YET ANOTHER METRICS COLLECTOR (YAMeC)

Marcus DiMarco and Brendan Gibbons

MSCS 710 – System Project

Professor Giorgio

March 2, 2025

PROJECT **OVERVIEW**

YAMeC is a metrics collection and archival application. It's designed to:

- Collect system metrics at both the system and per-process levels
  - (CPU usage, GPU usage, RAM usage, NIC usage, Disk usage)
- Insert those metrics into a local database, maintaining multiple levels of granularity
- Display those metrics through a user-friendly web interface
- Provide system administrators and power users with historical insight into system utilization

YAMeC **COMPONENTS**

SPRING BOOT CONTROLLER
01

Contains main backend logic and controllers for all other components

JNI BRIDGE
02

Allows Java code to invoke C++ functions via intermediary functions

C++ SYSTEM CALL MODULE
03

Contains various system calls which collect metrics from the OS

SQLITE DATABASE
04

Records the various metrics and devices reported by the OS

LOCAL WEB APP
05

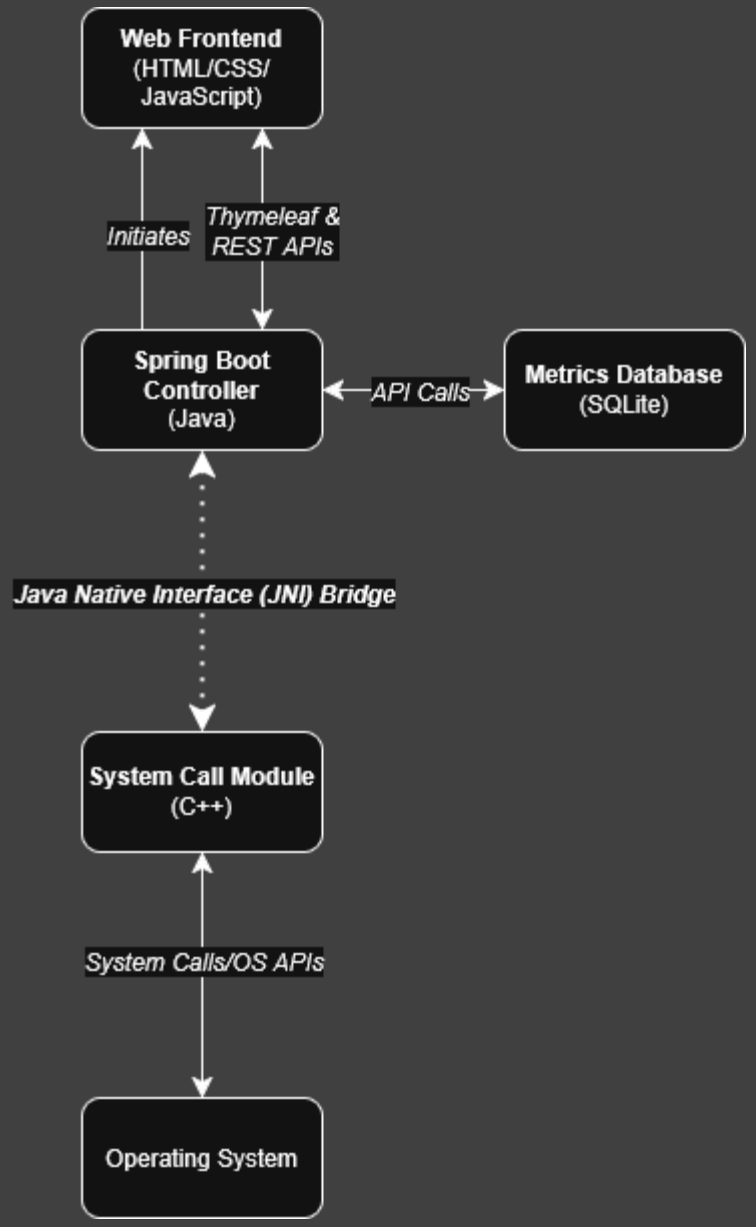
Local UI for viewing both current and historical data

JAVA
Chosen for the development team's familiarity in the language

C++ AND JNI
Provide direct access to OS system calls while minimizing overhead in learning new languages

