System Planning: Project Charter

System Monitoring Server

Henry Zektser, Zemenu Guanche, Jorge Castro, Martha Agbugui

Prof. Michael Walsh

9/17/2023

Table of Contents

[Problem Statement 3](#_Toc145580322)

[Technology Solution 3](#_Toc145580323)

[Project Benefits 3](#_Toc145580324)

[Stakeholders 3](#_Toc145580325)

[Development Process 4](#_Toc145580326)

[High-Level Timeline 5](#_Toc145580327)

[Technology Tools 6](#_Toc145580328)

[Front-end Technology 6](#_Toc145580329)

[Back-end Technology 6](#_Toc145580330)

[Database and Data Query Language 7](#_Toc145580331)

[Containerization and Development Environment 7](#_Toc145580332)

[Hosting and Cloud Infrastructure 8](#_Toc145580333)

[Current Solutions 8](#_Toc145580334)

[References 11](#_Toc145580335)

# Problem Statement

Inefficient system monitoring, disjointed incident management, and inadequate communication during downtime undermine the operational resilience and customer satisfaction (Team, 2022).

# Technology Solution

The proposed solution is a comprehensive systems monitoring server that combines monitoring, alerting, and user visibility in a unified platform. Specifically, the system will allow users to define servers to monitor by providing an IP or an http(s) URL, and a list of users’ email addresses to alert in the event of issues. The system will continually monitor all IPs via ping and URLs via curl, and email alerts in the event of invalid status codes. A status page, showing a list of monitored servers, their current pass/fail state, and any recent issues will also be provided.

# Project Benefits

* Enables alerts of server downtime to be received within 5 minutes of an adverse event.
* Enables a 30-day historical view of all events.
* Enables a single-page view of all monitored systems and current status.

# Stakeholders

This business problem affects a wide spectrum of stakeholders across the organization, including:

* **DevOps Teams:** DevOps teams encounter difficulties in identifying issues early due to a lack of real-time insights, leading to extended downtime and increased operational costs. The technological factor is the primary influencer for this group.
* **IT Administrators:** IT administrators grapple with manual incident management processes, causing delays in response and resolution times, adversely affecting system performance and user experience. The technological factor is the primary influencer for this group.
* **Support Engineers:** Support engineers face erosion of customer trust when they fail to provide timely and transparent updates about incidents, potentially leading to customer churn (Digital, 2023). The technological factor is the primary influencer for this group.

# Development Process

The Software Development Life Cycle (SDLC) for this project will adopt a hybrid approach, combining the Waterfall method for the Planning, Analysis, and Design Phase with the standard Agile Scrum method for the Implementation and Support Phase. This hybrid approach aims to leverage the structured planning and documentation benefits of Waterfall in the early stages while embracing the flexibility and iterative development practices of Agile Scrum in the later phases.

**Planning, Analysis, and Design Phase (Waterfall Method):** During this phase, a comprehensive project plan will be created outlining the project scope, objectives, and requirements. Extensive research, analysis, and documentation will take place to define the software's architecture and design specifications. Reviews and sign-offs will be conducted at key milestones to ensure alignment with the project's goals. This phase will follow a sequential approach, with clear timelines and deliverables.

**Implementation and Support Phase (Agile Scrum Method):** Once the design phase is complete, the project will transition to Agile Scrum for implementation and support. The project will be divided into sprints with a length of one week. Daily stand-up meetings are not practical given team commitments, so a daily “virtual standup” will be conducted via Slack to facilitate communication and address any challenges. Backlog grooming sessions will be held weekly to refine and prioritize the product backlog, ensuring that the most critical features are addressed first. Retrospectives will be conducted at the end of each sprint to evaluate progress, identify areas for improvement, and adapt the development strategy accordingly.

By integrating the Waterfall and Agile Scrum methodologies in this SDLC, the project aims to combine the strengths of both approaches. The early stages benefit from the structured planning and design of Waterfall, while the later stages embrace the agility, collaboration, and adaptability of Agile Scrum. This hybrid approach ensures a thorough understanding of requirements and design before transitioning into an iterative development process that can respond effectively to changing needs and priorities.

# High-Level Timeline

|  |  |  |
| --- | --- | --- |
| Milestone | Start Date | End Date |
| Systems Planning | 8/21/23 | 8/28/23 |
| Systems Analysis | 9/1/23 | 9/29/23 |
| Systems Design | 9/29/23 | 10/31/23 |
| Documentation | 10/31/23 | 12/1/23 |
| Systems Implementation | TBD (IFT402) | TBD (IFT402) |

# Technology Tools

This project leverages a number of open-source technologies across the tech stack in order to accelerate development timelines:

## Front-end Technology

1. **TypeScript**: TypeScript was chosen for its ability to add strong typing to JavaScript, catching errors at compile-time rather than runtime. This reduces the likelihood of bugs, enhances code quality, and facilitates collaboration among developers (*JavaScript With Syntax for Types.*, n.d.).
2. **React**: React is a popular and well-supported library for building user interfaces. Its component-based architecture promotes reusability and modularity, allowing for efficient development and maintenance of complex UIs. Additionally, React's virtual DOM improves performance by minimizing unnecessary re-rendering.
3. **Patternfly:** Patternfly is a UI framework that provides out-of-the-box components for enterprise application interfaces. It’s tight integration with TypeScript and React enables engineers to quickly assemble attractive UIs without significant investment in basic web development (*PatternFly • About PatternFly*, n.d.).

