Portable Telemetry with real-time performance insights

Telemetry

Telemetry is the process of gathering the performance data of any product and communicating it to a remote location for monitoring and analysis. This technique is commonly used to keep track of the performance of various products with ease.

In a computer system, Software agents or instrumentation embedded in systems collect real-time performance and usage data in terms of metrics & logs.

Task Manager & htop

Traditional methods to measure your system load use softwares such as Task Manager (for windows) or libraries and functions such as htop/top (for linux).

These methods are os dependent and may or may not show relevant information unless the user has had relevant experience dealing with systems.

Technologies Used

Core Engine Layer

- ➔ Python
- ➔ Psutil

Reads CPU, memory, disk, network, load, processes.

Produces snapshots.

Terminal UI Layer (v1.0)

- ➔ Textual
- ➔ Rich

Reactive terminal layout.
Bars, panels, sparkline.
Keyboard interaction.

Web API Layer

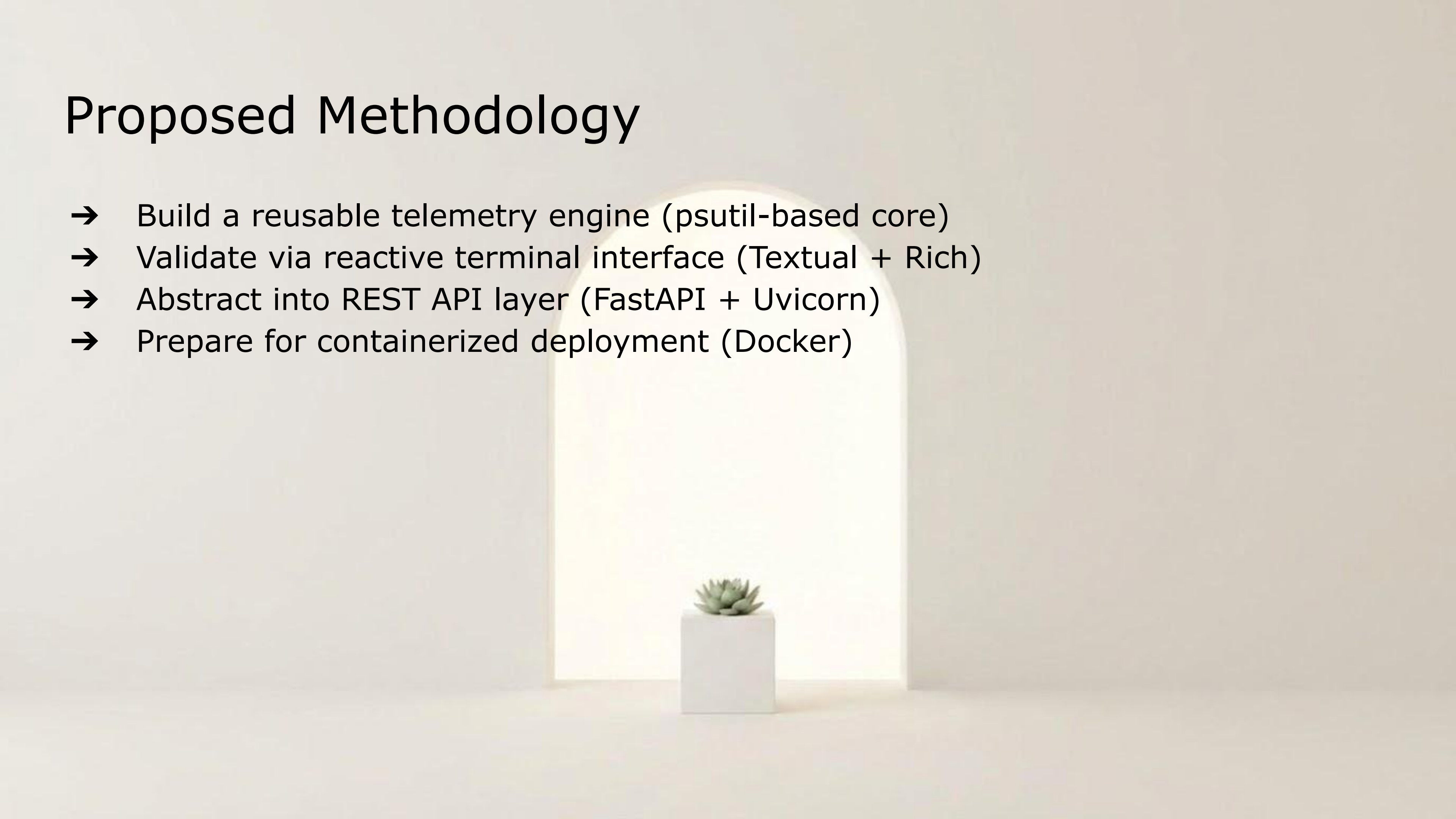
- ➔ FastAPI
- ➔ Uvicorn (ASGI server)

HTTP routing.
JSON serialization.
Request handling.
Network port listening.