SQLITE
Offers a lightweight local database to minimize overhead on user systems

SYSTEM ARCHITECTURE
OVERVIEW

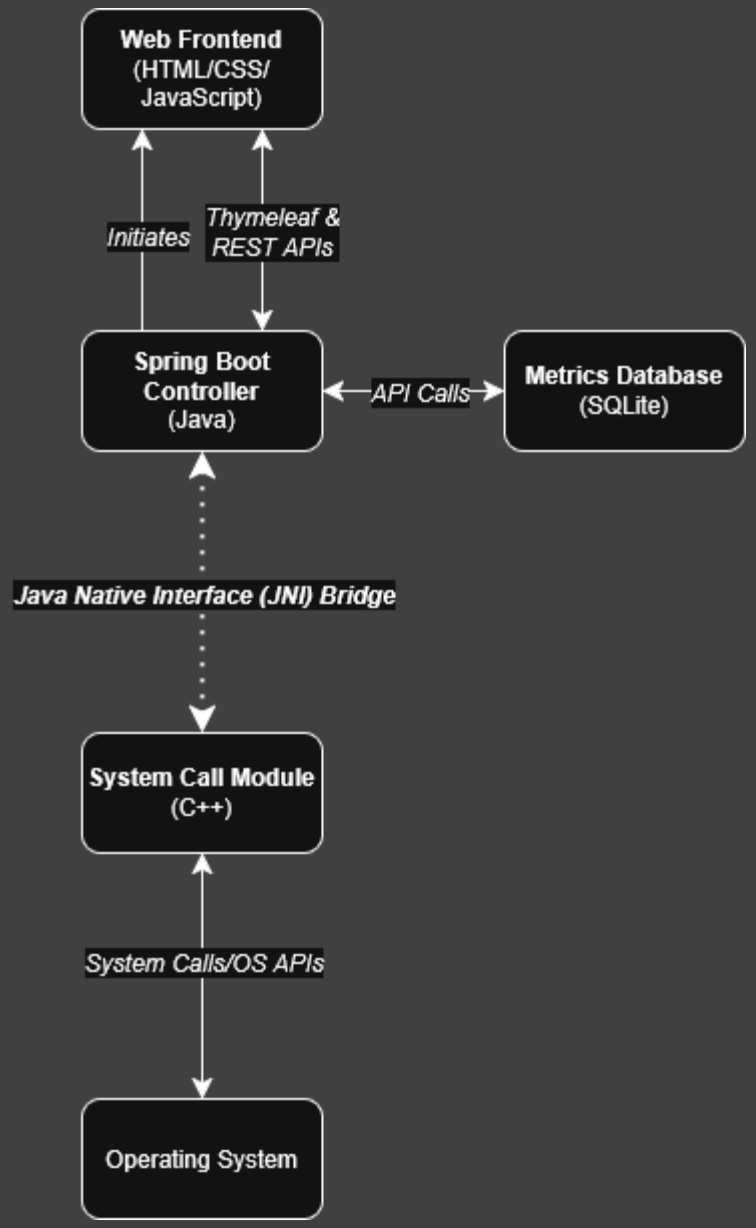


SPRING BOOT CONTROLLER

Module: yamec-app

Responsibilities:

- Web Application Hosting
- REST API Endpoint
- Data Presentation Logic
- JNI Interaction
- Error Handling

