A Web-based User Interface Design for a Cloud Monitoring System

University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Final Year Project - BSc in Computer Science

University College Cork, Department of Computer Science

Student: Michael Mullarkey

Student Number: 112457292

Supervisor: Prof. John Morrison

Second Reader: Dr. Sabin Tabirca

Mentor: Dr. Vincent C. Emeakaroha

11th April 2016

Abstract

Cloud monitoring systems are essential for inspecting and monitoring cloud activity effectively.

This project implements a web-based user interface for an existing Cloud monitoring system. The interface provides the ability to configure and start backend monitoring tools which are implemented in java. The interface also allows for user management in the form of creating users and defining roles.

It also provides the ability to visualise graphs with real time and static data of metrics from virtual machines it is connected to.

While there are other applications that have similar functionality, this application is offered as an open source and extensible solution to cloud monitoring systems.

Declaration of Originality

In signing this declaration, you are confirming, in writing, that the submitted work is entirely your own original work, except where clearly attributed otherwise, and that it has not been submitted partly or wholly for any other educational award.

I hereby declare that:

- This is all my own work, unless clearly indicated otherwise, with full and proper accreditation;

- With respect to my own work: none of it has been submitted at any educational institution contributing in any way towards an educational award;

- with respect to another's work: all text, diagrams, code, or ideas, whether verbatim, paraphrased or otherwise modified or adapted, have been duly attributed to the source in a scholarly manner, whether from books, papers, lecture notes or any other student's work, whether published or unpublished, electronically or in print.

Name: Michael Mullarkey

Signature: ____________________ Date: ____________________

Acknowledgments

I have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

A special thank you must go to my mentor, Dr Vincent C. Emeakaroha, for providing me with the opportunity to work on a project of which I have interest in and for the continued support and mentorship over the last nine months. I would also like to thank my supervisor Prof. John Morrison for his invaluable advice and help during the project. Another thanks must go to my family and friends for their forever continued support.

Lastly, thanks most go to the system administrators for their continued help with maintaining the labs, providing technical support and providing equipment whenever needed.

Table of Contents

1 INTRODUCTION 1
1.1 INTRODUCTION 1
1.2 MOTIVATION 1
1.3 AIMS & OBJECTIVES 1
1.4 MAIN BENEFICIARIES 2
2 ANALYSIS 3
2.1 INTRODUCTION 3
2.2 CLOUD MONITORING 3
2.2.1 Cloud Computing 3
2.2.2 Cloud Monitoring Systems 3
2.2.3 Cloud Monitoring Systems Challenges 4
2.3 EXISTING MONITORING PLATFORMS/TOOLS REVIEW 4
2.3.1 Uptrends Infra 4
2.3.2 Monitis 5
2.3.3 StackDriver 6
2.3.4 Graphite 7
2.4 GRAPHING TECHNOLOGY REVIEW 9
2.4.1 Cacti 9
2.4.2 D3.js 9
2.4.3 HighCharts 10
2.4.4 JFreeChart 11
2.5 REQUIREMENT ANALYSIS 12
2.5.1 Functional Requirements 12
2.5.2 Non-Functional Requirements 13
2.6 CONCLUSION 14
3 DESIGN 15
3.1 INTRODUCTION 15
3.2 APPLICATION DESIGN 15
3.2.1 Deciding on Application Template Design 15
3.2.2 Advancing on Application Design 17
3.2.3 Navigation Tab 17
3.2.4 Monitor Management 18
3.2.5 Monitor Configuration 19
3.2.6 Activity Log 19
3.2.7 Dashboard 20
3.2.8 User Management 21
3.2.9 Alert configuration 21
3.2.10 Alert Management 22
3.3 USER ACCESS 22
3.4 SOFTWARE DEVELOPMENT METHODOLOGIES 23
3.5 CONCLUSION 24
4 IMPLEMENTATION 25
4.1 INTRODUCTION 25

4.2 CLOUD MONITORING PROJECT 25
4.2.1 Monitor Management 25
4.2.2 Monitor Configuration 26
4.2.3 Activity Log 27
4.2.4 User Management 27
4.2.5 User Configuration 27
4.2.6 Alert Configuration 28
4.2.7 Alert Management 28
4.2.8 Sending Alerts 29
4.2.9 Dashboard 29
4.2.10 Static Graphing 30
4.2.11 Real-Time Graphing 30
4.3 PROGRAMMING LANGUAGE & TECHNOLOGY 31
4.3.1 Java 31
4.3.2 Servlets 31
4.3.3 JSP 31
4.3.4 JavaScript 32
4.3.5 MySQL &Hibernate 32
4.3.6 Bootstrap 32
4.3.7 MVC 32
4.4 LIBRARIES 33
4.4.1 HighCharts 33
4.4.2 Flot & FlotJF 33
4.4.3 JQuery 34
4.4.4 Validator.js 34
4.4.5 JavaMail 34
4.4.6 Apache Tomcat 34
4.5 TESTING 35
4.5.1 Junit Testing 35
4.5.2 Regression Testing 35
4.5.3 Client Side Testing 35
4.5.4 Browser Compatibility Testing 35
4.5.5 Usability Testing 36
4.6 ISSUES & WORKAROUNDS 36
4.6.1 Graphing 36
4.6.2 Dashboard Functionality 37
4.7 CONCLUSION 37

5 EVALUATION 38

5.1 INTRODUCTION 38
5.2 WORK COMPLETED 38
5.2.1 Monitoring Functionality 38
5.2.2 Alert System 38
5.2.3 Admin Functionality 38
5.2.4 Dashboard Functionality 39
5.2.5 Report Management 39
5.3 WORK UNCOMPLETED 39
5.3.1 Integration 39
5.3.2 Search Functionality 39
5.3.3 Alert Functionality Testing 39

5.3.4Exporting Graphs *39*
5.4PUBLIC EVALUATION 40
5.5USE CASES 41
5.6CONCLUSION 43
6 CONCLUSION 44
6.1INTRODUCTION 44
6.2PROJECT SUMMARY 44
6.3FUTURE WORK 44
6.4LESSONS LEARNED 44
7 BIBLIOGRAPHY & REFERENCES 46
8 APPENDIX 47
8.1OTHER USE CASES 47

1 Introduction

1.1 Introduction

Cloud is a booming, ever expanding industry. This growth has led to advancements in technology associated with cloud computing and cloud monitoring. Due to this growth there has been needs for cloud monitoring systems. Cloud monitoring systems allow users to configure and start monitor tools, visualise metrics and drill down into data, along with a host of other functionality.

Cloud monitoring systems have grown in popularity from year to year due to the need of companies and organisations to monitor and gain access to their data. Often these organisations do not have the resource to do this themselves, therefore they offload that responsibility to a third party company, who will manage and support their cloud monitoring needs.

1.2 Motivation

Cloud monitoring systems can often be laborious and unintuitive. Many systems available today are considered overly complicated and daunting for both experienced users and even people who are new to cloud monitoring systems.

When companies or even users outsource their monitoring needs to third party organisations, they can never be completely confident they're seeing all of your data, as more often than not you do not have full control over the cloud environment the third party is running.

The task of developing a system which counteracts these issues by delivering an easily accessible, intuitive, portable and extensible cloud monitoring system that can be used by both experienced and novice users alike. This system should also give the user full control and access to all of the data and functionality.

1.3 Aims & Objectives

The objectives of this project is to develop an application that will:

- Provide a free, open source and intuitive solution to cloud monitoring systems.
- Provide users the ability to easily configure monitoring tools.
- Provide the ability to start back-end monitoring tools.
- Allow user configuration and management.
- Have a visualisation interface for displaying monitored data using graphs.
- Provide the ability to visually "drill down" into the monitored data.

- Provide a system will do all this and more in fashion that will make it completely approachable for users of all levels and disciplines.

1.4 Main Beneficiaries

The biggest beneficiaries of the project I feel will be users who are new to the cloud computing technology and wish for an easy to use, user friendly cloud monitoring system to help negate the large learning that comes with using other cloud monitoring systems.

Beneficiaries may also include a more advanced user who wish for a more lightweight and user intuitive application. This application provide option for users of all levels.

2 Analysis

2.1 Introduction

In doing the analysis we hope to analyse and critique the difficulties associated with developing a web-based user interface design for a cloud monitoring system. Specifically pointing to the problems associated with usability, extensibility and portability.

This section will also go over essential software practices such as requirement analysis, stating functional and non-functional requirements that are essential to the projects development.

2.2 Cloud Monitoring

2.2.1 Cloud Computing

Cloud computing is an internet based computing that allows for shared resources such as processes and memory to be shared and utilises when needed. Sharing these resources allow for greater coherency, performance and scalability.

Cloud computing has become a highly demanded service due the advantages of scalability, high computing power and cheap cost of its services.

References to cloud computing date back to appearing as early as 1996, with Comaq referencing cloud computing in many aspects in their internal document. Cloud computing found its own niche in the 2000's as being a rational solution too many of the scalability and storage issue facing many organisations then and now.

Organisations such a NASA integrated cloud in its OpenNebula project by becoming the first open-source software for deploying private and hybrid clouds. Other leaps in cloud development include hosting and storage capabilities with Amazon AWS, OpenStack and Microsoft Azure just to name a few popular cloud hosting solutions.

2.2.2 Cloud Monitoring Systems

Cloud monitoring systems combines technologies and software in an effort for managing cloud environments. Cloud monitoring systems host an array of capabilities including monitor and manage resources, configure and start monitoring tools, provide data access to end users, user management and tracking management just to name a few.

There are many professional/high end cloud management systems available today, such ones include StackDriver, Zenoss, Monitis, Rackspace etc. These monitoring systems are often used by larger organisations and are not feasible for the average user running their small cloud environment. Instead these users will user smaller, more lightweight monitoring systems such as Graphite.

2.2.3 Cloud Monitoring Systems Challenges

When dealing in cloud computing there're often a host of challenges that have to be dealt with. Companies will often have to deal with business logistics such as user accessibility. Companies using cloud environments to host their own public cloud do not own the technology which hosts the cloud environment. This means that the owners of these public clouds do not have a full access to their cloud, as the cloud environment where it is hosted is not on their own network.

Public cloud services must also adhere to the infrastructure architecture set by the cloud environment provider, which may lead to reorganisation of the public cloud components.

Often cloud monitoring systems can be overly complicated and unintuitive to users of all levels. This can lead to new users becoming confused or even disinterested.

2.3 Existing Monitoring Platforms/Tools Review

2.3.1 Uptrends Infra

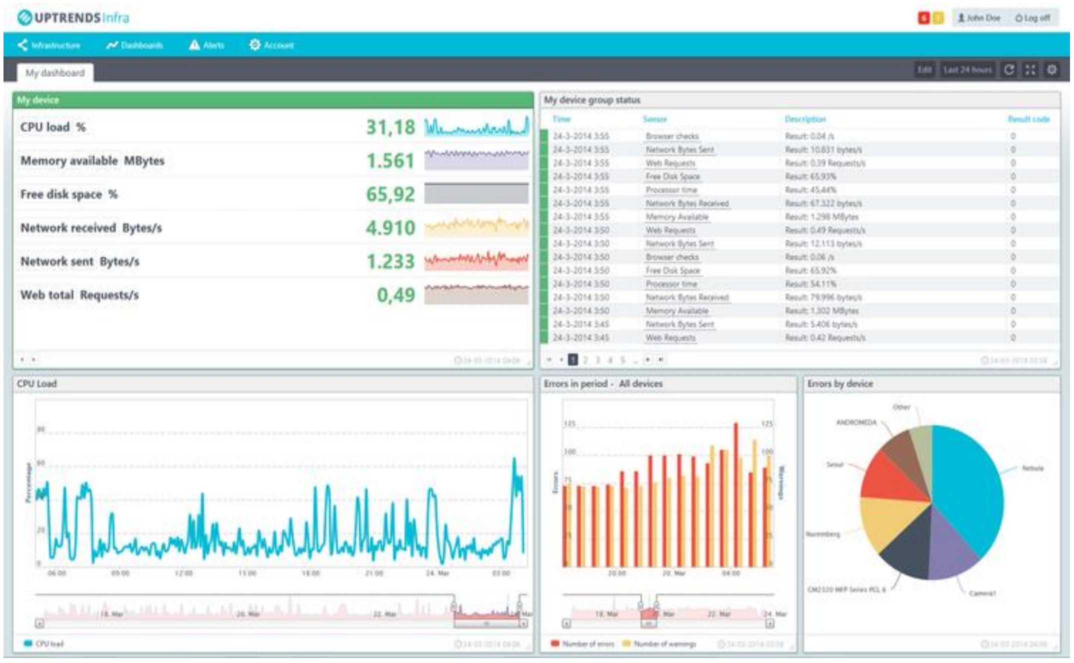
Background:

Uptrends Infra provides an all-in-one hosted network monitoring solution. Receive alerts, analyse errors and prevent future issues.

