Project Plan

Project Name: A cloud-native performance monitoring system

Client: Research Institutes of Sweden (RISE)

List of Contents

1. Introduction

This document will provide all the initial details about the project specification, who is the client, and the requirements needed. The paper is organized in different sections, each section describes individual responsibilities, group members, defined roles, project description, and the time plan for the following steps. It also ensures high-quality work in every single phase of the process.

We will develop an application that monitors resource usage of a virtual machine, running on AWS. The application contains two parts, a CloudWatch API, as well as a graphical dashboard, in the form of a website, responsible for visualizing those metrics. Users will be able to customize the dashboard to their own needs, and administrators will be able to toggle on or off the collection of specific resources.

2. Project Description

2.1 Purpose of the system

The system provides an easy way to monitor system resources of a virtual machine, which does not necessarily require the user to have SSH (Secure Shell) access to the machine. Users will be able to choose what resource metrics they want to view in a graphical interface, using different types of graphs, and for how big a time span they want to monitor some resource. As in; a given graph should be configurable to view a resource over a few minutes, a few hours, or even a few days (or something along those lines).

2.2 Interaction with existing systems

Our system will interact with existing resource log files on Linux, which it will scour for its own log file. It will depend on Apache or Nginx, since we will serve users a graphical dashboard, using a web site. To make it an actual package that can easily be deployed, we might choose to interact with existing systems and packaging systems on Linux, such as system and deb/rpm, respectively. There may be additional interactions if the group chooses to continue with AWS or OpenStack.

2.3 User types in the system

There should be two types of users, an admin, and regular users.

The admins only job is to allow people to become users, and they should be able to turn off the collection of specific metrics from the dashboard.

The regular users and the admin should be able to access the dashboard and change their view, meaning that they can choose what metrics to view, for how long a period and so on.