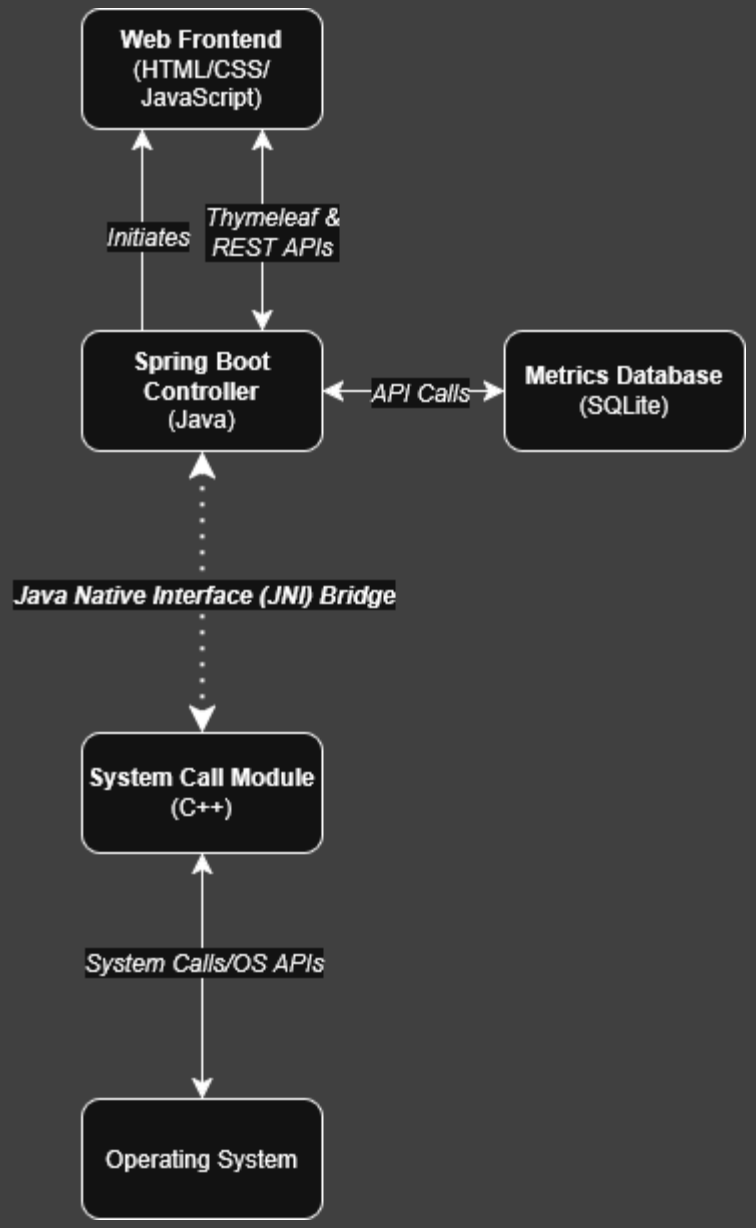


JNI & C++
SYSTEM CALL MODULE

Module: yamec-jni

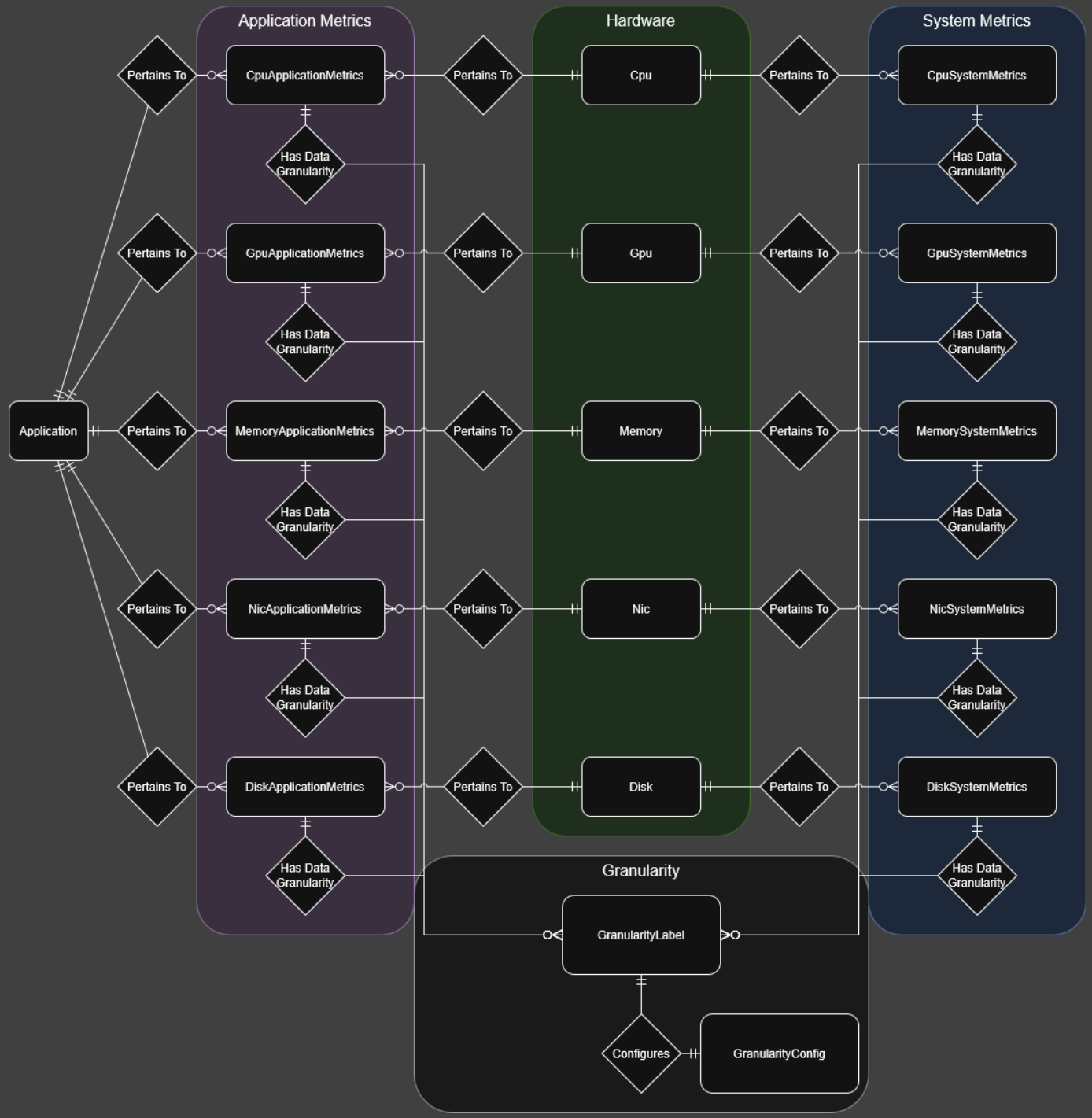
Responsibilities:

