* Implement basic scheduling algorithm in Python (based on priority & time)
* Connect to task data
* Add final touches to UI (loading states, error handling, etc.)
* Do complete end-to-end testing

✅ Output:

* Scheduler working, EDA working, clean UI

**✅ Tools Checklist:**

* VS Code / IntelliJ
* Postman for testing APIs
* GitHub for version control
* Cloudinary account
* Python environment (Anaconda or venv)
* Angular CLI setup

Want me to generate a **Notion template** or **Google Sheet** for tracking this day-by-day?