Many Fortune 500 companies and small to medium sized companies, rely on their network monitoring tools **[1]**.

Key Features:

- Asynchronous/real time data monitoring (CPU load, requests, etc.).
- Clean design.
- Intricate Notification system.
- Mobility compatible.
- Completely customizable dashboard.
- PDF & Excel report exports.



My device

CPU load %	31,18
Memory available MBytes	1.561
Free disk space %	65,92
Network received Bytes/s	4.910
Network sent Bytes/s	1.233
Web total Requests/s	0,49

My device group status

Time	Sensor	Description	Result code
24-3-2014 3:55	Browser checks	Result: 0.04 /s	0
24-3-2014 3:55	Network Bytes Sent	Result: 10.831 bytes/s	0
24-3-2014 3:55	Web Requests	Result: 0.39 Requests/s	0
24-3-2014 3:55	Free Disk Space	Result: 65.93%	0
24-3-2014 3:55	Processor time	Result: 45.44%	0
24-3-2014 3:55	Network Bytes Received	Result: 67.322 bytes/s	0
24-3-2014 3:55	Memory Available	Result: 1.298 MBytes	0
24-3-2014 3:50	Web Requests	Result: 0.49 Requests/s	0
24-3-2014 3:50	Network Bytes Sent	Result: 12.113 bytes/s	0
24-3-2014 3:50	Browser checks	Result: 0.06 /s	0
24-3-2014 3:50	Free Disk Space	Result: 65.92%	0
24-3-2014 3:50	Processor time	Result: 54.11%	0
24-3-2014 3:50	Network Bytes Received	Result: 79.996 bytes/s	0
24-3-2014 3:50	Memory Available	Result: 1.302 MBytes	0
24-3-2014 3:45	Network Bytes Sent	Result: 5.406 bytes/s	0
24-3-2014 3:45	Web Requests	Result: 0.42 Requests/s	0

Figure 1 Uptrends Infra Dashboard - media.bestofmicro

Conclusion:

Uptrends Infras clear and crisp display allows for ease of use. It gives the user exactly what they need without overwhelming them with information. It does all this while providing the user with all necessary functionality such a real time monitoring, performing deep analysis and addresses errors/alerts as they happen.

2.3.2 Monitis

Background:

Monitis allows for easy network and cloud monitoring integration to whatever cloud environments you may need. It is completely customisable with the ability to include additional plug-ins through its open API **[2]**.

Key Features:

- Allows users to set up individual monitors as they please.
- Client email/messaging system.
- Additional online help manuals make the site more usable from the beginning.
- Web map shows regions connected to instances.
- Different tabs for different sections within the one dashboard view (Status, monitors, etc.).
- Allows user to choose different layout of page data (columns & rows).
- Highly customisable dashboard.

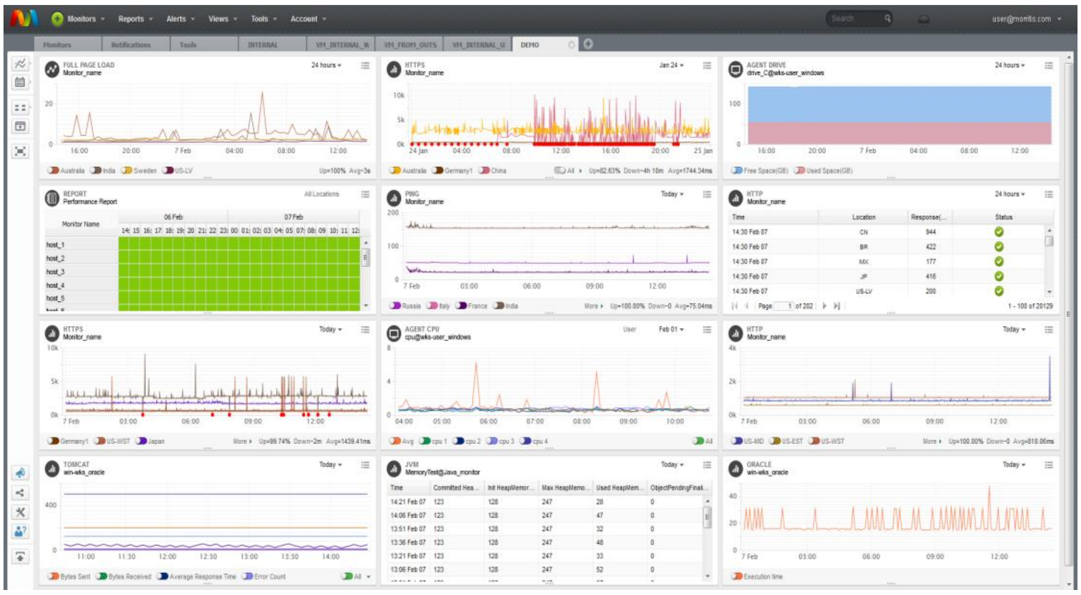


Figure 2 Monitis Dashboard – www.monitis.com

Conclusion:

Monitis provides a highly customisable individual user experience with its cloud monitoring system. Through using the application, I found myself sometimes overwhelmed with all the different icons and options on any single one page, I found a lot of the icons to not be standard for what they were presented to do which led to extra confusion. Technically speaking the application offered everything a good monitoring system should with extra handy functionality such as the ability to mix and match monitors from different sections, allowing the user to create professional reports for things such as performance and transactions transactions.

2.3.3 StackDriver

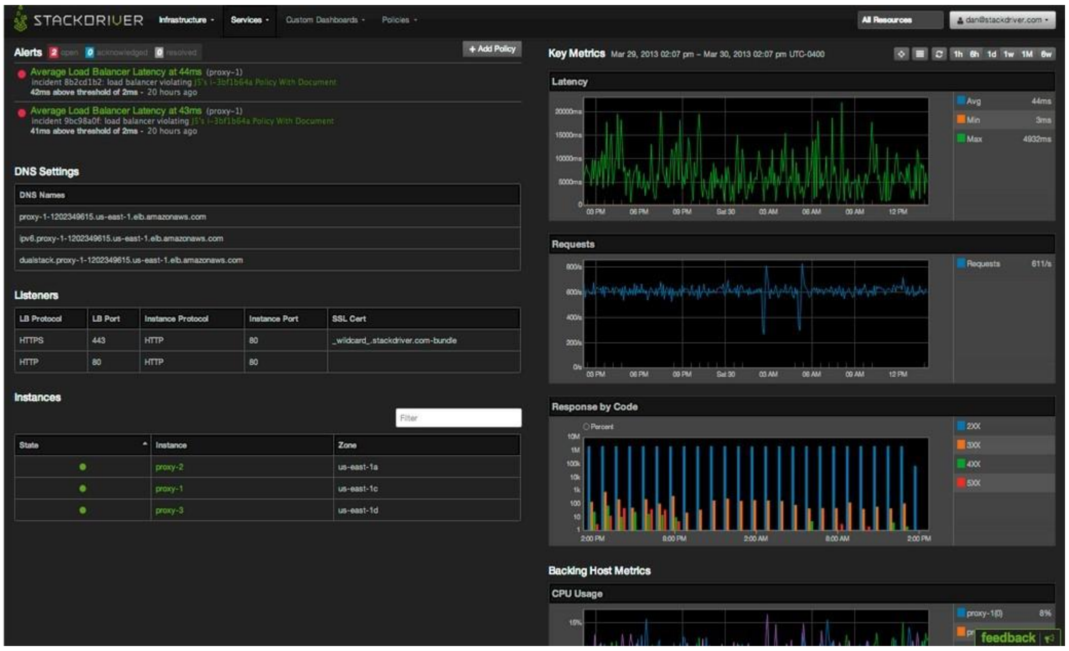
Background:

StackDriver aim to help DevOps manage large, distributed applications running in the public cloud. It visualizes application, system and infrastructure metrics and also provides a policy system to alert users when predefined thresholds are breached **[3].**

Key Features:

- Elastic monitoring – Allows environments to scale automatically.
- Automation – reboots instances when memory exceeds a certain threshold.
- Adds capacity when available capacity goes below the threshold.
- Uptime monitoring – monitors instance health, configure uptime checks.

- Event logging – view most recent changes in cloud and instances. View user changes and all configuration changes.
- Detectability – able to find root cause using analytics, tells you when there is the chance to optimize cloud performance.
- Stylish, sleek design.



Alerts 2 open 0 acknowledged 0 resolved

Average Load Balancer Latency at 44ms (proxy-1)
Incident 8b2cd1b2: load balancer violating JS's i-3bf1b64a Policy With Document
42ms above threshold of 2ms - 20 hours ago

Average Load Balancer Latency at 43ms (proxy-1)
Incident 9bc98a0f: load balancer violating JS's i-3bf1b64a Policy With Document
41ms above threshold of 2ms - 20 hours ago

DNS Settings

DNS Names
proxy-1-1202349615.us-east-1.elb.amazonaws.com
ipv6.proxy-1-1202349615.us-east-1.elb.amazonaws.com
dualstack.proxy-1-1202349615.us-east-1.elb.amazonaws.com

Listeners

LB Protocol	LB Port	Instance Protocol	Instance Port	SSL Cert
HTTPS	443	HTTP	80	_wildcard_.stackdriver.com-bundle
HTTP	80	HTTP	80	

Instances

State	Instance	Zone
●	proxy-2	us-east-1a
●	proxy-1	us-east-1c
●	proxy-3	us-east-1d

Figure 3 StackDriver Dashboard - www.stackdriver.com

Conclusion:

I found StackDriver to be by far the most sophisticated and fully fleshed out application while conducting my research. Although StackDriver was easy to navigate, it was very clear that there was a learning curve when it came to operating and understanding the functionality of the site. While it is clear that StackDirver is intended for larger scale organisation and higher levelled user with a better understanding of cloud monitoring system it was still quite daunting to try and learn the ends and outs of the sites functionality.

2.3.4 Graphite

Background:

Graphite is a free open source and lightweight software (FOSS) tool for monitoring and graphing the performance of computer systems. Graphite is a highly scalable real-time graphing system. **[4]**

Key Features:

- Completely open source and extensible.
- Extremely lightweight,
- Allows for real time monitoring,
- Data can be visualised through graphite's web interfaces.
- Full admin and user management.
- Full access to data.

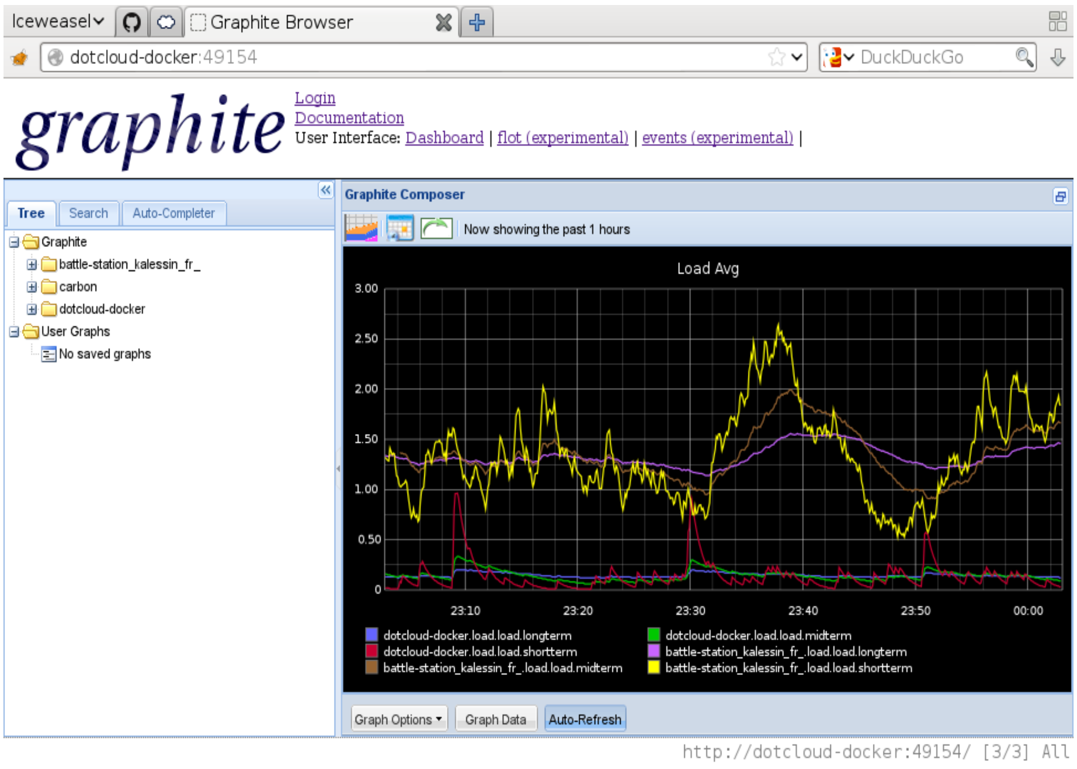

Iceweasel
Graphite Browser
dotcloud-docker:49154
DuckDuckGo
graphite
Login
Documentation
User Interface: Dashboard | flot (experimental) | events (experimental) |
Tree
Search
Auto-Completer
Graphite
battle-station_kalessin_fr_
carbon
dotcloud-docker
User Graphs
No saved graphs
Graphite Composer
Now showing the past 1 hours
Load Avg
3.00
2.50
2.00
1.50
1.00
0.50
0
23:10
23:20
23:30
23:40
23:50
00:00
dotcloud-docker.load.load.longterm
dotcloud-docker.load.load.shortterm
battle-station_kalessin_fr_.load.load.midterm
dotcloud-docker.load.load.midterm
battle-station_kalessin_fr_.load.load.longterm
battle-station_kalessin_fr_.load.load.shortterm
Graph Options
Graph Data
Auto-Refresh
http://dotcloud-docker:49154/ [3/3] All

Figure 4 Graphite Dashboard - graphite.wikidot.com

Conclusion:

Graphite is a good lightweight solution to cloud monitoring tools. It allows for full control of data and allows for admin and user management, all things I hope to integrate into my own application.

