**🔍 Current Phase Summary: “Culture Project App Development”**

**✅ Known Vision & Goals**

| **Area** | **Detail** |
| --- | --- |
| **You Are** | The visionary/project manager |
| **I Am** | The lead developer (ChatGPT), interfacing with GitHub Copilot |
| **Copilot Is** | The embedded coding assistant who implements |
| **Main Goal** | Build a modular app that generates and manages a comprehensive cultural content library |
| **Initial Output Target** | A deeply structured **SharePoint** knowledge repository |
| **Eventual Output Target** | A public-facing **web app or portal** |
| **Pain Point** | Volume and complexity of content; you’re the sole curator |

**📦 Current System Inventory**

**🧠 Conceptual Framework / Meta-Level**

* Cultural entries are structured: region → language → subculture → traits
* Supporting files (Word, PDF, etc.) already exist and are partially parsed
* PGLS Culture Guide style is your working model
* You are curating and enriching from a combination of:
  + Academic docs
  + Internal PGLS knowledge
  + External open sources
  + Implicit AI-curated insights

**🔧 Dev Stack & Tooling**

| **Component** | **Status** |
| --- | --- |
| **VS Code + Copilot** | Actively in use |
| **Docker** | Running a sample app (Node-based); app containerization is now viable |
| **GitHub** | Not fully confirmed if repo is connected, but VS Code project workspace appears clean |
| **Language** | Node.js (currently), but Python or hybrid possible |
| **Langdetect** | Mentioned — potential NLP tool for text input classification |
| **Multimedia Handling** | Not yet implemented (text-only so far) |
| **Autonomy Level** | No automation or background processing yet — manual triggers only |

**🧭 Gaps & Challenges**

| **Area** | **Issue** |
| --- | --- |
| **App Purpose** | Still general — needs to translate from “organize culture” to a functional user experience |
| **Data Flow** | No ETL pipeline yet for turning source files into structured cultural entries |
| **Storage** | No database or content management layer yet |
| **Scheduler/Daemon** | No “run while I’m away” automation implemented |
| **UI/UX** | None — CLI/terminal only or test-server-only via Docker |
| **SharePoint Output** | No integration or automation toward SP export yet |
| **Content Validation** | No NLP-based deduplication, enrichment, or classification layer |
| **Scalability** | Manual content processing is the bottleneck — no async, task queue, or indexing logic yet |

**🧠 Recommendation: Next Milestone**

🔨 **Build the Minimum Viable Cultural Library Engine (MV-CLE)**

**📁 Proposed Functional Stack:**

| **Layer** | **What It Does** | **Status** |
| --- | --- | --- |
| **Ingestor** | Ingest .docx, .pdf, .txt cultural files | Needed |
| **Parser** | Extract structured cultural fields (region, name, language, traits) | Needed |
| **Classifier** | (Optional) Use langdetect or regex/NLP to validate and categorize entries | Starting |
| **Database/Store** | Flat-file JSON store or SQLite for now | Not built |
| **Output Formatter** | Translates data into SharePoint-friendly JSON or HTML blocks | Not built |
| **Dockerized Runner** | Can be triggered on boot or schedule | Partial (basic container works) |
| **Cron/Scheduler** | Ensures regular updates/autonomy | Not present |

**🧭 Proposed Next Steps**

Here’s a logical sequence to build toward your automation dream:

1. **Define the cultural data schema**  
   Let’s lock in what fields we’re tracking for each culture (region, language, gender norms, communication, etc.)
2. **Build the parser module**  
   Converts .docx and .pdf entries into JSON following the schema
3. **Create a content\_store folder and simple database (JSON/SQLite)**  
   So entries can be stored, updated, and reused
4. **Build CLI triggers for:**
   * Parse all docs
   * Add new entry
   * Export for SharePoint
5. **Dockerize the cultural processor**  
   So the full tool runs on a schedule or can be triggered remotely
6. **Integrate SharePoint output/export**  
   Export as static HTML, JSON, or scriptable .aspx stubs to drop into SharePoint

Would you like me to now:

🔁 **Refactor your Copilot prompt** to start building this Minimum Viable Cultural Library Engine?

If yes, we’ll start with:

**Step 1: Define culture\_entry\_schema.json and setup parsing logic.**

Just say the word — I’ll build the Copilot-ready prompt and code for it.