Development Document

1.0.0-20250314

Table of Contents

[Table of Contents 1](#_Toc163479875)

[1 Introduction 3](#_Toc163479876)

[1.1 Project Background 3](#_Toc163479877)

[1.2 Project Objectives 3](#_Toc163479878)

[1.3 Audience 3](#_Toc163479879)

[1.4 Document Summary 3](#_Toc163479880)

[2 Project Overview 3](#_Toc163479881)

[2.1 Project Name 3](#_Toc163479882)

[2.2 Project Description 3](#_Toc163479883)

[2.3 Target Audience 3](#_Toc163479884)

[2.4 Project Scope 3](#_Toc163479885)

[3 Demand Analysis 3](#_Toc163479886)

[3.1 User Requirements 3](#_Toc163479887)

[3.2 System Requirements 3](#_Toc163479888)

[3.3 Use Case Analysis 3](#_Toc163479889)

[3.4 Data Model 3](#_Toc163479890)

[4 Design 3](#_Toc163479891)

[4.1 Architecture Design 3](#_Toc163479892)

[4.2 Database Design 3](#_Toc163479893)

[4.3 Interface Design 3](#_Toc163479894)

[4.4 UI Design 4](#_Toc163479895)

[4.5 Flowchart 4](#_Toc163479896)

[5 Develop 4](#_Toc163479897)

[5.1 Development Environment 4](#_Toc163479898)

[5.2 Coding Standards 4](#_Toc163479899)

[5.3 Module Design 4](#_Toc163479900)

[5.4 Test Plan 4](#_Toc163479901)

[6 Test 4](#_Toc163479902)

[6.1 Unit Testing 4](#_Toc163479903)

[6.2 Integration Testing 4](#_Toc163479904)

[6.3 System Testing 4](#_Toc163479905)

[6.4 Performance Testing 4](#_Toc163479906)

[7 Deployment and Maintenance 4](#_Toc163479907)

[7.1 Deployment Plan 4](#_Toc163479908)

[7.2 User Manual 4](#_Toc163479909)

[7.3 Operation and Maintenance Documents 4](#_Toc163479910)

[7.4 Troubleshooting 4](#_Toc163479911)

[8 Summary and Evaluation 4](#_Toc163479912)

[8.1 Project Summary 4](#_Toc163479913)

[8.2 Project Evaluation 5](#_Toc163479914)

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Name** | **Reason For Changes** | **Date** |
| 1.0.0 | Sven.tan | Initial Revision | 2025/03/14 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Approved By

Approvals should be obtained for project manager, and all developers working on the project.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Signature** | **Department** | **Date** |
|  |  |  |  |
|  |  |  |  |

# Introduction

## Project Background

Do not have a secure way of monitoring the activities running on the user devices as well as a transparent manner to check the system data from a centralized endpoint. A interactive self-developed dashboard would allow for faster and more efficient trouble shooting.

## Project Objectives

1. Collecting IT Server Logs

2. Analyzing Log Data

3. Generating Reports and Findings

4. Automation and Ongoing Monitoring;

## Audience

**System Administrators** – Monitor VMs, respond to alerts.

**Engineers** – Using of the vms to run engineering tasks.

**Developers** – Extend the system, debug issues.

## Document Summary

This document details the **architecture, database design, API endpoints, and deployment process** of the Next.js monitoring dashboard to allow for the understanding and use of how to push the use of the prototype to deployment.

# Project Overview

## Project Name

VM Monitoring Dashboard

## Project Description

The VM Monitoring Dashboard is a **Next.js-based web application** that integrates with **PostgreSQL via Prisma ORM** to monitor Virtual Machines (VMs) running on a host system. The dashboard provides real-time insights into VM status and serves as an **Intrusion Detection System (IDS)**, alerting administrators in case of anomalies.

## Target Audience

YWL engineers that requires the use of licensed software

## Project Scope

Dashboard and test vms

# Demand Analysis

## User Requirements

Can sort and filer through logs

## System Requirements

Easy to maintain and manage, secure and performant

## Use Case Analysis

Users use the machine, machine reports to postgresql database. Dashboard queries the database, Dashboard sends reports to the admin

## Data Model

Syslogs, Auth logs, System metrics, Memory usage, Devices Information, Notes for deployment and other details

# Design

## Architecture Design

Nextjs as a full stack framework (MIT)

Prisma as an orm ( Apache 2.0)

Ubuntu as a testing VM (<https://ubuntu.com/legal/open-source-licences> )

Postgresql ([**PostgreSQL License**](https://www.opensource.org/licenses/postgresql))

Npm Xlsx (Apache2.0)

Npm Sonner, RadixUI, tailwind, Recharts, React (MIT)

Nodejs (<https://github.com/nodejs/node/blob/main/LICENSE> )

Python and Pip packages ( Open BSD)

## Database Design

model logs {

id Int @id @default(autoincrement())  
name String  
host String?  
timestamp DateTime @default(now())  
piuser String?  
pid Int?  
action String?  
cpu Float?  
mem Float?  
command String?  
port Int?  
ipAddress String?