2.4 Graphing Technology Review

2.4.1 Cacti

Background:

Cacti is a complete network graphing solution designed to harness the power of RRDTool's data storage and graphing functionality. Cacti provides a fast poller, advanced graph templating, multiple data acquisition methods, and user management features out of the box. All of this is wrapped in an intuitive, easy to use interface that makes sense for LAN-sized installations up to complex networks with hundreds of devices **[5]**.

Key Features:

Pros:

- Stores all necessary information to create graph and populate them with data in a MySQL database.
- Lots of documentation.
- Built in user management.
- Open source & free.

Cons:

- Not particularly aesthetically pleasing.
- Written in php.
- Not asynchronous.
- Old technology.

My Thoughts:

Cacti is a fairly robust lightweight graphing tool that on the outer shell seems to do all the necessary things that I need. However, it is an old api and although it is written in php, a language I know, I would still prefer a JavaScript or Java graphing library as it would integrate better within the project.

2.4.2 D3.js

Background:

D3.js is a JavaScript library for manipulating documents based on data. D3 helps you bring data to life using HTML, SVG, and CSS. D3's emphasis on web standards gives you the full capabilities of modern browsers without tying yourself to a proprietary framework, combining powerful visualization components and a data-driven approach to DOM manipulation **[6].**

Key Features:

Pros:

- Large array of libraries.
- Huge community.
- Plenty of resources.
- Extremely flexible API.
- Asynchronous.
- Modern design.

Cons:

- Steep learning curve.

My Thoughts:

D3.js is one of the biggest charting library's out there at the moment and has gained great popularity by its ability to do pretty much anything. But with its great flexibility it seemed to have created a steep learning curve, this paired with a confusing syntax may prove challenging to fully optimise within the project.

2.4.3 HighCharts

Background:

Highcharts is a charting library written in pure JavaScript offering an easy way of adding interactive charts to your web site or web application. Highcharts currently supports many chart types, including line, spline, area, areaspline, column, bar, pie, scatter, bubble, gauge and polar chart types. Many of the chart types can be combined in one chart. Users can export the chart to PNG, JPG, PDF or SVG format at the click of a button, or print the chart directly from the web page **[7]**.

Key Features:

Pros:

- Sleek modern design.
- Easy to follow syntax.
- Quick and responsive.
- Expansive range of graph types.
- Large community.

Cons:

- Not much support provided outside of the php language.

My Thoughts:

Highcharts is a very aesthetically pleasing charting library, it also has great flexibility in viewing graphs. All this paired along with that is a pure JavaScript library means everything can be rendered client side. However, there is some concerns when it comes to support outside of php, from looking over forums there seems to be no support for server side request languages other than php.

2.4.4 JFreeChart

Background:

JFreeChart is a free 100% Java chart library that makes it easy for developers to display professional quality charts in their applications **[8]**.

Key Features:

Pros:

- Completely open source.
- Well-documented api.
- Completely written and implemented in Java.

Cons:

- Old design.
- Not fully asynchronous.
- Does most of the rendering server-side.

My Thoughts:

JFreeChart is a graphing language which is completely written in Java, which means I could begin implementing it relatively quickly as I am quite familiar with the Java Language. However due to its old and clunky design, along with the fact it doesn't support true real time rendering mean it more than likely will not be using this library.

2.5 Requirement Analysis

For any software project setting the requirement analysis is an important and integral part of its process and design. Before development of the project began I set the Functional requirements to define the specific behaviour or functions of the application and the non-functional requirements to define what should be implemented but not critical to the application functioning.

2.5.1 Functional Requirements

Monitoring

- Ability to execute uptime and status monitoring.
- Ability to monitor individual metrics such as CPU, GPU, memory etc.
- Allow the user to Start and stop monitors and metrics with a button click.
- Allow the user to group virtual machines and metrics.
- Allow the user to configure monitoring tools.
- Allow the user to completely edit tools and configurations.

Alert System

- Allow the user to configure and setup alerts.
- Allow the user completely edit alerts.
- Alert should send email to the creators email address once alert is activated.

Admin Functionality

- Allow admin users to create either normal or admin users.
- Allow admin users to edit users.
- Allow admin users edit all configured monitor tools.
- Allow admin users access to all user logs.

Dashboard

- Allow user to create both real time and static graphs.
- Allow user to drill down into graph data.
- Allow user add and remove graph on the fly.

Report Management

- Allow user to export image of individual graphs.
- Allow admin access to all user’s activity

Search

- Allow user to access to a full site search function.

2.5.2 Non-Functional Requirements

- Interoperability

  Interoperability is a key non-functional requirement as the cloud monitoring system needs to work with any cloud environments without any restricted access or limitations.

- Extensibility

  Extensibility is needed as the application will need to take future growth into consideration. As your existing cloud environment grows and changes so those the application if it is to remain relevant to those applications it depends on.

- Response Time

  Response time is the total amount of time it takes to respond to a request for service. As my application requires graphing and monitoring many metrics at the same time, response time is crucial to have seamless operability.

- Robustness

  Robustness is needed in terms of the ability for the application to be able to deal with points of failure. This is an important requirement as this application has many input forms. Also when connecting to environments and starting monitoring tools it is crucial that the application gives constructive errors without the application failing.

- Maintainability

  Maintainability is needed meet new requirements of the system or supporting system requirements and make future maintenance easier.

- Documentation

  Documentation is an integral part of this application. Documentation is needed so that there is a clear and concise reference for the functionality of the application. As this application is required to be extremely usable for even new users, it is important it supports this non-functional requirement.

- Testability

  When developing an application with so many possible points of failure and so many inputs it is important that the system itself supports testing, for testing an array of different testing environments. An application with high testability will mean the system will find faults from testing much more efficient.

2.6 Conclusion

By completing this planning and research of the project it has given me a much idea in how to implement and design my project.

By conducting research into existing cloud and network monitoring applications it has given me great insight into what should actually be implemented into my own application, such as different web application design, look and feel, and also the different types of functionality.

Likewise, by researching into existing graphing technologies, it has given me the ability to accurately pick out which graphing library to use.

Other researching was conducted in the choice of programming languages and frameworks to use.

Due to this research it gives the project a good foundation to begin design and development.

3 Design

3.1 Introduction

As this applications most highlighted feature is its usability, portability and extensibility, it made it crucial to have a successful design process. Much of the design time went into developing various different screen layouts with different functionalities, this is shown through mock up designs. This section also goes through the process of designing and defining the software mythology of choice and different user roles/

Note: Features referenced in this section are explained in the implementation section of this report.

3.2 Application Design

Beginning the application design, the overall consensus was to create a clean, modern and highly intuitive user interface. In order to fully capture the best design, we used the research of existing monitoring systems and came up with 3 pre-prototype designs. In this section we will go over these proposed designs.

By building all of these templates in Microsoft paint it gave the opportunity to quickly represent ideas in understandable format to express concepts and show for extra feedback.

3.2.1 Deciding on Application Template Design

Design 1

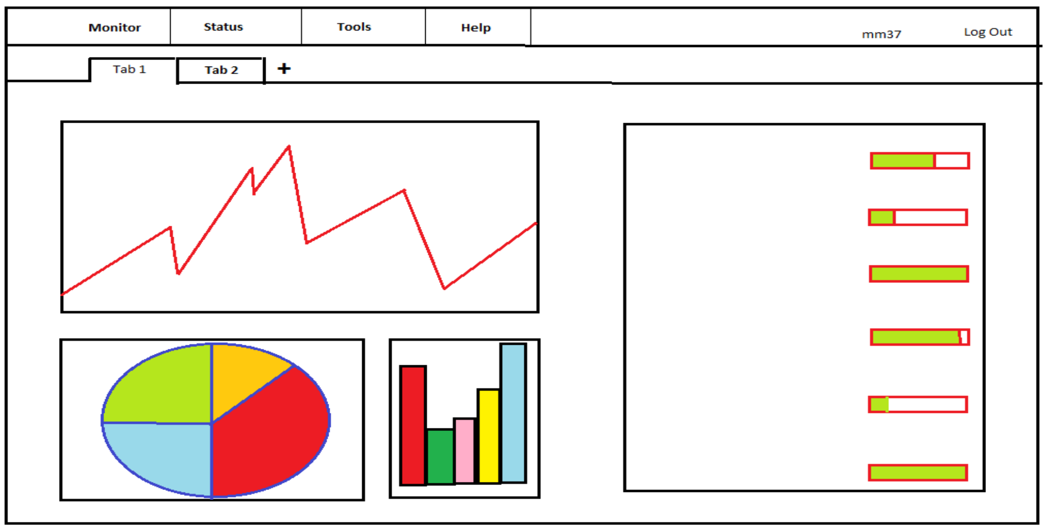


Figure 5 Design 1

Design 1 integrates a lot of functionality that was seen in previous monitoring systems. The most common functionality found in other monitor systems is the flexibility and freedom in its dashboard. This dashboard allows the user to have multiple tabs for showing off different personalised views. These dashboards will be populated by a multitude of different metrics being monitored. The rest of the design is fairly standard with a horizontal navigation with user options found in the top right of the navigation.

Design 2

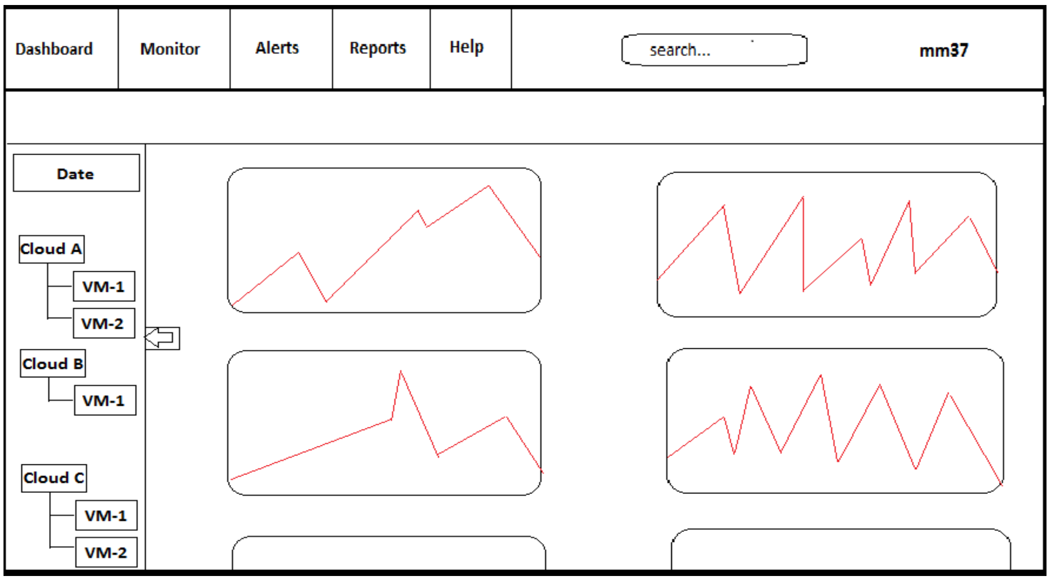


Figure 6   Design 2

Design 2 allows the user a small bit less flexibility than design 1's dashboard, however, this is traded off for more intuitive design of the application for the user. This design allows sectioned off areas for the graphs and unlike design 1, it allows for a foldable side section that shows monitors hierarchy, it also allows the user to start showing graphs for these specific machines. Much of the inspiration came from the graphite monitoring system for this design.

