**Phase 0 – Project Initiation & Planning**

**Goal: Establish the foundation for the TIP project.**

* **Requirements Gathering**
  + **Define primary use cases (threat feed aggregation, enrichment, correlation, alerting, etc.).**
  + **Identify stakeholders: SOC team, threat analysts, customers, MSSPs.**
  + **Determine target industries (finance, healthcare, etc.).**
* **Competitive Analysis**
  + **Review leading TIPs (Anomali, ThreatConnect, MISP, Recorded Future).**
  + **Identify unique differentiators (e.g., AI-powered IOC correlation, automated remediation).**
* **High-Level Architecture Draft**
  + **Data ingestion → normalization → enrichment → analytics → visualization → integration.**
  + **Decide on on-premises, cloud, or hybrid deployment.**
* **Budget & Resource Allocation**
  + **Estimate costs for infrastructure, licenses, team salaries.**
  + **Assign roles: Data/AI Engineer, Backend Engineer, Security/Network Engineer, Frontend/Flutter Dev, DevOps.**

**Phase 1 – Data Acquisition & Ingestion**

**Goal:** Collect raw threat intelligence from multiple, reliable sources in various formats and protocols.

**1.1 Source Identification & Validation**

* Threat feed discovery (commercial, open-source, industry sharing groups)
* Source trustworthiness scoring
* Data freshness checks
* API key/credential management for sources

**1.2 Multi-source Threat Feed Ingestion**

* STIX/TAXII feeds
* Open-source feeds (OSINT) – AlienVault OTX, AbuseIPDB, FeodoTracker, PhishTank, URLHaus, MISP community feeds
* Commercial feeds – Recorded Future, ThreatConnect, Anomali ThreatStream, Intel 471
* Industry-specific feeds – FS-ISAC (finance), Health-ISAC (healthcare)
* Government/standards – CISA feeds, NVD, CERT advisories (US-CERT, ENISA)
* Darknet sources (honeypots, malware sandboxes)
* Unstructured sources: Dark web forums
* Social media
* Internal logs: SIEM, firewall, IDS/IPS alerts.

1.3 **Data Types**

* Indicators of Compromise (IOCs) – IPs, domains, URLs, file hashes, email addresses, registry keys
* Threat actor profiles – names, aliases, motivations, attack patterns
* Vulnerability intelligence – CVEs, CWEs, exploit info
* TTPs (MITRE ATT&CK mapping)
* Exploit kits
* Phishing campaigns
* Malware samples

1.4 **Acquisition Methods:**

API pull

webhook push

periodic crawling

RSS feeds

streaming ingestion/feeds

email parsing

file drops

1.5 Operational Requirements

Rate-limiting/ frequency control (real-time, hourly, daily)

Retry logic for unstable APIs

Secure transport (HTTPS/TLS 1.3)

Connection monitoring

Failover sources

1.6 Ingestion Layer Technologies (WHY INGESTION LAYER IS IMPORTANT)

**Why:** Scalability/ High-volume, reliability/ fault-tolerant, ease of transformation.

**Tools:** Logstash, Fluentd, Kafka Connect, custom Python Scripts/Go collectors, Apache NiFi

**Monitoring:** Metrics (latency, error rate), alerting for feed failures, heartbeat checks.

Ingestion validation rules (schema conformity, missing fields).

Ingestion error handling and retry strategies

1.7 **Monitoring & Logging for Ingestion**

* Data volume metrics
* Source downtime alerts
* Feed ingestion lag monitoring

1.8 **Pre-ingestion Validation** *(often skipped but critical)*

* Feed health checks.
* Data quality scoring.
* Source authenticity verification.

**1.9 Connector Development**

* STIX/TAXII clients.
* API clients for vendor feeds (Recorded Future, VirusTotal).
* Web scraping for OSINT.

**Phase 2 – Data Normalization & Storage**

**Goal:** Convert heterogeneous data into a consistent, queryable format and store it efficiently. In other words, Convert raw feeds into a unified, queryable format. In other words, Convert all feeds into a consistent schema.

**2.1 Preprocessing & Cleaning**

* Duplicate IOC removal
* Field standardization (IP, domain, file hash formats)
* Timestamp normalization (UTC conversion)
* Language detection & translation (for multilingual threat reports)
* Encoding fixes.

