Project Plan — Final Project

Objective: Review course progress (Weeks 1–6) and present a practical, reproducible plan for a final project: a digital evidence ingestion → cleaning → analysis → visualization pipeline.

---

1. Brief summary of the progressive project (Weeks 1–6)

Week 1 — Setup & scoping - Set up development environment (text editor, Python virtualenv, git). - Defined scope: build a lightweight forensic evidence pipeline for incident triage.

Week 2 — Data acquisition & basic scripts - Implemented acquisition helpers and small scripts to collect artifacts from sample devices (system info collector and browser-history parser). Example artifacts targeted: system info dumps, browser history, registry hives, and exported files for images and logs.

Week 3 — Artifact parsing - Built parsers for specific artifact types (e.g., EXIF metadata extractor for images, and structured parsing for browser history). Tested on sample files and verified basic fields (timestamp, device, path).

Week 4 — Normalization & integrity - Added hashing for integrity checks and basic normalization (timestamp formats, canonicalized field names). - Began assembling parsed outputs into a single, consistent on-disk format for downstream analysis.

Week 5 — Cleaning pipeline - Implemented deduplication, basic decoding of binary payloads (base64 / encoded blobs), and early PII redaction rules. - Created scripts to convert parsed outputs into tabular formats for quick inspection.

Week 6 — Preliminary analysis & report skeleton - Performed simple frequency/time-series checks (activities over time) and sketched visualization ideas. Assembled a rough project README and started a final-report outline.

(Note: the above reflects iterative work on small scripts for system/browser artifacts, EXIF readers, and a cleaning/ normalization pipeline.)

---

2. Final Project Plan — Overview

Project title: Digital Evidence Pipeline for Incident Triage

Goal: Ingest a single consolidated "raw evidence" file, clean & normalize artifacts, run intelligent analyses to find anomalies and extract entities, then present findings through clear, reproducible visualizations and a concise final report.

Deliverables

- `project_plan.md` (this file)
- `raw_evidence.jsonl` (hypothetical raw evidence input)
- `clean_evidence.parquet` (cleaned dataset for analysis)
- `analysis_notebook.ipynb` (reproducible analysis steps and plots)
- `dashboard.html` or `dashboard/` (visual interactive output)
- `final_report.pdf` and `slides.pdf` (final write-up and slide deck)

---

3. Hypothetical new "raw evidence" file — description & schema

Filename: `raw_evidence.jsonl` (JSON Lines; one JSON object per line).
Why JSONL: streamable, easy to ingest incrementally, tolerant to mixed artifact types.

Each JSON object (example schema):

```
{
  "evidence_id": "uuid-1234",
  "acquired_at": "2025-09-30T03:12:45Z",
  "device_id": "DEVICE-001",
  "source_path": "\\\\Users\\Alice\\Pictures\\IMG_001.jpg",
  "artifact_type": "image|browser_history|system_log|registry|pcap|other",
  "raw_payload_base64": "<optional base64 blob for file contents>",
  "parsed_metadata": {
    "exif": { "datetime_original": "2025-08-01T10:02:00", "gps": {"lat": 14.6,
"lon": 121.0}},
    "browser": { "url": "https://example.com", "title": "Example" }
  },
  "hashes": { "md5": "...", "sha256": "..." },
  "acquisition_method": "live|imaged|exported",
  "original_format": "jpg|sqlite|evtx|pcap",
  "notes": "free-text examiner notes"
}
```

Additional details: - `artifact_type` drives parsing logic. - `raw_payload_base64` is optional — used only when we need the file contents inline (images, small files). Larger binary evidence can be referenced by path and transported alongside the JSONL file. - Use ISO 8601 timestamps and UTC for all times.

---

4. Step-by-step data cleaning plan

1. **Ingest & validation**
2. Read JSONL line-by-line; validate required fields (evidence_id, acquired_at, artifact_type). Reject or quarantine malformed lines to `quarantine/`.