Design 3

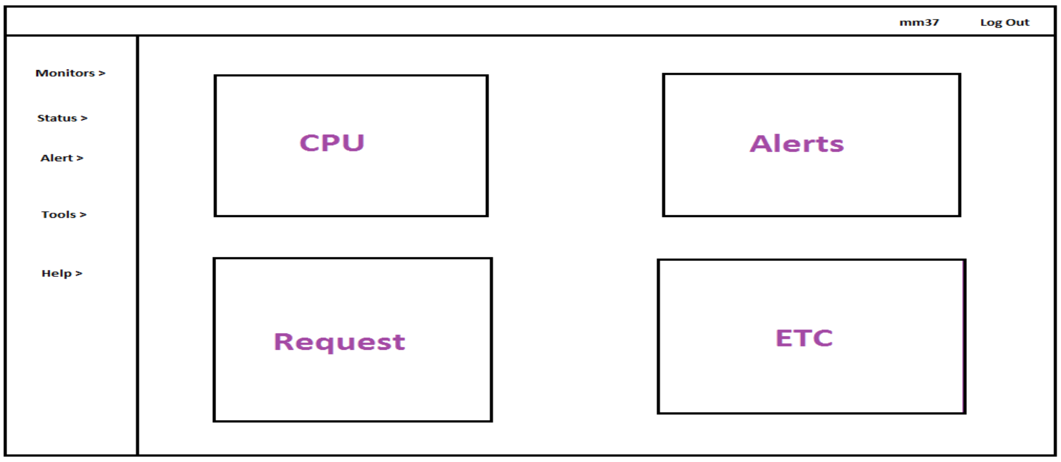


Figure 7 Design 3

Design 3's inspiration comes windows phone tile design. Each section will be broke up using tiles for each monitoring function. And inside each section they will have their respective tiles for functionality. This design has both vertical and horizontal navigation, with the vertical navigation allowing the user to change between sections and the horizontal used for user functionality.

Chosen Design

After a lot of discussion and productive feedback it was design 2 that was picked to be the template design for the application. This design is very intuitive and with the technology that is present in the user interface and application design world, it will be possible to give a very modern and clean look. This design also makes good use of space by using foldable section that when closed open up more space for the screen to use. This design will also be compatible with mobile and tablet screens.

3.2.2 Advancing on Application Design

After deciding on the application design that was to be used. It was then time to decide on what the functionality of buttons and also sketch out some of the more important pages to give an overall feel and look to the application and flesh out the desired functionality.

A more in depth review of these functionalities will be given in the implementation.

3.2.3 Navigation Tab

After much consideration and deliberation, the first draft of functionality on the main navigation tab was decided. For the dashboard the user has the ability to go straight into their own dashboard. For the monitor and alerts the user has the option to either configure or manage (view and edit) their previous created monitor or alert.

For reports, an admin can view all user activity. There is also a help section for a help manual and FAQ of the overall site functionality, and lastly is the user option which allow the user to sign out and in, and if they're an admin they can go into their admin environment.

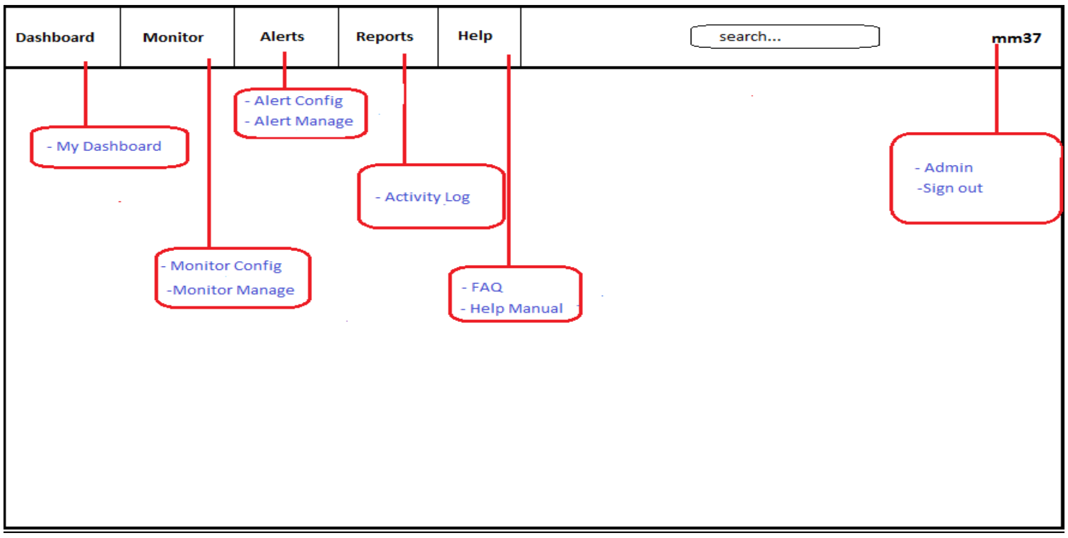


Figure 8 Navigation Design

3.2.4 Monitor Management

This page allows the user to manage and edit all of their monitors and proxies that they have configured. It also allows the user to start and stop the monitors individually.

This page design breaks up each section to present a clean and intuitive display for the user to interact with. It provides an edit tab for every entry which allows the user to edit or remove each monitor. From here the user can also start and stop monitors using the symbolled buttons.

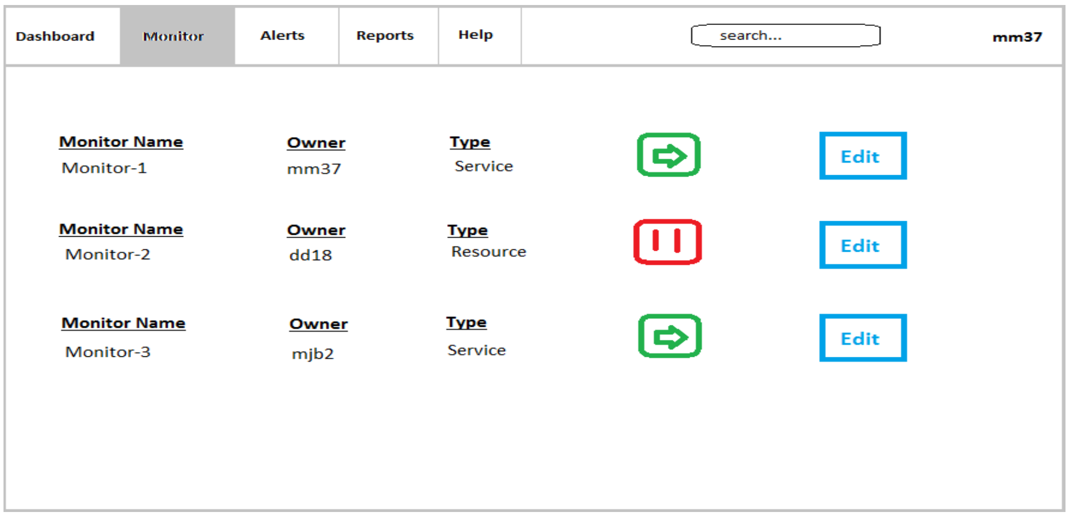


Figure 9 Monitor management Design

3.2.5 Monitor Configuration

This pages allows the user to configure service and resource monitors, along with the ability to configure proxies.

As this page has many inputs, it was crucial to break it up efficiently so as not to confuse to user between the forms. To do this a JavaScript technology known as accordions were used to effectively distinguish between configuring service, resource and proxies.

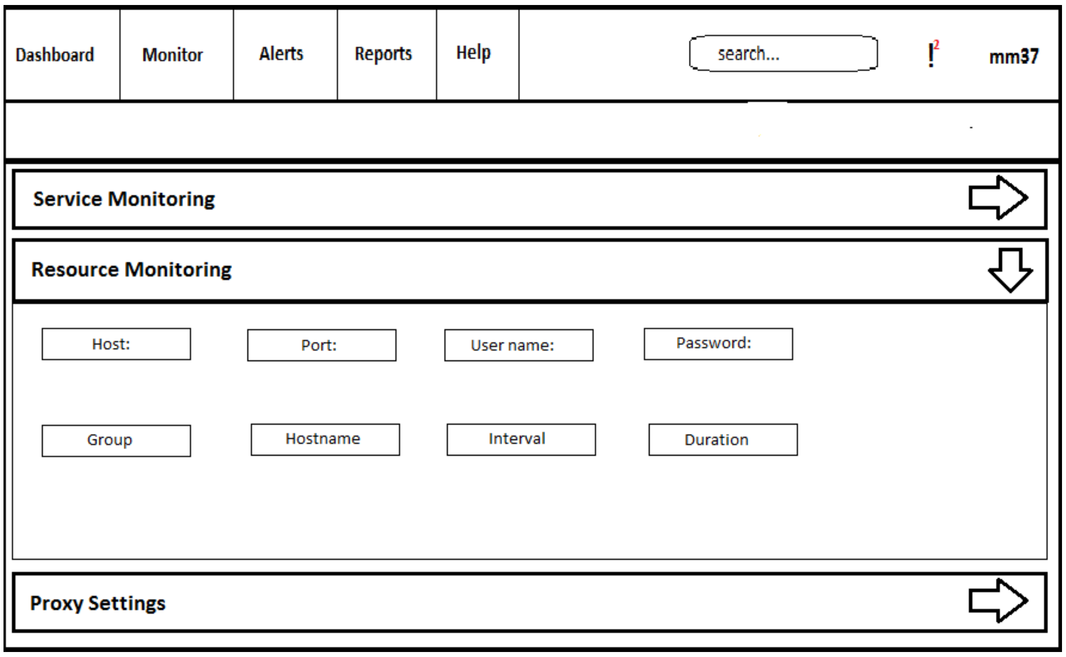


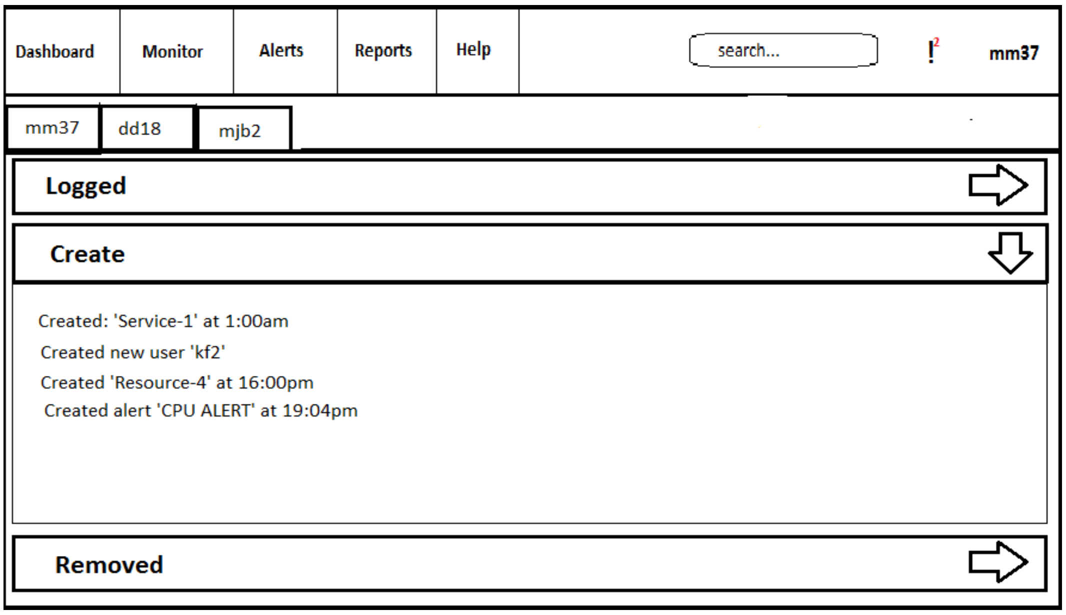
Figure 10 Monitor Configuration Design

3.2.6 Activity Log

The activity log allows the user to view all actions of each user.