- System Metric Collection
- JNI Function Implementation
- Data Conversion
- Native Library Building



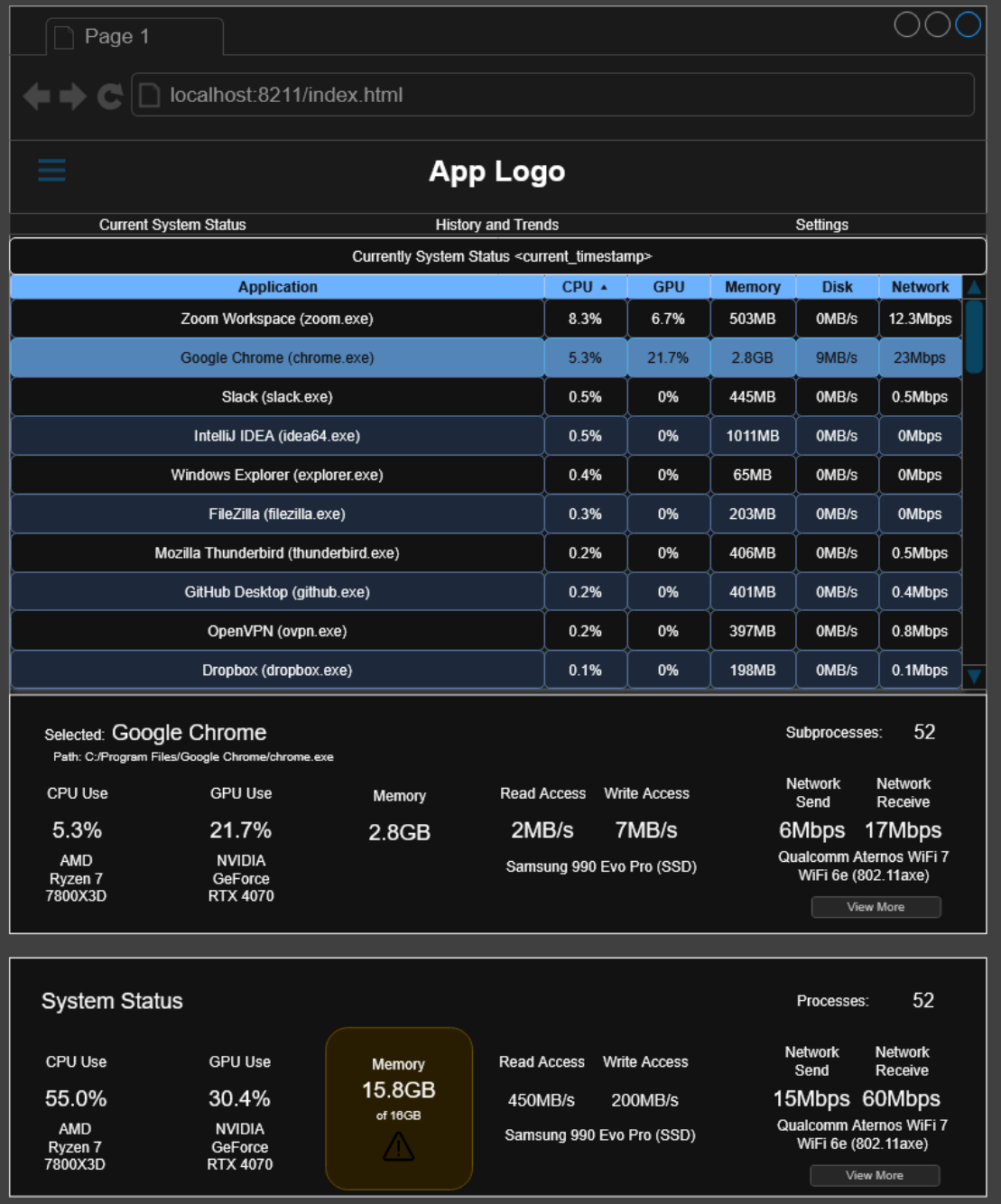
DATABASE **DESIGN**

- Metrics are saved per application and per hardware device
- Timestamps and duration are recorded to provide the system's status over time
- GranularityLabel: Granularity levels differentiate metrics of varying time precision
- GranularityConfig: User can configure the level of time precision (RecordTimespan) and when metrics' precision is adjusted (TimeToAge)



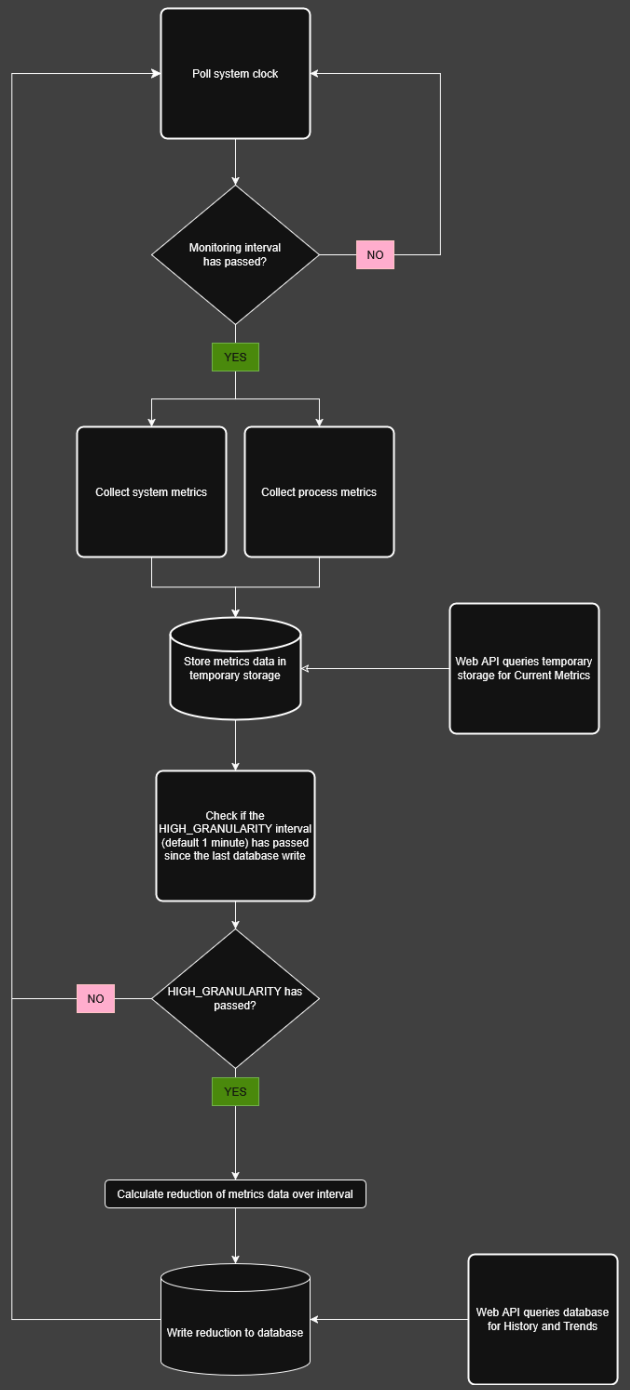
WEB **FRONTEND**

- Applications are shown in a list-based interface
- Current System Status auto-updates and has the highest precision of data
- History and Trends shows average, maximum, and minimum statistics over the selected timespan, as well as a graph view of utilization (not shown)
- Potential issues are indicated with color-coding and icons or dotted lines (graphs) for accessibility



Mockup designed for illustration purposes. It may not be fully accurate to the final product.

PROCESS **FLOW**



PROJECT **MILESTONES**

PHASE 01

C++ system call module, initial Spring Boot application

PHASE 02

Spring Boot application complete, integrated with C++ system call module via JNI bridge

PHASE 03

Web app & REST APIs complete

PHASE 04

System and integration testing, build & deployment

THANK **YOU**