Conception and Organization

1) Conception of the software solution

Overall Architecture

We used a **client–server model**:

- **Frontend**: Next.js + React + shadcn/ui.
    - Handles login/signup, sidebar navigation, analyze chat, history display, and dashboard charts.
- **Backend**: FastAPI (Python).
    - Provides authentication (/auth/register, /auth/login, /auth/me),
    - Analysis service (/analyze/),
    - History retrieval (/analyze/history),
    - Stats endpoints (/stats/*) to power dashboard charts.
- **Database**: PostgreSQL via SQLAlchemy.
    - Stores users, roles, analyses (message, result, timestamp).
- **LLM Integration**: AI analysis with structured output (judgment, explanation, tips). Used Gemini 2.0 flash for the project.
- **Threat Intelligence Modules**:
    - WHOIS check for domain age,
    - VirusTotal check for URLs,
    - Heuristics (keywords, suspicious URLs, sender anomalies).

The architecture ensures **separation of concerns**:

- UI and UX handled entirely on the frontend,
- Security, analysis, and persistence centralized in backend.

2) Task Breakdown and Order

Planned Order

1. **Setup project repos** (frontend + backend separately).
2. **Backend auth** (register/login, JWT-based token).
3. **Frontend auth pages** (Login, Signup, token storage).

4. **Sidebar + routing** in frontend.
5. **Analyze page** with dummy chat.
6. **Connect analyze page to backend** (real analysis).
7. **Save & display history**.
8. **Dashboard** with charts (line, bar, pie).
9. **Seed data** for testing dashboard stats.
10. **Polish UX**: dark mode toggle, avatar initials, etc.
11. **Security & privacy checks**.
12. **Documentation**

3) Tools for Organization

- **GitHub**: private repository, feature branches, frequent commits with meaningful messages.
- **GitHub Issues / Tasks**: tracked steps like “Implement JWT auth”, “Hook up analyze endpoint”, “Dashboard charts”.
- **Commits strategy**: step-by-step commits (auth → analyze → history → dashboard → seeding).

4) Feedback on Self-Organization

- I **iterated in small steps**: finishing auth completely before moving to analyze, then history, then dashboard. This avoided mixing too many concerns.
- I **tested frequently** using Postman (backend APIs) and manual interaction in frontend before moving forward.
- When errors arose (e.g., token decoding, pie chart only showing “Other”), I debugged carefully and applied **incremental fixes** (changing backend schema, updating frontend parsing).
- I **used commits as checkpoints**: each step ended with a working commit before moving on.
- Time management: I planned to finish backend core features first, then gradually integrate frontend visualization, ensuring the backend APIs always powered real data.