Activity is logged using file saved and read locally. Every time a user is created a file is created for that user, then every time they perform an action it is saved to their specific file.

In designing the activity log, the main concern was the way in which to present the large array of data. To deal with this large data tabs are used to separate user, then accordions are used to separate certain actions to user may perform.



Created: 'Service-1' at 1:00am
Created new user 'kf2'
Created 'Resource-4' at 16:00pm
Created alert 'CPU ALERT' at 19:04pm

Figure 11 User Activity Log Design

3.2.7 Dashboard

The dashboard allows the user to graph real-time and static line graphs. It gives the user options such as positions and dates. It also shows the user configured monitor in a treatable structure.

This allows the user to both see monitor configuration and view live graph data and edit as need be.

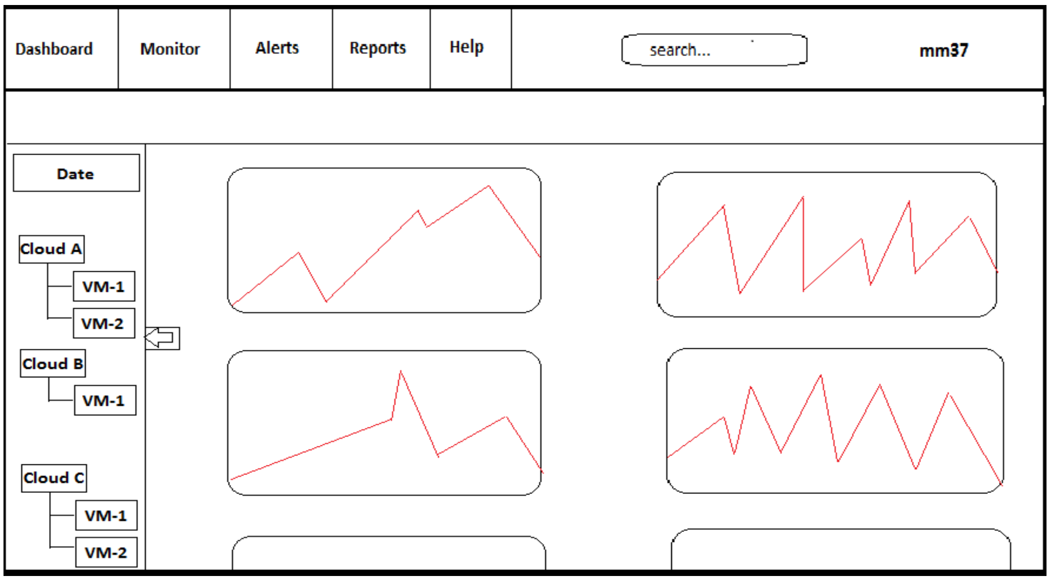


Figure 12 Dashboard Design

3.2.8 User Management

These pages are to let admin users configure and manage users. They also allow the user to configure other users, edit user parameters and remove users.

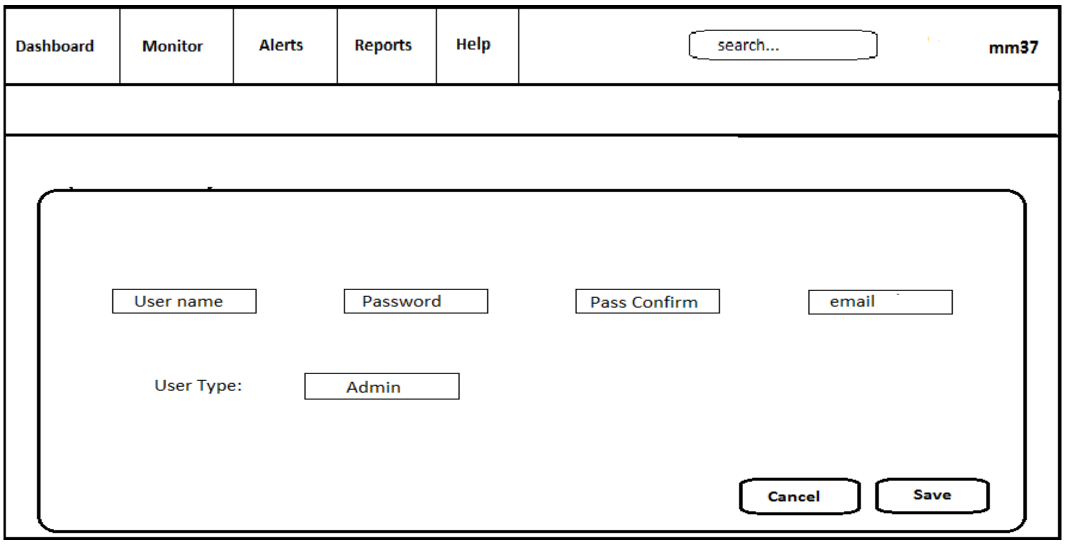


Figure 13 User Configuration Design

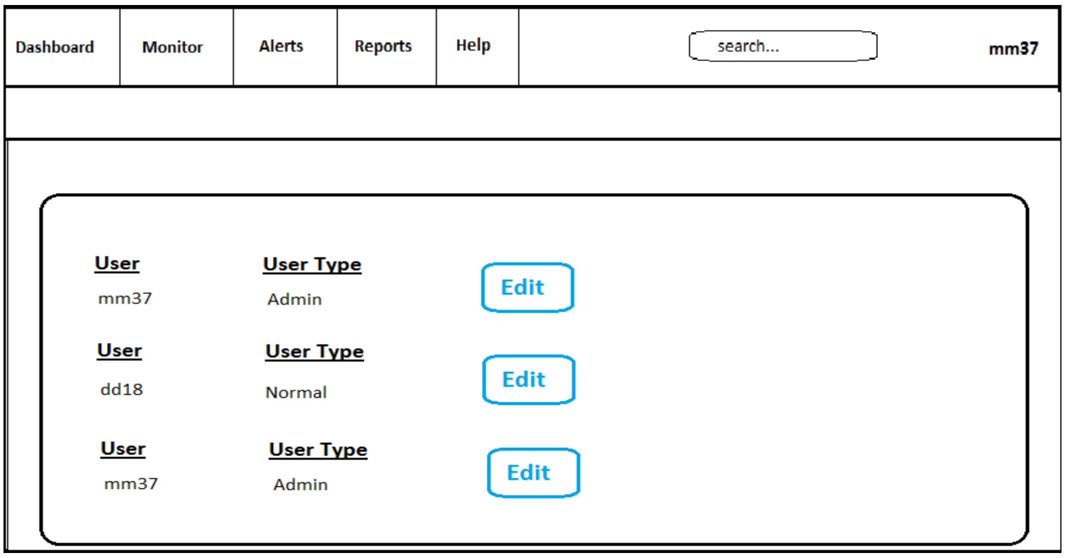


Figure 14 User Management Design

3.2.9 Alert configuration

This page allows the user to set up and configure alerts. This is a simple form that lets the user set the metric they wish to monitor and set the threshold that the metric must not past.

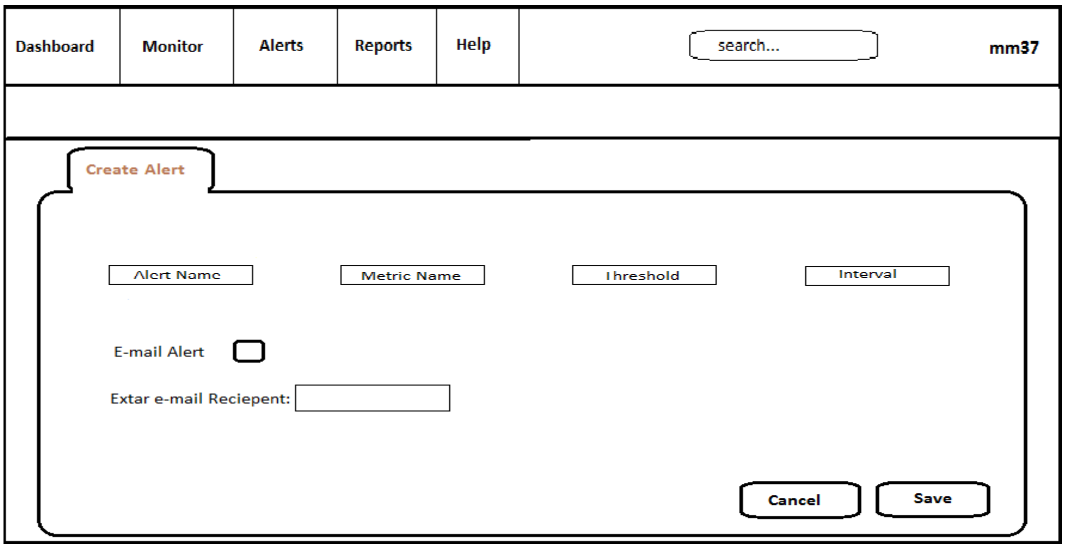


Figure 15 Alert Configuration Design

3.2.10 Alert Management

This page allows the user to manage their already configured alerts. This management page allows them to view the alerts details, edit details and remove the alert completely.

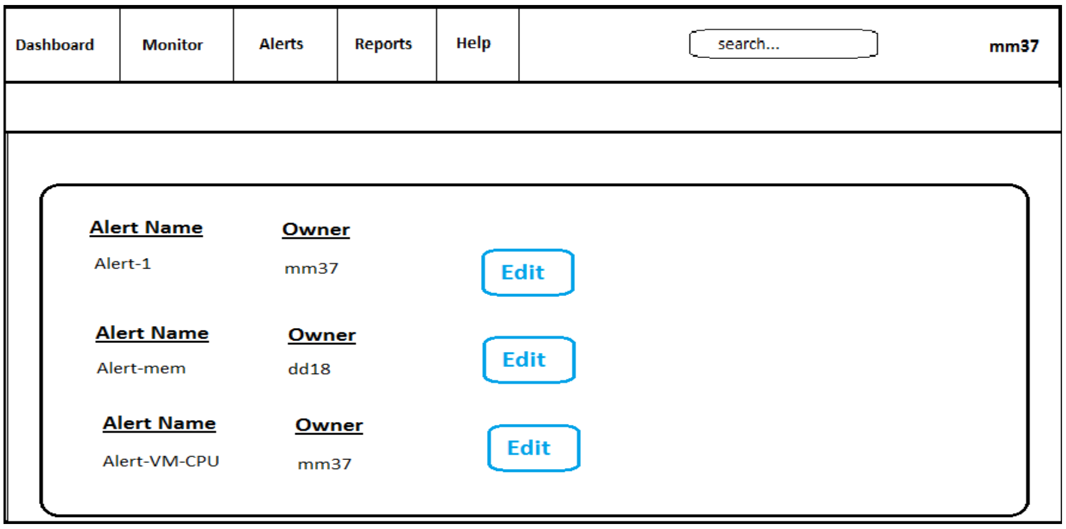


Figure 16 Alert Management Design

3.3 User Access

User access is split into three different categories non-user, standard user and admins. Access is limited for non-user and standard users with admins with full access and functionality.

Non-users just have the ability to view the website in a static environment. Normal users have basic functionality. And admins have full access to all site functionality.

	Non-User	Standard User	Admin
Browse site	YES	YES	YES
See monitoring data	NO	YES	YES
Setup dashboard	NO	YES	YES
Start/Stop monitoring	NO	YES	YES
Group nodes	NO	YES	YES
Set alerts	NO	YES	YES
Download reports	NO	YES	YES
Edit user profile	NO	NO	YES
Higher configuration settings	NO	NO	YES
Create other users	NO	NO	YES
Manage site overheads	NO	NO	YES
View site activity	NO	NO	YES
View user logs	NO	NO	YES

Table 1 Defining user access

3.4 Software Development Methodologies

Before development of any system, setting a development methodology is crucial to the systems success. Before development began I researched two methodologies that would suit this project the most, agile (Scrum) and waterfall were the most logical options.

Using the scrum methodology would allow for development updates during set intervals, along with testing on the updates inside the given interval. This would allow me test as I developed effectively. Scrum also allows me to prioritise and analyse upcoming task and situate them as I need.

Using the waterfall methodology allows me to have a very set plan of action in a sequential manner. Waterfall allows for a steady flow of development at one step at a time, which eliminates chop and changing through different development cycles. A typical waterfall could follow through as such, requirements, design, implantation, verification and maintenance.

In the end, waterfall was the chosen methodology as it is easy to manage due to its rigidity of flow. Waterfall is also more suited to smaller projects and were project requirement are well understood, as in this project.

