Final Project – Proposal Submission

Project: Monitoring a Containerized URL Shortener Webservice

Project Description:
This project focuses on developing and monitoring a fully containerized URL shortener webservice. The application will allow users to shorten long URLs and redirect them using short codes. The service will be instrumented with Prometheus metrics to monitor its performance and health. Grafana will be used to visualize these metrics and provide insights into usage trends. The entire stack, including the application and monitoring tools, will run locally using Docker and Docker Compose.

Group Members & Roles:

- **Osama Ashraf:** Team Leader / Monitoring Engineer (Prometheus)
- **Ahmed Sabry:** Developer Engineer (back-end)
- **Ebrahim Mohammed:** Development and Documentation (Front-end and GitHub)
- **Mohamed Elsayed:** Visualization (Grafana)
- **Ahmed Saeed:** Containerization (Docker and Docker-compose)

Team Leader:
Osama Ashraf

Objectives:

- Build and containerize a functional URL shortener webservice. •

  Implement custom Prometheus metrics to track performance.
- Use Grafana for real-time visualization and analysis.
- Configure alerting and enable data persistence.
- Deliver a stable, well-documented, and fully monitored stack.

Tools & Technologies:

- Python (Flask)
- SQLite
- Docker & Docker Compose
- Prometheus
- Grafana
- Git & GitHub

Milestones & Deadlines:

- **Week 1:** Build & Containerize the URL Shortener ○ Deliverables: Webservice code, Dockerfile, docker-compose.yml
- **Week 2:** Instrument with Prometheus Metrics
  - Deliverables: Updated code with /metrics endpoint, prometheus.yml, compose integration

- **Week 3:** Visualization with Grafana
  - Deliverables: Grafana dashboard, connected data source, full monitoring stack
- **Week 4:** Alerting, Persistence & Documentation
  - Deliverables: Volumes configuration, alert setup, complete documentation

---

KPIs (Key Performance Indicators):

1. **Infrastructure**
   - Proper setup of Docker and Docker Compose for containerized environments.
2. **Deployment**
   - Successful local deployment using Docker Compose.
3. **Code Integration & Testing**
   - Smooth Git integration and reliable local testing.
4. **Monitoring & Reliability**
   - Proper implementation of metrics, logging, alerts, and dashboards with high system uptime.