Field	Content
Purpose	Design and deploy a lightweight Linux-kernel monitoring agent that captures WaveServer AI CMD traffic and fiber-link health, shares it with user space through /proc + mmap, enriches it with AI-based anomaly detection and traffic forecasting, and streams the results to a cloud dashboard for real-time observability and auto-remediation.
Project Title	Smart Kernel-Based Monitoring Agent for Fiber-Optimized Optical Networks (SKMA-FON)
Project Manager	Soufian Carson
Project Team	• Soufian Carson — PM & Lead Developer• Teammate A — Kernel Module Developer• Teammate B — Cloud/API Engineer• Teammate C — AI/ML Engineer• Teammate D — Front-End & UX Designer
Start Date / End Date	07 Jun 2025 – 30 Aug 2025 (12 weeks)

---

2. Scope

Features to be delivered

- Linux kernel module that exports real-time CMD & fiber metrics via /proc/optifiber/myinfo.
- Shared-memory mapping using mmap() for zero-copy user-space access.
- User-space agent (Python) that polls the buffer, preprocesses data and pushes to the cloud.
- AI service (edge ONNX + cloud endpoint) for anomaly detection & 15-minute traffic forecasting.
- Cloud time-series database (InfluxDB Cloud) storing raw & AI-enriched metrics.
- Web dashboard (React + Chart.js) for live visualization, alerts and historical trends.
- Auto-remediation webhook that can call (or simulate) WaveServer MCP to provision extra CMD capacity when forecast > 90 % utilization.
- Complete DevOps pipeline (GitHub Actions, Dockerfiles, IaC script).

Out of scope

- Mobile app (UI)
- Hardware encryption module integration (future phase)
- Production deployment on live WaveServer hardware (lab simulation only)

---

3. Schedule

A. Work-Breakdown Table

Task ID	Task Description	Start	End	Responsible	Est. Hours	Progress
T1	Requirements & architecture workshop	06/07	06/09	Soufian C.	12	0 %
T2	Prototype kernel buffer & /proc entry	06/10	06/17	Teammate A	40	0 %
T3	Implement mmap() handler & unit tests	06/18	06/24	Teammate A	30	0 %
T4	Write Python agent: mmap reader + pre-processing	06/18	06/24	Soufian C.	25	0 %
T5	Build edge AI (ONNX) & cloud AI endpoint (SageMaker)	06/25	07/05	Teammate C	45	0 %
T6	Cloud ingestion API & InfluxDB Cloud setup	06/25	07/03	Teammate B	28	0 %
T7	Front-end dashboard (React)	07/04	07/17	Teammate D	45	0 %
T8	Alert engine & auto-remediation webhook	07/18	07/24	Teammate B	20	0 %
T9	Integration testing (kernel ↔ agent ↔ cloud ↔ UI)	07/25	08/07	Whole team	50	0 %
T10	Documentation, user guide & training video	08/08	08/18	Soufian C.	24	0 %
T11	Final demo, slide deck & retrospection	08/19	08/30	Whole team	20	0 %

A simple Gantt chart can be generated in Excel with the above dates.

---

4. Team Organization

A. Roles

- **Project Manager (Soufian)** — timeline, risks, stakeholder comms.
- **Kernel Developer (Teammate A)** — C coding, kernel APIs, /proc, mmap.
- **Cloud/API Engineer (Teammate B)** — REST API, DB schema, alert micro-service.
- **AI/ML Engineer (Teammate C)** — dataset prep, model training, ONNX conversion.
- **Front-End Designer (Teammate D)** — React UI, Grafana theme, UX testing.

B. Member profiles

(Attach résumés in appendix; each one-paragraph bio describes experience with Linux, AWS, React, etc.)

---

5. Requirements Documentation

A. Functional Requirements

1. System must collect CMD throughput (Gbps), error counters, and link status every second.
2. Agent must expose an HTTP endpoint /metrics returning latest JSON packet.
3. System shall flag an anomaly when anomaly_score ≥ 0.8.
4. Dashboard must refresh visuals ≤ 2 seconds after metric arrival.
5. Webhook shall POST to MCP API when forecast utilization > 90 %.

Non-Functional Requirements

- Kernel overhead < 2 % CPU on Intel i5-8250U.
- End-to-end latency from kernel update to dashboard render ≤ 3 s.
- TLS 1.2 for all cloud traffic.
- System uptime target 99.5 % during demo week.

B. Specifications (diagrams)

- **Use-case diagram**: Roles (Agent, Dashboard, AI, Auto-remediator).
- **Flow of events**:

1 — Deployment & Boot

#	Event	Main Actors	Notes
1	**DevOps pipeline** builds monitoring_module.ko, user-space agent, and container images.	GitHub Actions / Docker	Images include Python agent + lightweight AI client libraries.
2	**Edge host (WaveServer-adjacent Linux box)** pulls latest container.	Containerd / systemd	The container starts automatically on boot.