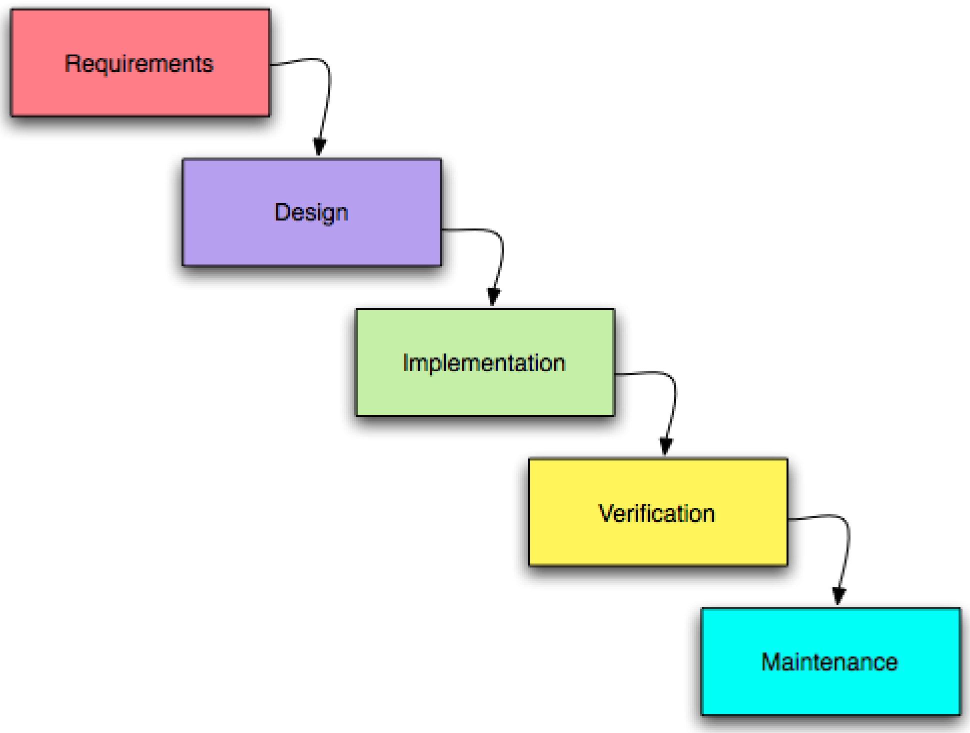


Figure 17 Waterfall Model

3.5 Conclusion

This design section acts as a preview to the implementation section and allows for the visualisation and explanation of the design process of the application and for an overall grasp of the intended architecture.

4 Implementation

4.1 Introduction

This section will walk through the entire implementation of the application including all coding and functionality development. This section will also show testing implemented in the application.

4.2 Cloud Monitoring Project

The cloud monitoring project source is made up of five packages. Four of these packages are made up of the main functionality, these packages are controller, database, model, and service. The fifth package is the FlotJF java wrapper for the Flot graphing library. The project also contains all the front-end packages such as JavaScript packages, along with the JSP and html pages. This section will involve going through main functions of the application and explain their implementation in great detail.

4.2.1 Monitor Management

Inside the monitoring management page the user can view and edit already configured service monitors, resource monitors and proxy settings. The user can also stop and start monitors. The user can also add and edit agents for resource monitors.

The functionality is built up of servlets that request to and from the monitor_management.jsp page. The servlets then use there corresponding service which is used to interact with the database.

Much of the functionality in each servlet and service has many similarities in terms of functionality and structure, however, due to extensibility and future palling reasons decided not to integrate these classes into a one “do it all” class.

Each servlet contains a doPost() method which is used to take requests from the user, it also fills variables that are taken from the view, these requests can come from inputs or session variable sets as the user logs into their account.

These variables will tell the servlet whether the user wishes to edit or remove the monitor or proxy, or whether in fact they wish to run the monitor.

The servlet then communicates with the service so it can update or save to the database to update the monitor or proxy.

4.2.2 Monitor Configuration

Inside the monitor configuration class the user can set up and create service monitors, resource monitors and add agents for any resource monitor by specifying the resource monitors id.

Monitor configuration has the same class structure as monitor management. The functionality is also built up of the same servlets and services that the monitor management uses. However, the monitor configuration can only access the creation of monitors and proxies. It specifies access to create monitors and processes by making sure the when the request is being processed that the 'create' variable inside the servlet is not null. This lets the servlets know that the user is creating an object. This is then sent to the service to check whether or not that monitor exists or not using isServiceExists() or isResourceExists(), if not, then it saves the newly created object to the database.

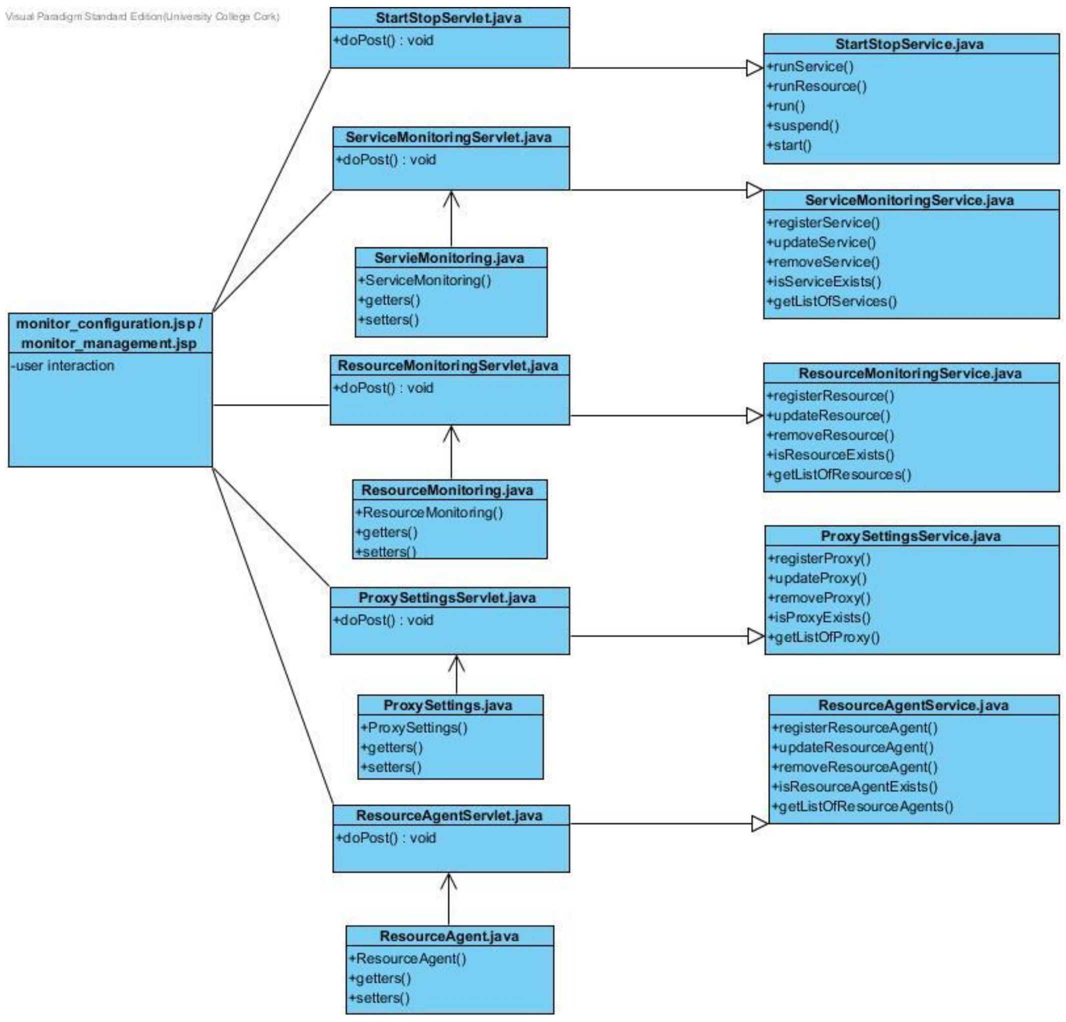


Figure 18 Monitor management/Monitor configuration class diagram

4.2.3 Activity Log

The activity log allows an admin user to view all user activity. The user activity logged includes logins, create actions, update and remove actions performed by the user.

An action is logged and saved when the user performs an action. The logging usually takes place inside a servlet after a request is passed. For example when the user creates a service monitor after the service monitor object has been saved to the database the servlet then logs the action by calling the log() method in the Logger.java class. It is then in that function logged and saved to that user local file in the eclipse directory path.

Then when the admin user wants to see all user activity, they access the report management JSP page. All user files are read and presented. This done by calling the read() function in the Logger.java class.

The Logger class uses BufferReader and BufferWriter to read and write to the text files.

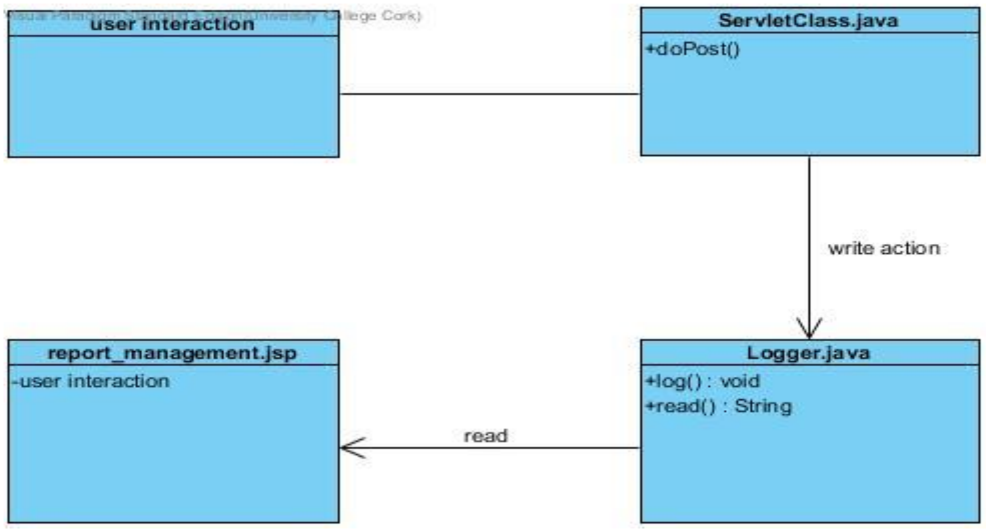


Figure 19 Activity Log class diagram

4.2.4 User Management

User management allows an admin to view and edit all users. From here the user can edit all user parameters and remove the user. This class structure also uses servlets to deal with an update or remove request. From the servlet passes on whether the service should update or remove that specific user object from the database.

4.2.5 User Configuration

User configuration uses the same class structure as user management. With user configuration the user only has the option to create a user. From the user configuration page the user sends the request to create a user and this makes sure the ‘create’ variable is not null, so it can access the create user functionality. This then sends the request to the RegisterService.java class, here the service checks if a user already exists with that same name by using the isUserExsists() method, and if not it saves that user object to the database.

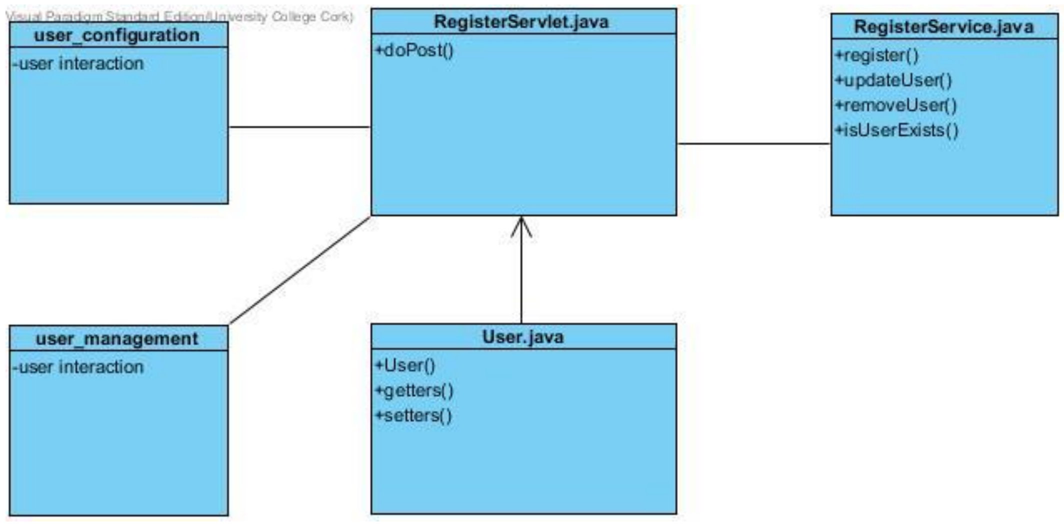


Figure 20 User management/User configuration class diagram

4.2.6 Alert Configuration

Alert configuration allows the user to create alerts so that they can be alerted if a certain metric falls bellows their set threshold.