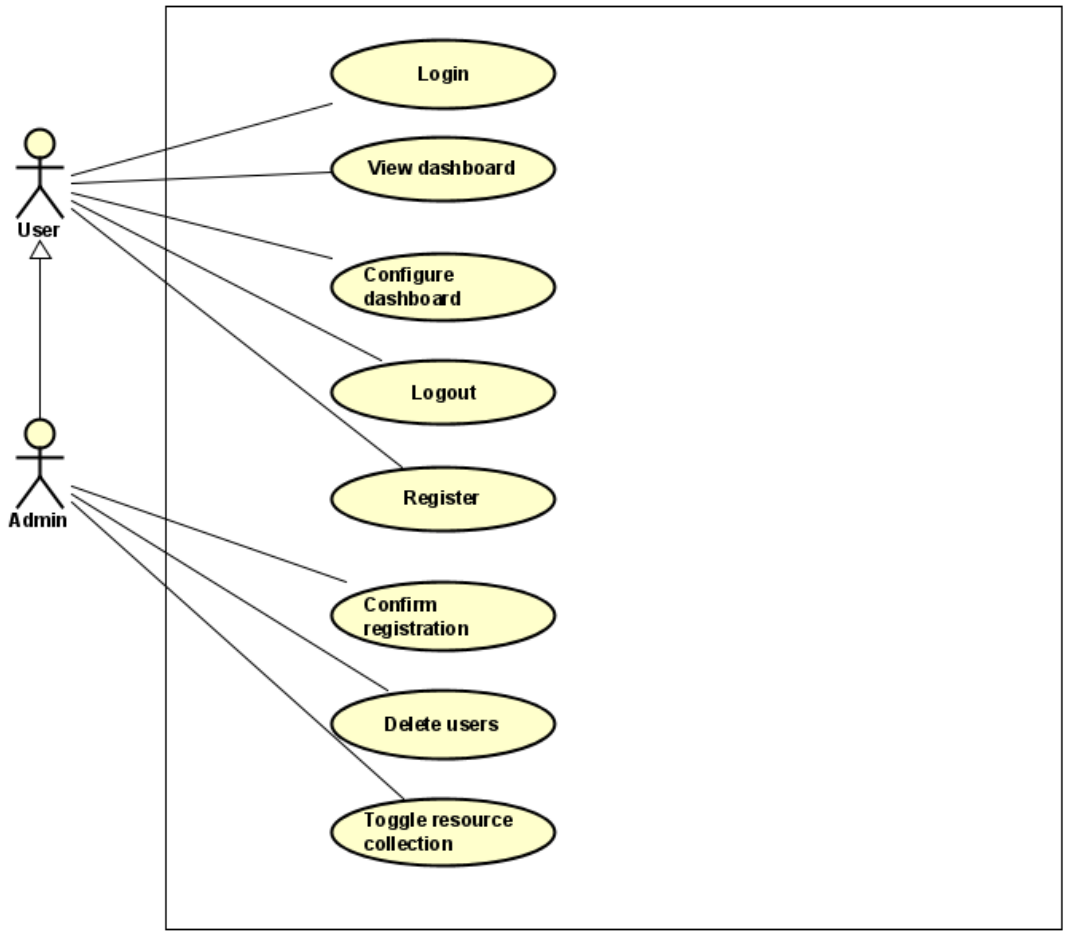

Figure 1: Use Case diagram of the application.

3. Work Organisation

3.1 Project Team

Name	Role	Contact Info
Emil Larsson	Project Manager	eln10007@student.mdh.se
Gezim Cako	Developer	gco20001@student.mdh.se
Gunnar Andersson	Git Manager	pan18010@student.mdh.se
Johan Karlsson	Developer	jkn18010@student.mdh.se
Roland Plaka	Developer	rpa20001@student.mdh.se
Sandra Eriksen	Developer	sen18014@student.mdh.se
Ville Svensson	Developer	vsn17004@student.mdh.se

3.2 Steering Group

Name	Contact Info
Jan Carlson	jan.carlson@mdh.se
Robbert Jongeling	robbert.jongeling@mdh.se

3.3 The client

The client is RISE (Research Institutes of Sweden). They are a research institute that collaborate with universities and companies to innovate and promote sustainable growth.

Name	Contact Info
Mahshid Helali Mogshadam	mahshid.helali.moghadam@ri.se
Mehrdad Saadatmand	mehrdad.saadatmand@ri.se

Role	Responsibilities	Members
Project Manager	Manages project in accordance with the project plan.	Emil Larsson
	Contact point with the client.	
	Monitors and reports progress to the steering group.	
	Heads project documentation.	
	Will also perform as a scrum master to make sure that the team is not distracted by outside factors during weekly sprints.	
Git Manager	Helps the team with anything related to Git. Responsible for managing the repository where the code is stored.	Gunnar Andersson
Developers	Design, implementation, and testing.	All team members
	Writes documentation.	
	Reviews and approves project deliverables.	

4. Development Process

The Agile development method chosen is SCRUM. Which includes a weeklong sprint, ending with a meeting every Monday where every member will report the status of each task they have completed under the sprint. A discussion regarding what the next sprint backlog will contain, and if necessary, re-writing earlier tasks that is not of sufficient quality.

However, instead of having a daily short meeting which is usually the norm in SCRUM the group will opt to instead use Discord as communication as it increases flexibility for each member.

While working on the implementation and design, we will test the functionality that we built frequently. As soon as we have got some parts to work together, we will test them to see that they provide the functionality we want.

4.1 Communication Overview

Type of Communication	**Method / Tool**	**Frequency**	**Information**	**Participants**
Project Meetings	Discord and in person.	Weekly (Mondays and on event)	Project status, problems, risks, altered requirements.	Project Team
Documents	Google Drive	During the project phases.	All relating documents.	Project Team Steering Group The client
Version Control	Git GitHub	During development and final phase.	Git will be used to keep track of changes to the source code. GitHub will be used to store and share code.	Project Team
Task Organization	Trello	During the project phases.	Organize our tasks and keep track of them.	Project Team

5. Delivery Overview

5.1 Deliverables and Receivers

ID	Deliverable	Planned Date	Receiver
D1	Project Plan	19.11.2020	Steering Group
D2	Design description (first version)	03.12.2020	Steering Group
D3	Design description (final version)	14.01.2021	Steering Group
D4	Product	14.01.2021	Steering Group Client
D5	Project Report	14.01.2021	Steering Group
D6	Individual Report	14.01.2021	Steering Group