**2.2 Normalization & Parsing**

* Parsing unstructured text into structured IOC fields
* Mapping custom feed formats to standard schemas (STIX 2.1, OpenIOC, MISP)
* Standardize formats (CSV, JSON, STIX, TAXII) into **unified internal schema**
* Map IOC attributes:  
  id, type, source, first\_seen, last\_seen, threat\_actor, malware\_family, confidence\_score, tags, TLP, enrichment\_data
* Deduplicate repeated IOCs
* Use **STIX 2.1 / TAXII** for interoperability
* Validate syntax (regex for IPs, domains, hash lengths)

**2.3 Database Schema Design**

* **Raw IOC table**
* **Enrichment table** (VirusTotal, WHOIS, Shodan, CVE data)
* **Metadata table** (source info, TLP)
* **Optional**: Neo4j for relationship graphs between actors/malware
* **IOC-centric schema with relationships (threat actor ↔ campaign ↔ IOC).**
* **Historical change tracking (Cumulative Table Design Pattern with complex datatypes)**
* **Schema for IOCs, actors, campaigns, relationships, TTPs**
* **Optimizing for fast lookup & correlation queries**
* **Choosing database types (NoSQL for speed, SQL for relational joins)**

2.4 Tech Choices

**Short-term/** **Hot storage**: Elasticsearch/OpenSearch for fast search & indexing

**Long-term/** **Cold storage**: PostgreSQL (JSONB for enrichment flexibility) or MongoDB

ORM/ Graph DB: SQLAlchemy (Python) or Prisma (Node.js) or (Neo4j) for relationships between threat actors, campaigns, and IOCs.

**2.5 Data Lifecycle Management** *(often missing)*

* Data lifecycle management (hot → warm → cold tiers)
* Data retention policy enforcement.
  + Define TTL for different data types.
* Tiered storage for cost optimization.

Phase 3 – Threat Data Management

**Goal:** Organize, enrich, and classify threat intelligence for operational use. In other words, Enable powerful search, enrichment, and classification.

**3.1 IOC Storage & Tagging (Sematic NLP STUFF MAYBE HERE)**

* UUID-based unique IDs
* IOC categorization/Tagging (malware, phishing, ransomware, etc.)
* Tagging by:
  + Threat actor
  + Malware family
  + Campaign name
  + TLP classification
  + Sector
* Confidence scoring & tagging with source reliability

**3.2 Advanced Search & Filtering**

* Full-text search for tags or enrichment fields
* Indexing for speed (Elasticsearch or PostgreSQL trigram indexes)
* Search by type (IP, domain, hash, CVE)
* Filter by (Query Filters):
  + Source
  + Threat actor
  + TLP
  + Date range
* Fuzzy search for partial matches
* Query filters (date range, source, TLP level, IOC type)
* IOC pattern search

**3.3 IOC Enrichment**

* Auto-query APIs after ingestion:
  + VirusTotal – malware detection, hash reports
  + WHOIS API – domain ownership, registration
  + Shodan – open ports, banners, vulnerabilities
  + CIRCL CVE API – vulnerability details
  + ASN (Autonomous System Number) resolution
  + GeoIP enrichment
* Asynchronous enrichment via queue workers:
  + RabbitMQ or Kafka + Celery/BullMQ

**3.4 TLP (Traffic Light Protocol) Classification & Labeling**

* TLP: RED (private), AMBER (limited), GREEN (community), WHITE (public)
* Enforce access control based on TLP in API & UI
* Automated + analyst override.
* Propagation rules enforcement.

**3.5 Threat Actor Profiling** *(missing)*

* Linking IOCs to known APT groups ( Campaign attribution tags (e.g., APT groups).
* MITRE ATT&CK mapping.

**3.6 Data Versioning & Change Tracking**

* Historical changes to IOCs and metadata (Cumulative Table Design Pattern)
* Rollback capabilities

**3.7** **Automated Enrichment**

* IP → ASN, geolocation, WHOIS.
* Domain → passive DNS, SSL cert history.
* File hashes → malware sandbox reports.

Phase 4 – Analysis & Correlation

**Goal:** Derive context and insights from collected intelligence. In other words, Turn collected intel into actionable insights.