This is simply done by the user sending the completed alert form, this then sends the request to the AlertServlet, and then the servlet sends it to the service to check if the alert already exists, if not it then stores that alert object in the database.

4.2.7 Alert Management

Alert management allows the user to update and remove already configured alerts. This is done as like other service, where depending on the request to the AlertService will either call update functionality or remove functionality. Depending on the request the selected alert object will either be removed or updated by the servlet calling its subsequent service which will deal with all the database practicalities.

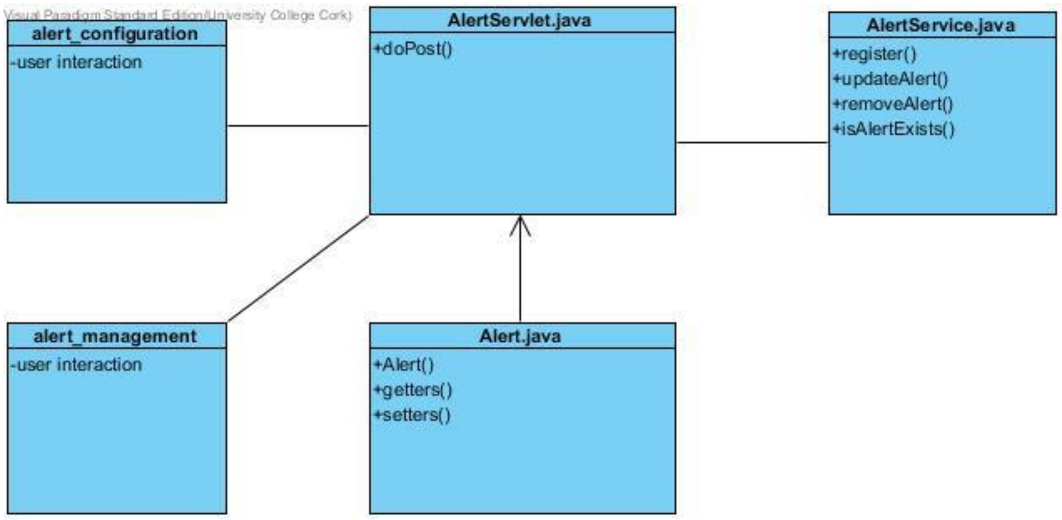


Figure 21   Alert management/ Alert configuration class diagram

4.2.8 Sending Alerts

Once a user sets an alert it will then be checked, whenever the checkAlert() function is called. At the moment these alert check are done manually when certain pages are loaded, as there no proper environments set up to those checks automatically.

Once the checkAlert() function is called it then runs through every alert set by that user in the database, and if that alert breaks it threshold it will then send off an alert to that users email address by using the sendEmail() method in the EmailUtility.java class. The email sending is done using the JavaMail api library using Googles smtp servers.

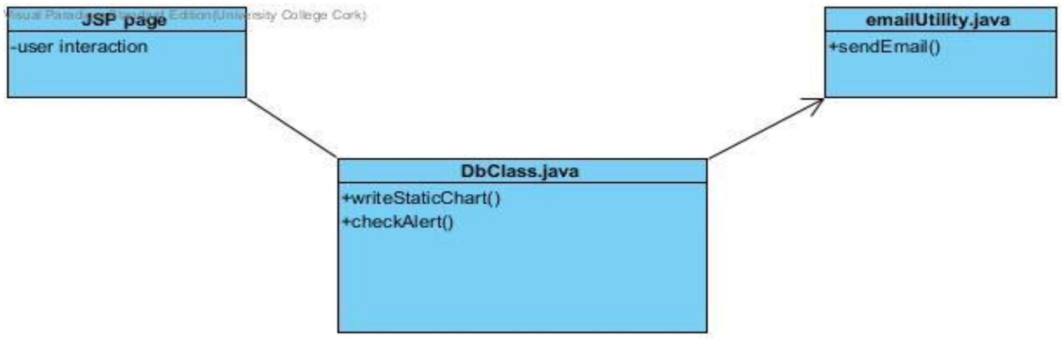


Figure 21   Sending alert class diagram

4.2.9 Dashboard

The dashboard allows the user to graph both static and real time graphs from chosen metrics that were previously configured on monitor configuration page. This page also allows the user view the hierarchy of their configured machines.

If the user wishes to graph a real time chart they must chose the location they wish for the chart to go, then click on the metric in the hierarchy table.

If the user wishes to graph a static graph the user must chose the position of the graph and also chose the two dates they wish to graph between and then click on the metric in the hierarchy table.

4.2.10 Static Graphing

When the user wishes to create a static graph it uses the HighCharts api library which is a solely JavaScript library, all of HighCharts functionality can be found in the monitoring.js file.

When the user clicks the metric they wish to graph and has two dates selected it sends a request using JavaScript function 'start()' , in that JavaScript function it calls the HighChartServlet class using an Ajax request and passes through the two dates selected as parameters, which from their requests the data from the method writeStaticChart() method in the DbClass.java class. The writeStaticChart() method then queries the database for data for the metric between the two dates. If there is data returns it populates the ChartModel.java model class.

Those models are then passed back to HighChartsServlet as an arraylist of data points. Then it is parsed manually to a JSON string and passed back to the HighCharts function waiting in the frontend. It is then graphed and presented with requested data.

4.2.11 Real-Time Graphing

Real-Time graphing is done using an api library called flot.js and a java wrapper FlotJF.

To create a real time graph the user must select the metric they wish to graph without selecting any dates. When the user clicks on the metric they wish to graph, it fires off a request to the JavaScript start() function from here the request is stored to a multi-dimensional array, once stored the update() method is called. This update() function updates each member in the array for whatever number is the set interval. In this update() function the getGraph() and getGraphOptions functions is called every time the update() function loops.

In the getGraph() function it calls the servlet class GetGraph.java, while passing through the needed parameters such as the machines name and the metrics name. Once called the GetGraph.java servlet calls the function getGraph() inside the GetGraphData.java class. This method returns many pieces of data such as the current time, the next data point and moving the chart line. It also calls the FlotJF api for creating the chart object and using its inbuilt methods.

The GetGraphOptions servlet is also called which is used to call the getGraphOptions() method inside the GetGraphData.java class. This method returns data such as grid data and axis data for the chart.

The data from the two method getGraph() and getGraphOptions() is then passed back to the respective servlets and from there the data is passed back to the charting function to update it.

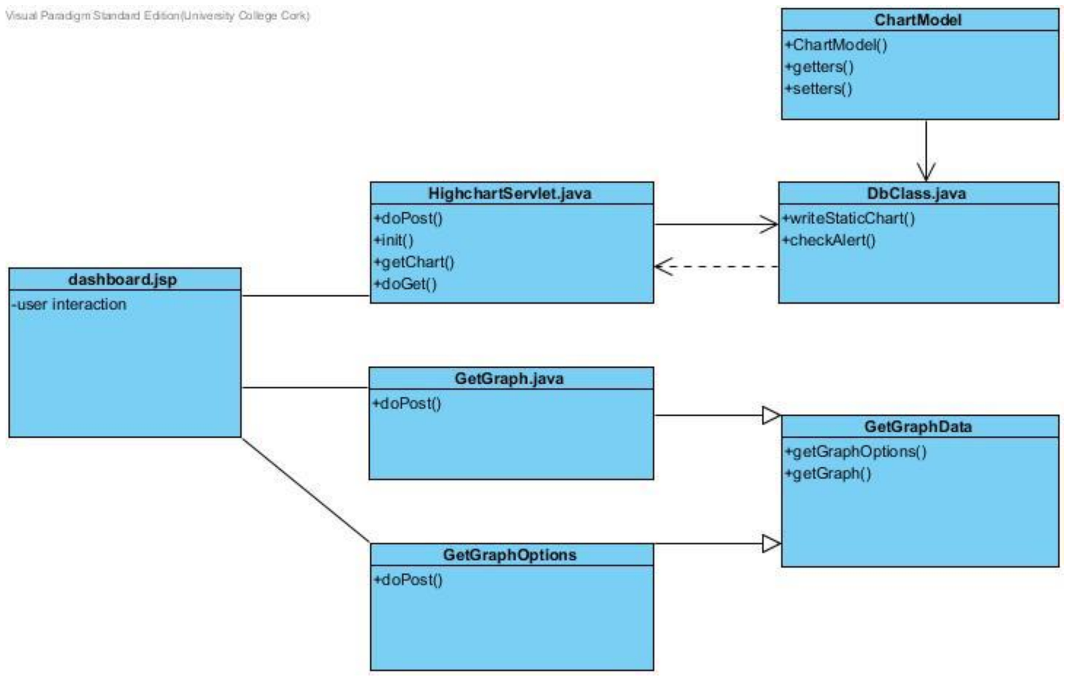


Figure 22 Dashboard/ Graphing class diagram

4.3 Programming Language & Technology

4.3.1 Java

Java is the primary language used in the development of this project as it was a requested pre-requisite for the project and I also have previous experience using the language.

Java is an extremely popular and well renowned object-oriented programming language used by some top industry names. Java gained this popularity from its versatility with use of interfaces, inheritances and polymorphism to name a few. Thanks to this popularity Java has a wealth of knowledge online along with great plug-in and API development.

4.3.2 Servlets

Java servlets are java classes that deal with requests with their doGet() and doPost() methods. They can be used to send data to other Java classes and databases, or receive data from classes

4.3.3 JSP

JSP also known as JavaServer Pages is a technology used in building dynamic and platform independent web-based applications.

JSP is written in the page/view itself, amongst the html, it also has the same syntax as the Java language itself, with a just a few small differences.

JSP can be compared to PHP or ASP.NET's Razor in the way it interacts on the client side.

4.3.4 JavaScript

JavaScript is the secondary language of the project. JavaScript will be used to deal with all the client side intricacies. A lot of the page interactions that cannot be done html or CSS will be done with JavaScript and/or JQuery. Also, the bulk of the graphing will be done with JavaScript using JavaScript graphing API's and AJAX for requesting the data for the graphs.

4.3.5 MySQL &Hibernate

MySQL is the primary database language. The application will use mySQL to query the database to access or return data.

The Hibernate framework is used to populate the database from Java classes once the application is running. The application also uses Hibernate to save, update and delete records from the database

4.3.6 Bootstrap

Bootstrap is a free and open source web development framework that incorporates technologies such as HTML, CSS and JavaScript. It uses all of these components to create an easy to use, clean and responsive user experience.

4.3.7 MVC

The software engineering architectural pattern being used is a Model-view-controller. The view will be the display that the user interacts with, the controller will act as the brain and model will give the blueprint of the actions.

The user interacts with the view, then based on the interaction will call a certain controller. The controller then takes the input and passes it on to either the view or model.

From there a view can display the returned command or a model can do an array of actions such as storing or adjusting the data.

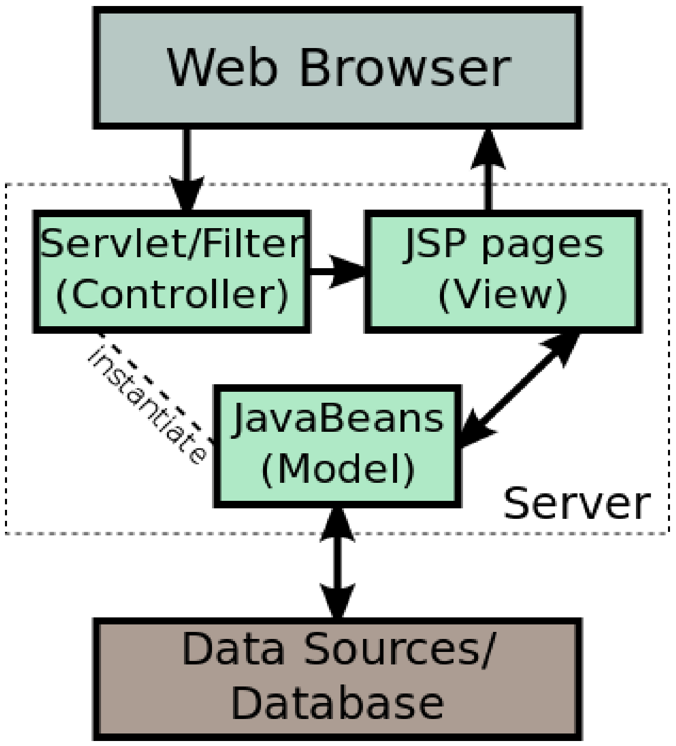


Figure 23 MVC diagram

4.4 Libraries

4.4.1 HighCharts

HighCharts is a pure JavaScript graphing library that allows for a range a graphing operations. HighCharts graphing library is used to graph the static charts is the application.

