**Phase 0 – Platform Architecture & Infrastructure Setup**

* Collaborate with Backend & Security Engineers to define **mobile API architecture** (REST/GraphQL/WebSockets).
* Set up **Flutter project structure**, state management (e.g., Riverpod, BLoC), and theming guidelines.
* Configure **CI/CD for mobile builds** (Codemagic, GitHub Actions).
* Establish **secure storage** for sensitive data (tokens, offline IOC cache).

**Phase 1 – Data Acquisition & Ingestion**

* Provide **UI for manual IOC submission** (analyst-reported indicators, incident details).
* Implement **real-time feed views** where ingestion data from the backend can be monitored live.
* Display ingestion status/errors returned from the backend to the user.

**Phase 2 – Data Normalization & Storage**

* Build **search and filter UI** for normalized data (IOC type, source, date range).
* Implement **offline caching** for essential normalized data using SQLite/Hive.
* Optimize data rendering for **large IOC lists** without performance drops.

**Phase 3 – Threat Data Management**

* Create **IOC detail views** with enrichment results (WHOIS, GeoIP, ASN, VirusTotal).
* Add **tag visualization** for TLP classification and IOC categories.
* Implement **saved search and watchlist** features for priority IOCs.
* Build **quick filter toggles** for high-confidence or high-severity threats.

**Phase 4 – Analysis & Correlation**

* Build interactive **timeline views** for IOC activity and campaign evolution.
* Visualize **relationship graphs** (IOC ↔ campaign ↔ actor).
* Integrate with AI Engineer’s **LLM API** for natural language threat queries (chatbot-style UI).
* Display **threat scoring badges and risk levels** in IOC lists.

**Phase 5 – Visualization & Reporting**

* Create **mobile dashboards** with threat heatmaps, attack trends, and geolocation maps.
* Build **customizable widgets** for analysts to track specific metrics.
* Implement **report viewer and download options** (PDF, CSV).
* Enable **sharing of selected threat data** securely via mobile.

**Phase 6 – Security Operations Integration**

* Implement **push notifications** for alerts (via Firebase Cloud Messaging).
* Create **alert center** with filtering by severity and status.
* Build UI to **acknowledge or escalate alerts** from the mobile app.
* Integrate with SOC workflows via backend APIs.

**Phase 7 – Governance, Compliance & Security**

* Work with Security Engineers to implement **MFA and secure login** (OAuth2, biometrics).
* Ensure **role-based UI visibility** (analyst vs admin vs manager views).
* Enforce **secure API calls** with token refresh & revocation logic.
* Prevent sensitive data from being stored insecurely on the device.

**Phase 8 – AI & Automation (if included)**

* Build **LLM-powered threat assistant** in the app for plain-English threat analysis.
* Provide **voice-to-threat-query** functionality for field analysts.
* Display **predictive alerts and suggested mitigations** from AI models.
* Implement **feedback forms** to improve AI outputs via analyst input.

**Expected Outcomes from Flutter Developer’s Work**

* **Intuitive, responsive, and secure mobile interface** for threat intelligence access.
* **Real-time, mobile-friendly dashboards** for SOC and analyst use.
* **Full integration** with backend APIs, AI features, and security controls.
* **Reliable push alert system** that ensures analysts never miss critical threats.
* **Offline-ready app** for use in field operations or low-connectivity environments.

Updated Plan:  
Here’s a role-specific breakdown for the **Flutter Developer** in your TIP pipeline, mapped directly to the phases you listed and considering the presence of Data Engineers, AI Engineers, Backend Engineers, and Security/Network Engineers.

**Flutter Developer – Responsibilities Across TIP Phases**

**Phase 0 – Project Initiation & Planning**

* **Participate in requirements gathering** for mobile client capabilities:
  + Threat feed viewing, IOC search, dashboard visualization, report access, push alerts.
  + Offline access considerations.
* **Define mobile UX goals** in line with SOC/analyst needs.
* Provide **effort estimation** for app development milestones.

**Phase 5 – Visualization & Reporting *(Main Impact Area)***

* **UI/UX Implementation**
  + Build dashboards for active threats, trending IOCs, heatmaps.
  + Implement IOC search and filtering screens.
  + Visualize relationship graphs (actor–IOC–campaign) using Flutter graph libraries.
  + Integrate geolocation maps for attack origins/targets.
* **Custom Reporting**
  + Enable PDF/HTML/JSON/CSV report viewing within the app.
  + Implement filters, export/share capabilities.
* **Data Consumption**
  + Integrate with backend REST/GraphQL APIs.
  + Handle pagination, caching, and real-time data streams (WebSockets or polling).

**Phase 6 – Security Operations Integration**

* **Push Notifications**
  + Implement FCM/APNs integration for real-time SOC alerts.
  + Support custom severity thresholds in user preferences.
* **Authentication & Authorization**
  + Implement secure login (OAuth2, JWT, MFA).
  + Enforce RBAC from backend (analyst/admin/read-only).
* **Incident Interaction**
  + Build UI to review, acknowledge, or escalate alerts directly from the app.

**Phase 7 – Testing & QA**

* Write **widget tests** and **integration tests** for Flutter UI.
* Collaborate with Backend & Security Engineers to perform **end-to-end security testing** (e.g., API calls from mobile).
* Test under **network constraints** (offline mode, high latency).

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

* Ensure **secure data storage** on device:
  + Encrypted local cache (Hive/Secure Storage).
  + Clear sensitive data on logout.
* Comply with GDPR/ISO policies for mobile data handling.
* Handle **data retention policies** (auto-expire cached IOCs).

**Phase 9 – Deployment & Release Management**

* Manage **mobile release cycles**:
  + CI/CD for Flutter builds (Fastlane, Codemagic, GitHub Actions).
  + Beta testing with TestFlight (iOS) and Internal Testing (Play Store).
* Integrate with backend staging environments for pre-release checks.

**Phase 10 – Monitoring & Continuous Improvement**

* Add in-app **feedback collection** for analysts.
* Implement **analytics** (privacy-compliant) to track feature usage and improve UI/UX.
* Work with AI Engineer to surface **LLM-generated summaries** in a user-friendly way.

**Where Flutter Developer Works with Others**

* **Backend Engineer** → Define API contracts, handle data formatting, set up secure API gateways.
* **Data Engineer** → Understand data structures and indexing to optimize mobile queries.
* **AI Engineer** → Integrate LLM results, threat summaries, and NLP-powered search into UI.
* **Security/Network Engineer** → Apply mobile security best practices, secure API endpoints, and enforce compliance rules.