3. **Integrity checks**
4. Recompute hashes for any payloads present; compare with `hashes` field; flag mismatches.
5. **Normalize timestamps**
6. Convert all timestamps to UTC and ISO-8601 canonical form. Keep original timezone info in `parsed_metadata` if present.
7. **Decode/parse payloads**
8. If `raw_payload_base64` present: decode, then run artifact-specific parsers (EXIF for images, SQLite parser for browser DB, EVTX for Windows logs, tangential tools for PCAP).
9. **Schema mapping & flattening**
10. Map parsed fields into a canonical column set (e.g., `event_time`, `device_id`, `actor`, `action`, `object`, `geo_lat`, `geo_lon`, `text`), leaving a free-text `raw_parsed_json` for unstructured content.
11. **Deduplication**
12. Use content hashes and (artifact_type, event_time, device) heuristics to drop exact duplicates and collapse near-duplicates.
13. **PII handling & anonymization**
14. Apply rules to redact or pseudonymize PHI/PII fields in analysis builds. Keep an encrypted mapping for reproducibility if needed.
15. **Enrichment**
16. Geo-lookup for coordinates, WHOIS / domain reputation lookup for URLs (optional), and mapping known hashes using local blacklist/whitelist.
17. **Quality checks & logging**
18. Produce a cleaning log with counts, errors, and actions applied.
19. **Persist cleaned data**
20. Save as Parquet or SQLite (schema + partitioning by `device_id` and date) for fast querying.

---

5. Intelligent analysis approaches

Primary analyses: - **Timeline reconstruction** — order events across devices (align by `event_time`) to build an incident narrative. - **Anomaly detection** — use statistical / ML-based detectors (e.g., isolation forest, LOF) on features like event frequency, time-of-day, unusual file hashes, or uncommon destination IPs. - **Classification** — classify text artifacts (e.g., classify a text blob as credential-leak, chat, error log) using a lightweight supervised model or rule-based classifier. - **Entity extraction** — use NLP (spaCy) to extract names, email addresses, IPs, domains, file names from text artifacts. - **Clustering & correlation** — group related events by similarity (e.g., clustering by file-hash, IP, or URL) to identify likely campaign artifacts. - **Hash / indicator matching** — match file hashes, IPs, and domains against known IOCs.

Model & tooling notes: - Start with rule-based logic + heuristics; move to simple supervised models if labeled examples exist. - Keep models interpretable; focus on explainability (feature importance, decision rules) for a forensic report.

---

6. Visualization plan

Core visualizations (deliver interactive + static variants): - **Interactive timeline** — events across devices, zoomable; clicking an event shows raw details. - **Activity heatmap** — activity counts by hour/day to show abnormal surges. - **Geospatial map** — plot GPS-tagged artifacts and cluster hotspots. - **Network/entity graph** — nodes as entities (IPs, domains, files, devices) and edges showing observed relationships. - **Sankey / flow chart** — show flow of data between hosts/services (e.g., upload -> external IP). - **Summary KPI cards** — total artifacts, suspicious artifacts, anomalies detected, top devices impacted.

Tools: Jupyter + Plotly/Altair for interactive outputs; export static PNGs with matplotlib when embedding into PDFs. For dashboards, use a static single-page HTML dashboard or a simple Streamlit app.

---

7. Key conclusions & sections to include in final report

1. **Executive summary** — top findings in 3–5 bullets (what happened, when, who/what was affected, confidence).
2. **Scope & data sources** — what was provided, acquisition methods, limits.
3. **Methods** — cleaning steps, enrichment, and analysis methods (brief, reproducible).
4. **Timeline of events** — annotated timeline with supporting artifacts.
5. **Anomalies & IOCs** — anomalous events, matched IOCs, and confidence levels.
6. **Entity relationships** — relevant entities and how they connect (graph snapshots).
7. **Recommendations** — containment, remediation, and next forensic steps.
8. **Limitations & assumptions** — gaps in data, uncertain timestamps, potential false positives.
9. **Appendices** — schema, scripts, raw counts, and commands to reproduce analysis.

---

8. Timeline & checkpoints (suggested)

- **Week 0 (kickoff):** finalize scope & sample data format.
- **Week 1:** implement ingestion + validation; sample ingest complete.
- **Week 2:** implement cleaning pipeline (decoding, parsing, normalization).
- **Week 3:** run enrichment & initial analyses (timeline + basic anomalies).
- **Week 4:** build visualizations and dashboard prototype.
- **Week 5:** finalize analysis, write report, prepare slides.
- **Week 6:** rehearsal & final delivery.

---

9. Reproducibility, ethics & security

- Keep all scripts in a Git repository with a `requirements.txt` or `environment.yml`.
- Log data transformations and checksum outputs to prove chain-of-custody.
- Store any PII in encrypted form; redact in public artifacts.
- Note any legal/ethical constraints when working with real evidence.

---

10. Quick next steps (what I will do first)

- Create a small example `raw_evidence.jsonl` (10–20 mixed artifacts) to iterate the pipeline.
- Implement and test ingestion + timestamp normalization.
- Produce a first-pass timeline visualization to confirm data quality and alignment.

End of project plan.