**4.1 IOC Correlation**

* Cross-source deduplication & correlation.
* IOC clustering by campaign.
* Match new indicators to historical records
* Detect overlapping campaigns or threat actors
* Time-based correlation (same IOC in different campaigns)
* Infrastructure linkage (domains → IPs → malware samples)

**4.2 Threat Scoring**

* Severity & reliability based on:
  + Source confidence ( Confidence scores based on source credibility & enrichment data).
    - Risk scoring models (rule-based or ML-driven)
    - Contextual risk scoring (industry, geography relevance)
  + Frequency of sighting

**4.3 Timeline Tracking**

* IOC appearance timeline
* Monitor Threat Campaign evolution over time

**4.4 Automated Threat Pattern Recognition** *(missing)*

* ML-based anomaly detection.
* Predictive threat modeling.
* MITRE ATT&CK mapping
* Kill chain phase identification

**4.5 Anomaly Detection & Predictive Analytics**

* Detect sudden surges in specific IOC types
* Predict future activity based on historical patterns

4.6 More ML AND DL

**4.6.1 Threat Pattern Detection**

* Time-series anomaly detection.
* Graph analysis for campaign tracking.

4.6.2 **Predictive Threat Modeling**

* ML models for early detection of emerging threats.

4.6.3 **Natural Language Processing**

* Summarize threat reports.
* Extract entities from unstructured intel.

4.6.4 **Risk Scoring & Prioritization**

* Calculate impact probability per asset or client.

**Phase 5 – Visualization & Reporting (User Interface: FLUTTER)**

**Goal:** Make intelligence actionable via intuitive interfaces and summaries. In other words, Give SOC teams a clear, fast understanding of threats.

**5.1 Dashboards**

* Active threat counts, trending IOCs
* IOC activity heatmaps
* Real-time alerts view
* campaign heatmaps.

**5.2 Geolocation Mapping**

* Threat Attack origin & target location plotting
* IP → country mapping

**5.3 Relationship Graphs**

* Graph-based actor–IOC–campaign visualizations: Show links between actors, malware, IOCs.
* Graph DB integration (Neo4j).

**5.4 Automated Reports**

* PDF, HTML, machine-readable reports/ threat summaries for old and new threats(JSON, CSV).
* Scheduled & on-demand generation.
* Executive vs analyst report modes
* Non-technical reports for leadership.

**5.5 Custom Report Builder**

* Drag-and-drop visual analytics
* Export & sharing options

Phase 6 – Security Operations Integration

**Goal:** Integrate with SOC workflows and automated defense systems. In other words, Bridge intelligence with active defense. In other words, Ensure TIP is secure and compliant.

**6.1 SIEM/SOAR Integration**

* Splunk, QRadar, Elastic SIEM, Cortex XSOAR, TheHive.
* Automated playbooks in SOAR

**6.2 Alerting & Notifications**

* Multi-channel alerts (Email, Slack, Teams, PagerDuty).
* Custom severity thresholds

**6.3 API Access**

* REST & GraphQL APIs with authentication.
* API authentication & rate limiting
* SOC tools can query TIP data

**6.4 Incident Response Hooks** *(missing)*

* Push IOCs directly to firewalls, EDRs, WAFs.

**6.5 Integration Testing & Validation**

* Simulated feed injection
* End-to-end alert delivery tests

6.6 EXTRA WORK BOIIIIIII

**SOAR Playbooks**

* Automated blocking of malicious domains/IPs.
* Incident ticket creation in JIRA/ServiceNow.

**Security Hardening**

**Network segmentation, TLS everywhere.**

**API rate limiting, IP whitelisting.**

**Compliance Checks**

**GDPR, ISO 27001, SOC 2.**

**Threat Model for the TIP Itself**

**Identify potential attack vectors on the platform.**

**Phase 7 – Testing & Quality Assurance (MIGHT BE SKIPPED)**

**Goal:** Validate platform stability and accuracy.

* **Unit & Integration Tests**
  + Test ingestion pipelines, enrichment logic, UI components.
* **Performance Testing**
  + Scalability under large feed loads.
* **Security Testing**
  + Penetration testing, vulnerability scanning.
* **User Acceptance Testing (UAT)**
  + Involve SOC analysts for real-world feedback.

**Phase 8 – Governance, Compliance & Security**

**Goal:** Ensure platform integrity, compliance, and data protection. In other words, Ensure lawful and secure handling of threat data.