5.2 Initial Time Plan

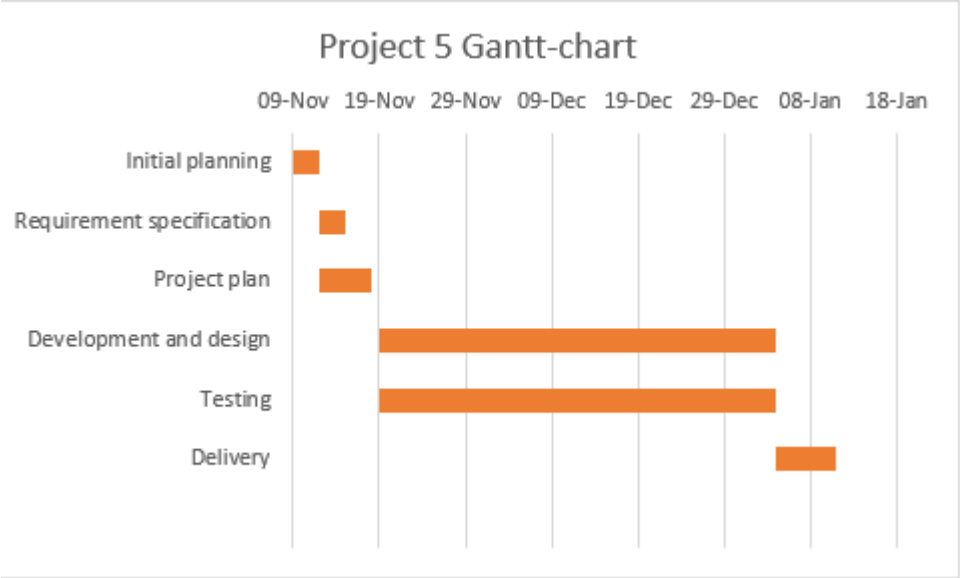

Figure 2: A Gantt chart of our time plan.

6. Quality Assurance

Meetings, discussions, and phases will be documented. By having weekly internal meetings, we can ensure that tasks are discussed and reviewed. The customer will provide feedback and be available via emails or meetings in person to prevent possible misunderstandings. By using agile methodology, we can divide the implementation phase into several sprints, where we can review our tasks before and after each sprint.

In these sprints we will also include the writing of documentation, alongside the implementations, as well as the creation of presentations. The documentations and presentations will be subject to internal review, to ensure their quality. We will ensure that each task within the sprint has a deadline, to have time for review. Reviews will be carried out by each team member, either on Discord or in physical meetings.

6.1 Risk Management

Risk	**Likelihood of Event**	**Mitigation Strategy**
For any personal reason, a team member quit from the group, leaving the rest of the team with a higher quantity of work per person.	Unlikely	The best way to prevent this risk is to discuss, and no individual task should be undertaken. Every phase should well documented.
Members of the team could have trouble managing the technologies due to the lack of experience with such technologies. It also may lead to longer development.	Certainty	The team members should try to consider their strengths and weaknesses when selecting tasks.
The relatively high number of team members could lead to difficulties in coordinating team meetings, actions, or decisions.	Unlikely	Communication and work group spirit could mitigate the risk.
The requirements can change after the designing phase which turns us back into the previous phases.	Somewhat Likely	Asking for any unclear requirements and constantly informing the client for every further development.

6.2 Requirements and Constraints

1. The application will log multiple metrics not yet specified fully by the client that measures performance such as throughput, response times et cetera. These metrics are to be displayed visually on a dashboard.

2. The client initially proposed to build this application by using either AWS services or OpenStack. However, we were later given a freer hand in choosing whether to do so or writing it from scratch entirely.

7. Initial Backlog

Database
Implement stored procedures for
- Upload and store
- Download/Retrieve information from DB to website
- Tables for User Accounts
- Stored procedures for account validation

CloudWatch API
- Write collector for some metrics (CPU, RAM initially?)
- Upload metrics to website backend
- Receive commands from website
- Turn off specific metrics collection (delete alarms)

Website Design
- Initial decision on theme
- Add charts
- Create Account
- Admin specific page
- Menus

Website implementation Frontend
- Initial React framework setup
- Implement Website Design
- Implement turning off metrics collection frontend

Website implementation Backend
- Implement Database/PHP two-way communication
- Implement API/PHP two-way communication
- Implement User validation
- Link charts with data
- Implement turning on/off metrics collection backend