---

2 — Kernel-Space Setup

#	Event	Main Actors	Notes
3	Container entry-point runs insmod monitoring_module.ko.	Host OS	Requires CAP_SYS_MODULE or baked-in module.

#	Event	Main Actors	Notes
4	init_module() allocates one page per site (e.g., 4 × 4 KB).	Kernel C code	Each page holds **struct site_stats** (throughput, errors, BER).
5	Module populates /proc/optifiber/myinfo and registers .mmap callback.	Kernel	Buffer marked reserved with SetPageReserved().

3 — User-Space Data Access

#	Event	Main Actors	Notes
6	Python/Go **agent** opens /proc/optifiber/myinfo (O_RDWR).	Agent process	Runs under the same container.
7	Agent calls mmap(), receiving a pointer to the shared pages.	Agent ↔ kernel	Zero-copy: no read() calls needed.
8	A lightweight **polling loop** (e.g., every 1 s) converts raw bytes into JSON dicts.	Agent	Example payload: {"site":"Dallas","throughput":1570,"errors":2}.

---

4 — Local Pre-Processing & AI Inference

#	Event	Main Actors	Notes
9	Agent runs **on-device feature extraction** (traffic delta, moving average).	NumPy / Pandas	Keeps packet rate, error trend, utilization %.
10	Pre-processed batch is sent to an **AI inference endpoint** (REST/gRPC).	HTTP/HTTPS	Two deployment options:
10a	**Edge AI**: a tiny ONNX model (Isolation Forest or LSTM) shipped in the container for offline/low-latency inference.	ONNX-Runtime	Works even when WAN is down.
10b	**Cloud AI**: send to managed service (e.g., AWS SageMaker Endpoint, Vertex AI, or an OpenAI function calling your fine-tuned model).	TLS	High accuracy, central training.
11	AI model returns: { "anomaly_score": 0.91, "forecast_next_15min_gbps": 1800 }.	Model	Threshold > 0.8 triggers alert.

5 — Cloud Ingestion & Persistence

#	Event	Main Actors	Notes
12	Agent pushes raw & AI-enriched metrics to cloud **time-series DB**.	InfluxDB Cloud / Timestream / Firebase	Retention policy 30 days.

#	Event	Main Actors	Notes
13	Metrics also streamed to **Kafka / Kinesis** for real-time pipelines.	Optional	Enables multiple consumers (dashboards, alert engine).

---

6 — Visualization & Alerting

#	Event	Main Actors	Notes
14	**Grafana / React dashboard** subscribes to DB or WebSocket and renders CMD utilization, forecasts, and anomaly heat-map.	Web	Chart updates every few seconds.
15	If anomaly score high or forecast > 90 % capacity, **alert micro-service** triggers:	Lambda / Cloud Function	
15a	Slack / Teams / email notification to NOC.	Twilio / SendGrid	
15b	Optional **auto-remediation** webhook: tells WaveServer MCP API to pre-provision an extra CMD module or shift traffic.	MCP north-bound API	

7 — Feedback to Kernel (Optional Closed Loop)

#	Event	Main Actors	Notes
16	Cloud decision engine posts a **config command** to the agent (MQTT / REST).	Agent	Ex: {"site":"Dallas","cmd_add":1}
17	Agent writes new config to /proc/optifiber/cmd_control (another proc entry).	write_proc()	Kernel adjusts its simulation parameters (or real driver in prod).
18	Kernel buffer now reports updated stats, loop continues from step 6.	Continuous loop	Demonstrates self-healing / auto-scaling.

8 — Shutdown & Cleanup

#	Event	Main Actors	Notes
19	DevOps issues docker stop or host reboots.	Host	
20	Container pre-stop hook runs rmmod monitoring_module.	systemd / Docker	
21	cleanup_module() frees pages, clears reservations, removes /proc entries.	Kernel	Ensures no memory leaks.

Adding AI: Practical Implementation Tips

Component	Minimal Viable Option	Production-Ready Option
Model Type	Isolation Forest for anomaly; simple ARIMA for forecast (scikit-learn on device).	LSTM/CNN trained in SageMaker; batch retraining daily; served via real-time endpoint.
Data Pipeline	Agent sends JSON over HTTPS to Firebase / Supabase.	Kafka → Flink → InfluxDB Cloud → Grafana Loki.
Edge vs Cloud	Ship ONNX model (few MB) in the agent container.	Hybrid: quick edge inference, cloud for heavy retraining / global view.
Security	Signed JWT per host; TLS to API.	AWS IAM roles, private VPC endpoints, mutual TLS, audit logs.