**8.1 Access Controls**

* RBAC (Role-Based Access Control)
  + Analysts, admins, read-only users
  + Least privilege principle.
* MFA for analysts and admins

**8.2 Auditability & Forensics**

* Track views, edits, deletions
* Audit logs for IOC modifications & user actions
* Immutable logging (e.g., WORM storage)

**8.3 Compliance & Legal**

* GDPR, CCPA, ISO 27001 alignment
* Threat data sharing agreements
* Right to be forgotten workflows.
* Handle personal data securely

**8.4 Data Retention & Disposal**

* Retention policies by IOC type & sensitivity
* Secure data deletion methods
* Auto-delete outdated IOCs
* Legal compliance and cost control.

**8.5 Secrets & Key Management**

* Vault integration for API keys and credentials (HashiCorp Vault, AWS Secrets Manager).
* Rotation policies

**8.6 Incident Response Readiness**

* Response runbooks for platform compromise
* Backup & disaster recovery plans

**8.7 Source Trust Management** *(missing)*

* Vetting feed providers, revoking compromised sources.

**Phase 9 – Deployment & Release Management**

**Goal:** Go live in a stable, secure way.

* **Staging Environment**
  + Simulate production with real data.
* **CI/CD Pipelines**
  + Automated builds, tests, deployments.
* **Progressive Rollout**
  + Beta with select clients → full release.

**Phase 10 – Monitoring & Continuous Improvement**

**Goal:** Keep TIP relevant and effective over time.

* **System Monitoring**
  + Infrastructure health, feed availability.
* **Threat Trend Tracking**
  + Emerging malware, zero-day trends.
* **Feedback Loops**
  + Analyst suggestions for new features.
* **Continuous Model Training**
  + Update ML/NLP models with new data.

**Phase 11 – Documentation & Training**

**Goal:** Ensure adoption and maintainability.

* **Technical Documentation**
  + API references, architecture diagrams.
* **User Manuals**
  + Walkthroughs for analysts and admins.
* **Training Sessions**
  + Webinars, video tutorials, onboarding guides.

FLUTTER AND LLM STAGES (OR SHOULD THEY BE PLACED)

If you’re adding a **Flutter app** and an **LLM** to your Threat Intelligence Platform, they don’t live in a single “phase” — they would interact with **multiple phases** as separate components:

**Where the Flutter App Fits**

* **Purpose:**
  + Acts as the **mobile client** for analysts, SOC teams, or customers to **view reports, search IOCs, get alerts, and query threat intel** on the go.
* **Best Placement in Phases:**
  + **Phase 5 – Visualization & Reporting** → Flutter app is another visualization channel besides dashboards.
  + **Phase 6 – Security Operations Integration** → App receives push notifications for alerts.
  + **Phase 7 – Governance & Security** → Ensure mobile authentication, RBAC, and secure API calls.
* **Needed Additions:**
  + Mobile API gateway layer (between backend and Flutter app)
  + Push notification service (Firebase Cloud Messaging or APNs)
  + Offline cache for reports (optional)

**Where the LLM Fits**

* **Purpose:**
  + Provide **natural language threat intelligence queries**, automated IOC summaries, correlation insights, or analyst support (“Explain this IOC in plain English”).
* **Best Placement in Phases:**
  + **Phase 4 – Analysis & Correlation** → LLM can process and summarize correlation results, find hidden relationships, or perform enrichment Q&A.
  + **Phase 5 – Visualization & Reporting** → Generate executive-friendly summaries, contextualized risk descriptions, and predictive narratives.
  + **(Optional New Phase 8 – AI & Automation)** →
    - IOC auto-classification
    - Threat clustering
    - Predictive attack pattern detection
    - Natural language query interface
* **Needed Additions:**
  + Prompt-engineered interface layer between the database and LLM
  + Rate-limited & sanitized data passing to prevent sensitive leakage
  + Model hosting strategy (local on-prem vs API-based like OpenAI/Azure)

**Overall Integration Flow**

1. **Data & Analysis Layers** (Phases 1–4) produce structured threat intel.
2. **LLM Service Layer** consumes this data:
   * Summarizes reports
   * Answers natural language queries
   * Enhances correlation & enrichment results
3. **API Gateway** serves both the Flutter app and any other client.
4. **Flutter App**:
   * Pulls dashboards & reports
   * Receives alerts
   * Sends natural language questions → forwarded to LLM → returns results.