PHASE 1: Planning & Requirements

Step 1: Define the Project Scope

- What threats will you detect?
- Who are the users?
- What data sources will you use?

Step 2: List Functional Requirements

- Log ingestion
- Threat visualization
- Anomaly detection
- Real-time alerts

Step 3: Choose Tech Stack

- **Backend**: Python + Flask or FastAPI
- **Frontend**: React.js / Streamlit / Dash
- **Storage**: MongoDB / PostgreSQL / Elasticsearch
- **Visualization**: Plotly, D3.js, Chart.js
- **ML**: Scikit-learn, PyOD, Keras

PHASE 2: Data Pipeline Setup

Step 1: Collect Data

- Firewalls (e.g., pfSense)
- Syslog
- Windows/Linux logs
- External APIs (AlienVault OTX, MISP, AbuseIPDB)

Step 2: Normalize and Store Data

- Use ETL tools (Logstash, Fluentd, custom Python scripts)
- Store logs in a centralized DB (MongoDB or Elasticsearch)

Step 3: Implement Tagging & Enrichment

- Add geolocation (GeoIP)
- Check IP/domains against threat feeds
- Tag with MITRE ATT&CK tactics

PHASE 3: Intelligence & Analysis Engine

Step 1: Basic Threat Detection

- Rule-based logic (e.g., failed login > 5 in 1 min)
- Blacklist matching (via IP/domain feeds)

Step 2: Apply Data Science / ML

- Train/test anomaly detection models (Isolation Forest, KMeans, Autoencoders)
- Cluster similar attack types
- Assign severity/risk scores

Step 3: Threat Correlation

- Correlate events over time and from multiple sources
- Detect patterns (e.g., lateral movement, privilege escalation)

PHASE 4: Dashboard Development

Step 1: Design UI/UX

- Define layout: Sidebar, Top Threats, Alerts, Map, Timeline, etc.
- Tools: Figma (for mockups), or directly in Streamlit/Dash

Step 2: Build Dashboard Features

- Top threat list
- Real-time threat map (geo-IP)
- Threat timeline (by severity/type)
- Search/filter logs
- User-based attack drill-down

Step 3: Add Alert System

- **Real-time alerts via:** - **[**Email (SMTP), Slack, Telegram, Discord bots]
- Integrate with webhook-compatible SIEM tools

PHASE 5: Testing & Optimization

Step 1: Test with Simulated Attacks

- Use tools like Metasploit, Kali Linux, or mock data
- Run penetration test logs or test malware sample

Step 2: Optimize Performance

- Use asynchronous log processing
- Add caching (Redis)
- Monitor performance (CPU, RAM, response time)

PHASE 6: Deployment & Maintenance

Step 1: Containerize with Docker

- Create [Dockerfile] and [docker-compose.yml]
- Include backend, frontend, and DB services

Step 2: Deploy on Cloud / VM

- Options: AWS EC2, Azure, GCP, or on-prem server
- Set up firewall rules, HTTPS, access control

Step 3: Monitor, Log & Maintain

- Enable logging for dashboard usage
- Monitor threat detection accuracy
- Regularly update threat feeds & models