- **Class/struct diagram**: show struct site_stats, Python MetricPacket, React StateStore.
- **ER diagram**: Influx schema (measurement = site_stats, tags = site, fields = metrics).
- **Decision table**: If anomaly & forecast thresholds trigger actions.

(todo: PDFs/images in template appendix.)

---

6. System Design

A. Conceptual Design (summary)

The kernel module periodically samples (or simulates) optical metrics and stores them in a reserved page. A user-space Python agent maps that page, converts bytes to structured JSON, performs local feature extraction and calls either an embedded ONNX model or a cloud SageMaker endpoint for anomaly detection and short-term forecasting. Enriched metrics are sent via REST to a Flask API fronting InfluxDB Cloud. A React dashboard subscribes to WebSocket updates for real-time visualization. An alert micro-service monitors the DB; when thresholds are breached it notifies Slack and can POST to a (simulated) WaveServer MCP endpoint to pre-provision additional CMD modules.

B. Report Formats

- Daily CSV export of site_stats.
- Weekly PDF capacity report (auto-generated by Python script).

C. Screen Layouts

- **Login / Swagger** page for API key.
- **Live Dashboard** with four gauges (one per site) and anomaly heat-map.
- **Settings** page to adjust alert thresholds and webhook URL.

D. Technical Design

- **Kernel**: C, Linux 6.x, procfs, remap_pfn_range.
- **Agent**: Python 3.12, numpy, onnxruntime, requests, Docker Alpine base.
- **Backend**: Flask 2, InfluxDB Cloud, AWS Lambda alert engine.
- **Frontend**: React 19, Vite, Chart.js.
- **CI/CD**: GitHub Actions, Docker Hub, Terraform for AWS infra.

E. Database Design

Measurement site_stats

Field	Type	Description
time	timestamp	influx auto field
site	tag	MicrosoftDC, Dallas, Dobbins, Stone
throughput_gbps	float	Current traffic
error_count	int	CRC/FEC errors
anomaly_score	float	0–1
forecast_gbps	float	15-min prediction

7. Technical Description

A. Key Interfaces & Modules

Module	Function	User	Special Notes
/proc/optifiber/myinfo	Raw shared page	Kernel ↔ Agent	4 KB per site
Python Agent REST /metrics	Current JSON snapshot	Dashboard, Alert svc	JSON schema v1
AI Endpoint /predict	Returns anomaly score & forecast	Agent	JWT auth
Alert Webhook	Sends {"site":x,"cmd_add":1}	MCP (simulated)	ISO 8601 timestamps

B. HW/SW Requirements

- Ubuntu 22.04 LTS VM, 2 vCPU, 4 GB RAM.
- Docker 24.0+, compose V2.
- AWS free-tier account (Lambda, IAM, InfluxDB Cloud).

C. Role/Permission Matrix

Role	View Dashboard	Edit Thresholds	Load Kernel	Call MCP
Admin	✓	✓	✓	✓
NOC User	✓	✗	✗	✗
DevOps	✓	✓	✓	✗

---

8. Data Management Plan

- **Data collected**: throughput, error count, anomaly score, forecast.
- **Access**: API keys scoped per role; IAM roles for cloud resources.
- **Protection**: TLS 1.2+, at-rest encryption via InfluxDB Cloud.
- **Backups**: Daily export to S3 lifecycle bucket (30-day retention).
- **Privacy**: No PII; metrics only.
- **Disaster recovery**: Terraform script can redeploy infra in < 30 min.

---

9. Test Plan

A. Approach

- **Unit tests**: Kernel functions mocked with KUnit; Python pytest.

- **Integration**: Docker-Compose stack bringing up kernel-enabled container + API + dashboard.
- **User Acceptance**: Simulated NOC users validate alerts.
- **Performance**: Use stress-ng to ensure CPU < 2 %.
- **Security**: OWASP ZAP scan on REST API.

B. Completion Criteria

- 100 % pass of critical unit tests.
- No Sev-1 or Sev-2 bugs open.
- Dashboard latency ≤ 3 s with 1 k msg/s load.

C. User Support

- Markdown user guide in repo.
- Video demo (5 min) on YouTube (unlisted).
- Contact email: support@skma-fon.dev.

---

10. Technical Support Plan

- **Training**: 1-hour Zoom workshop; slides + lab instructions.
- **Installation**: One-command make deploy (Terraform + Docker).
- **Ongoing Support**: GitHub Issues; SLA 48 h response.
- **Updates**: Semantic versioning; monthly releases.
- **Troubleshooting**: FAQ in README—covers dmesg, rmmod, TLS errors.
- **Support Hours**: 9 am – 6 pm EST; contact 404-555-0123.

Appendix

- Diagrams (use-case, class, ER, deployment).
- Team resumes.
- Sample Grafana dashboard screenshot.
- Slide deck for final presentation.