## Back-end Technology

1. **Node.js**: Node.js is a server-side JavaScript runtime that offers non-blocking I/O and a single-threaded event loop, making it highly efficient for handling concurrent connections. Its ecosystem is rich in libraries and frameworks, including Express.js.
2. **Express.js**: Express.js is a minimal and flexible Node.js web application framework that simplifies building robust, RESTful APIs. Its middleware architecture streamlines the development process and supports various extensions and plugins.
3. **Golang (Go)**: Golang was chosen for its efficiency, strong performance, and simplicity. Go's statically typed nature and built-in concurrency support make it an excellent choice for building scalable and performant back-end API services. Go has strong support for GraphQL and JSON data, allowing easy interchange of data with the other chosen technologies. It also has a vibrant open-source community and a focus on writing clean and maintainable code (Barney & Gillis, 2023).

## Database and Data Query Language

1. **MongoDB**: MongoDB is a NoSQL document database that provides flexibility and scalability. Its schema-less design is well-suited for projects with evolving data structures, and it offers horizontal scaling options (MongoDB, n.d.). Given the flexibility of GraphQL, which can accommodate changes in the data schema, MongoDB complements the technology stack nicely.
2. **GraphQL**: GraphQL was chosen as the query language for its flexibility and efficiency in data retrieval. It allows clients to request exactly the data they need, reducing over-fetching and under-fetching of data. This is particularly beneficial in front-end development, where minimizing data transfer enhances application performance (*Introduction to GraphQL | GraphQL*, n.d.).

## Containerization and Development Environment

1. **Docker**: Docker containers provide a consistent and reproducible environment across different stages of development, from local development to production deployment. Containerization simplifies the setup of the development and production environments, ensuring that the application behaves consistently in various settings.
2. **JetBrains GoLand**: GoLand is an IDE specifically designed for Go development. It offers a rich set of features, including code navigation, intelligent code completion, and debugging tools tailored to the Go language (*GoLand by JetBrains: More Than Just a Go IDE*, 2021). Using an IDE like GoLand improves developer productivity and code quality.

## Hosting and Cloud Infrastructure

1. **Amazon Web Services (AWS)**: AWS is a leading cloud service provider known for its reliability, scalability, and extensive suite of services. AWS Elastic Beanstalk, for instance, simplifies deploying and managing containerized applications, making it an ideal choice for hosting our Docker containers. AWS also provides infrastructure as code (IaC) tools like AWS CloudFormation for defining and deploying resources in a repeatable and automated manner.

# Current Solutions

The server monitoring space is a competitive and evolving market, with several existing solutions designed to address the business problem of monitoring servers, performing health checks, and providing real-time status information. Current market offerings include:

1. **Open-Source Monitoring Tools (e.g., Nagios, Zabbix)**: These are popular open-source tools widely used for server monitoring. They provide a range of features for monitoring services, performing checks, and alerting on incidents. While they offer flexibility and are cost-effective, they can be complex to set up and maintain (Gupta, 2023).
2. **Commercial Monitoring Services (e.g., New Relic, Datadog)**: These are cloud-based services that offer comprehensive server monitoring, application performance monitoring (APM), and infrastructure monitoring. They provide a wide range of features, including real-time dashboards, incident management, and scalability (Datadog, 2016). However, they often come with subscription fees, which can be costly for larger deployments, and lengthy implementation processes are required to fully leverage the tools.

This project differs from current offerings in several ways:

**1. Focus on Simplicity and Specificity**

* This tool focuses on simplicity by offering core functionality (http and ping checks) without overwhelming users with excessive features.

**2. Customization and Scalability**

* This tool provides ease of customization for organizations that want to tailor monitoring to their specific requirements.

**3. Cost Efficiency**

* Existing commercial solutions require significant investment in service fees, while existing open-source solutions incur significant hosting and administration cost. This tool’s minimal footprint mitigates both hosting and administration cost.

**4. Learning Curve:**

* This solutions emphasis on simplicity and user-friendly interfaces reduces the learning curve for new users, making it more accessible to a wider audience.

# References

Barney, N., & Gillis, A. S. (2023). Go programming language. *IT Operations*. https://www.techtarget.com/searchitoperations/definition/Go-programming-language

Datadog. (2016, November 4). *Datadog + New Relic: Monitor every layer of your stack | Datadog*. Datadog + New Relic: Monitor Every Layer of Your Stack. https://www.datadoghq.com/blog/new-relic-apm-synthetics-monitoring/

Digital, Q. (2023). Webinar: How to Build Customer Trust with Outage Communications. *Questline Digital*. https://www.questline.com/blog/webinar-build-trust-outage-communications/

*GoLand by JetBrains: More than just a Go IDE*. (2021, June 10). JetBrains. https://www.jetbrains.com/go/

Gupta, P. (2023). Zabbix vs Nagios. *EDUCBA*. https://www.educba.com/zabbix-vs-nagios/

*Introduction to GraphQL | GraphQL*. (n.d.). https://graphql.org/learn/

*JavaScript With Syntax For Types.* (n.d.). https://www.typescriptlang.org/

MongoDB. (n.d.). *What is MongoDB?* https://www.mongodb.com/what-is-mongodb

*PatternFly • About PatternFly*. (n.d.). PatternFly Docs. https://www.patternfly.org/get-started/about/

Team, P. F. (2022). Importance of having a good monitoring system. *Pandora FMS Monitoring Blog*. https://pandorafms.com/blog/why-you-need-a-monitoring-system/