Containerization Layer

- ➔ Docker

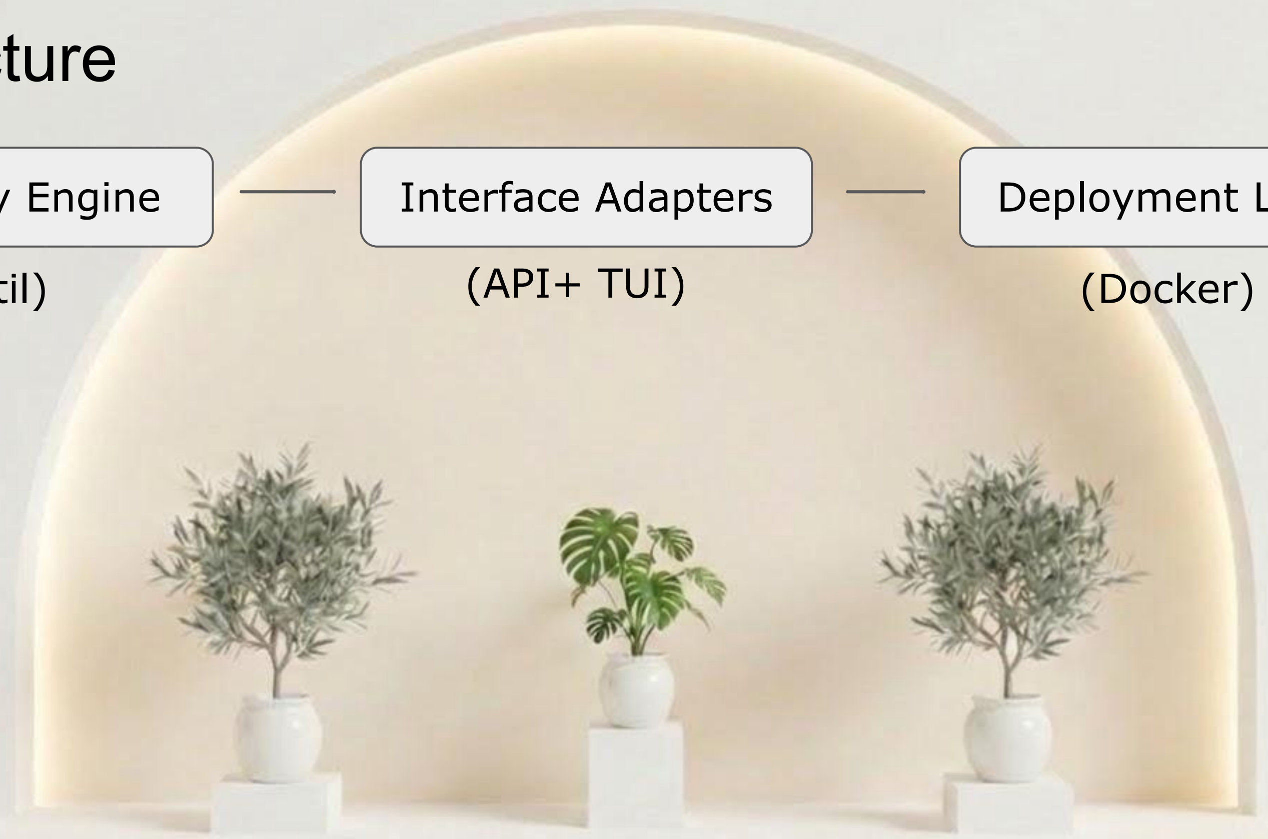
Packages runtime.
Installs dependencies.
Exposes port 8000.
Runs uvicorn inside container.



Proposed Methodology

- ➔ Build a reusable telemetry engine (psutil-based core)
- ➔ Validate via reactive terminal interface (Textual + Rich)
- ➔ Abstract into REST API layer (FastAPI + Uvicorn)
- ➔ Prepare for containerized deployment (Docker)

Architecture



$$\begin{tikzcd}\boxed{\text{Telemetry Engine}} \arrow[r] & \boxed{\text{Interface Adapters}} \arrow[r] & \boxed{\text{Deployment Layer}} \\ \text{(psutil)} & \text{(API+ TUI)} & \text{(Docker)}\end{tikzcd}$$

Scope

Web Dashboard Expansion

- ➔ Real-time WebSocket streaming
- ➔ Interactive charts (CPU, network, disk history)
- ➔ Multi-device access via browser

Advanced Process Control

- ➔ Process filtering & search
- ➔ Resource throttling insights
- ➔ Safe process termination controls

Deployment & Portability

- ➔ Dockerized runtime environment
- ➔ Cross-platform execution
- ➔ Cloud or self-hosted deployment
- ➔ Multi-node monitoring extension

Conclusion

- ➔ Design a layered, extensible system monitoring architecture.
- ➔ Separate telemetry engine from interface layers.
- ➔ Deliver both terminal and web-ready service interfaces.
- ➔ Introduce time-series visibility and interactive controls.
- ➔ Enable portable deployment through containerization.

End-user outcomes

- ➔ Improved real-time observability through structured telemetry.
- ➔ Enhanced user experience via reactive UI and interaction.
- ➔ Achieved portability through container-ready architecture.
- ➔ Established a scalable foundation for future monitoring services.