}

For logging of general system information, login, logout ad pid activities

model auth {

id Int @id @default(autoincrement())  
timestamp DateTime @default(now())  
username String  
log\_entry String

}

Logging only based on the auth logs in /var/log/auth.log stored with /etc/systemd/system/auth\_log\_monitor.service

model memory\_usage{

id Int @id @default(autoincrement())  
total\_memory BigInt  
used\_memory BigInt  
free\_memory BigInt  
available\_memory BigInt  
percent\_usage Float  
host String?  
time DateTime @default(now())

}

Total memory usage used for the devices stored with psutils

model system\_metrics {

id Int @id @default(autoincrement())  
timestamp DateTime @default(now())  
sensor\_name String  
value\_type String  
value Float  
host String

}

Logged using sudo apt install lm-sensors

model notes{

id Int @id @default(autoincrement())  
title String  
time DateTime @default(now())  
description String

}

For storing of notes related to deployment

model devices{

id Int @id @default(autoincrement())  
name String  
ip\_address String?  
mac\_address String?|  
password String?  
time DateTime @default(now())  
notes String

}

Storing of devices information

## Interface Design

DATABASE\_URL="postgresql://admin:host-machine@192.168.1.26:5432/logs\_database?schema=logs"

The devcies interfaces through a bash script or python to the postgresql database and prisma ensure the readability and type safety of the frontend. Ubuntu uses psutils or im-sensors to read system information of system.

## UI Design

Navigation Tabs: For syslog, auth log and devices

VM Overview: Table with sortable columns (Name, CPU, Memory, Status).

VM Detail View: Graphs for CPU, Memory, and Disk usage.

Alerts Panel: List of active alerts with severity indicators.

## Flowchart

Example flow of a persistent login

Gnome detects wrong login ->  
Sends to Ubuntu auth.log ->  
Auth.py detects through Ubuntu Daemon ->  
Daemon runs auth.py ->  
Auth.py sends postgres insert command to 192.168.1.26:5432 ->  
Prisma queries database for auth logs ->  
Data sent back to auth-logs-table->  
Nextjs populates the data according to the ui defined table ->  
Nextjs checks for rule related to persistent login ->  
Sends api request to smtp server ->  
Email sent from server to sysadmin ->  
Allows for investigation efforts through export of excel file and dashboard search tools

# Develop

## Development Environment

Can explore using git as a version control manager for code base. To change the ip address of the postgresql to the device and port it is hosted.  
Nodejs and pnpm environment, preferably have access to the file system to store node packages and generation of types for prisma.

pnpm install (install node modules)

pnpm l (introspect database -> remember to pnpm g after)

pnpm g (generate types)

pnpm S (prisma studio -> remember to pnpm g before)

pnpm p (migrate database -> try to use ? in data type to avoid dropping table)

pnpm dev (start server)

## Coding Standards

<use server / client>  
<import statements>  
<usestate>  
<useeffect>  
<ts functions>  
<react component>  
<dialogs or modals>

## Module Design

All reusable ui components in /ui folder

All api calls in /api

All public images and assets in /public

Layout -> Page -> Chart /table -> Actions

## Test Plan

Yet to write

# Test

## Unit Testing

Will do one for pg\_dump

## Integration Testing

Not sure, only for Ubuntu so far, haven’t figured out how to do for windows or other os

## System Testing

Eslint and typechecking with prisma, by default nextjs will do errors if there are failed areas that cant compile.

## Performance Testing

Only 30k logs so far, will have to see if it can scale to more devices under load. Average of 20logs a minute, yet to come up with a reliable infra.

# Deployment and Maintenance

## Deployment Plan

Do on prototype and slowly do it with actual users before pushing for deployment. Prod and Staging CI development might be required for seamless failover, could use a pooler based approach for database

## User Manual

Internal notes system, code comments as well as read me file. Feel free to update this document as well.

## Operation and Maintenance Documents

Not sure

## Troubleshooting

Can have a multi webserver and db approach that does a master and slave system when trouble shooting, if server crashes, restore from pg\_dump file.

# Summary and Evaluation

## Project Summary

This is still in prototype, needs team review and evaluation.

## Project Evaluation

Yet to receive windows key or engineering licence.