The reason for choosing HighCharts as the graphing library for static graphs is because of its great plug-ins and extreme flexibility. HighCharts offers a host of functionality that is not seen in many other graphing technologies, such functionality as being able to drill into graph data, export graph images and its detailed graph animations.

All these reasons along with well supported community made it the top choice for static graphing

4.4.2 Flot & FlotJF

Flot is a pure JavaScript plotting library for jQuery while FlotJF is a Java Framework to generate the JSON data for use with Flot. FlotJF was built to simplify the process of generating graphs with J2EE applications using Flot **[9] [10]**.

Flot and FlotJF are used for real-time graphing. FlotJF is used as an api to create graphs and supplying graphing data in the backend using Java classes, which I then rendered in the front-end.

The reason for using Flot and FlotJF came down to the benefit of how lightweight it is. Flot is a small charting library, however, whatever it lacks in size and functionality it makes for in speed and extensibility.

As graphing real time can be a strain on the application due to the sheer amount of requests it was important to choose an extremely lightweight charting api.

4.4.3 JQuery

JQuery is a well-known and popular JavaScript library used for anything and everything. The main uses of JQuery in this application are web page functionality such as page animation and also the use of AJAX for grabbing data asynchronously from the backend.

The reason for using JQuery is because of its huge array of functionality, along with large online community and well documented plug-ins.

4.4.4 Validator.js

Validator.js is a free and open source library used for client authentication of inputs and forms. Validator is used especially for Bootstrap application and provides real time asynchronous authentication **[11]**.

The reason I choice Validator.js was due to the fact it was specially built with Bootstrap in mind. There are many other JavaScript validation libraries that do just as good as job as validator.js on normal pages. However, with Bootstraps animations such as modal and accordions, many of these other validations library wouldn't work due to JQuery only validating what the page can see when it loads. When a modal pop up was called many of them were unable to validate it. Validator.js on the other hand was able to overcome these issues.

4.4.5 JavaMail

JavaMail is a free java api used for the automation of sending email. This is used in the site for alert functionality. Anytime a configured threshold is breached it will send an email to the owners email address.

The reason for choosing JavaMail is because it is the well renowned mail library for Java development. Due to its popularity means there are many examples to its use, which gave it the biggest advantage as many other mail libraries are often not popular or under documented **[12]**.

4.4.6 Apache Tomcat

Apache Tomcat is an open source software implementation of the Java Servlet, JavaServer Pages, Java Expression Language and Java WebSocket technologies **[13]**.

Tomcat is used to run and host java web application locally inside eclipse. Tomcat had to be used as it is the only server compatible with the eclipse IDE.

4.5 Testing

4.5.1 JUnit Testing

JUnit was the test framework used to test functionality of the application. JUnit was the obvious choice for testing as the whole backend of the project is built in Java.

JUnit tests were written for specific methods that dealt with functionality in the backend. These tests were written in a test class that allowed use of the assert method to check the expected result against the actual result.

4.5.2 Regression Testing

Every two weeks a full application regression test would be implemented. This regression test consisted of testing all old functions over again before the two weeks along with the new functions implemented within those two weeks. If any bugs or faults occurred, they would be dealt with before testing finished on other functions. This insured that older functions remained compatible with newer updates to the program.

4.5.3 Client Side Testing

JavaScript code was tested as functions were implemented. Testing was conducted using the Chrome browser in-built console and debugger. It was ensured that all possible branches could be explored and that the program threw no errors with any given input.

4.5.4 Browser Compatibility Testing

In order to ensure that the application would be accessible to multiple different browsing platforms and be able to make users aware of browser functionality differences it was important to do full application usability tests on different browsers.

The application was developed and tested solely using the Chrome browser, due to this the application works as intended and as expected

When using the application on windows version of Firefox, the applications functionality is unchanged and the styling has slight changes, but nothing of serious note. However, when testing the application in a Linux environment testing in Firefox had some more serious changes to JavaScript animations and functionality, this made some the applications function un-useable. Errors such as not recognising a date selection input stop the user from being able to create a static graph.

When testing the application on Internet Explorer the site functionality was fine, although, the styling was changed quite a bit and left the application not an astatically pleasing as Chrome or Firefox versions.

4.5.5 Usability Testing

To test the usability of the application, a small test was set up so that a group of users could do specific actions within the application. This was done in order to test how easily the application was to use and understand.

These results will showcase the difference between a user with an in-depth knowledge of the application and a new user. The results below show this disparity.

Instruction	**Developer**	**User1**	**User2**	**User3**	**User Average**	**Difference**
Configure a resource monitor & add one agent to it	15	23	26	22	21.5	+6.5
Start a service monitor	3	5	6	5	4.75	+1.75
Create a static graph between the 9th and 10th of November	8	15	22	18	15.75	+7.75
Create a real time graph and then remove it	5	10	12	8	8.75	+2.75
User Average:	7.75	13.25	16.5	13.25	12.6	+4.85

Table 2 Usability test results

4.6 Issues & Workarounds

4.6.1 Graphing

Throughout the project the biggest challenge I faced was the graphing, and in particular the real time graphing.

Originally it was planned to use the graphing library for both the real time and static charts. The static chart was implemented first using HighCharts, with relative ease after researching and learning its api. I then started to develop real time graphing using HighCharts, this however lead to many problems due to how I was handling requests using servlets. After researching this problem, I found little to know help, even after contacting HighCharts developers on their forum, they noted their lack of knowledge of use with JSP and servlets, and acknowledged the issue, their advice was to change the way in which the JSON was being returned from the servlet. After formatting JSON everyway known possible, there was still no solution.

To stop this issue taking up even more time, I decided to cut my losses and move onto a different charting api for real-time graphing and keep HighCharts for static charting.

After researching Flot.js and FlotJF I began implementation, integration went smoothly after learning its api.

4.6.2 Dashboard Functionality

One small issue I came that was not in the original design was setting graph location. The original idea consisted on clicking on the graph the user wished to graph and then that graph would go into the next free graph spot, however due to complexities in this approach and lack of time to implement it, I was forced to create a simplified version that allows the user to choose the graph location before graphing. This also need to be done when deleting the graph.

When given extra time, I feel like the original concept could be implemented with relative ease.

4.7 Conclusion

This section showcases how the application was developed by using class diagrams to demonstrate how the classes interacted and show the methods used in the independent functionality.

5 Evaluation

5.1 Introduction

This section is used to evaluate the entire project and give a personal, honest and detailed appraisal of how well the project met its initial vision. This section will also go over the final solution of the project and show examples uses.

5.2 Work Completed

Through the completion of this project it shows that the majority of the main functional requirements of this project were met.

The application allows the user to:

5.2.1 Monitoring Functionality

- Start & stop monitors with ease through the monitor management of the application.
- Configure and group monitors through both the monitor configuration and monitor management pages.
- Add agents to specific resources in both the monitor configuration and monitor management.
- Completely edit and configure already existing monitors through the monitor management page

5.2.2 Alert System

- To configure and setup alerts through the alert configuration page
- Edit and remove already configured alerts
- Send email alerts to user when alert breaks threshold

5.2.3 Admin Functionality

- Allow admins to create users of any type (normal and admins).
- Allow admin users to edit and remove other users.
- Allow admin users to edit all configured monitors, regardless of original creator.
- Allow admin users to access all user log activity.

5.2.4 Dashboard Functionality

- Create both real time and static graphs
- Drill down in static graph data.
- Add and remove graph to and from the dashboard

5.2.5 Report Management

- Allow admin access to all user's activity

5.3 Work Uncompleted

5.3.1 Integration

Some of the functionality within the application needs to go through more rigorous testing, as the main idea of the project was to integrate it into an OpenStack environment, this however was not possible under the given time constraints. Meaning a lot of the functionality was tested by passing my own data.

Functionality that must be tested include passing genuine real time data from an actual virtual machine to a graph. Along with other functionality such as configuring a monitor hosted on a cloud environment.

Although much testing is needed I am confident that my code will integrate well as I have built it with extensibility and configurability in mind.

5.3.2 Search Functionality

One of the design features that was highlighted in the functional requirements was the ability to have a full site search function. But due to the architecture to support not being designed in time due to time constraints it was not possible to complete this task, although I do feel at the moment it is not a function of huge importance, I do feel as the application grows it will be necessary, and will definitely go down as future work.

5.3.3 Alert Functionality Testing

Due to UCC proxy server restriction of access to the Gmail smtp client I was unable to complete intensive testing on the functionality. However, on my own machine I was able to complete a couple of scenario tests and fix any issue accordingly. That said, more testing I needed before there is full confidence in the function.

5.3.4 Exporting Graphs

Exporting Graphs was a functional requirement that was not met, this was due to time constraints and prioritising functionality, unfortunately this was not a priority.

Given extra time I am more than confident in my ability this implement this functionality.

5.4 Public Evaluation

In order to evaluate the usability aspects of the project a questionnaire was provided to testers. Testers consisted of my computer science colleagues who have obvious technical knowledge and other students outside of the department with little technical knowledge.

In the questionnaire they were asked to rate the questions relating to functionality from 1 – 10. Below shows the result of the questionnaire.

Rate how easy was the application to use?

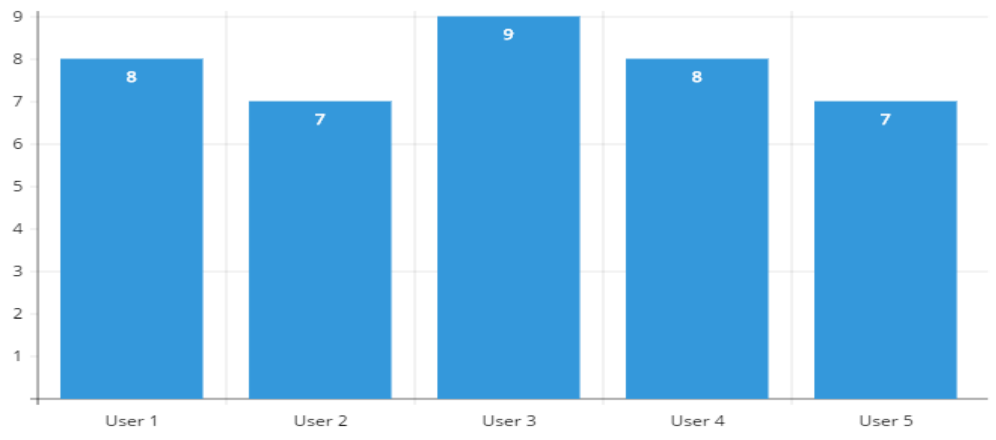


Figure 24 Questionnaire 1

Rate the overall look and feel of the application?

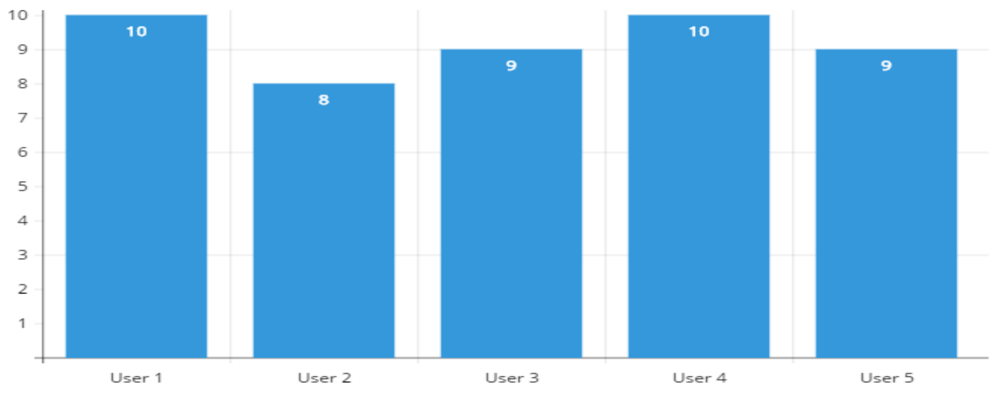


Figure 25 Questionnaire 2

Rate how easy the application was to understand?

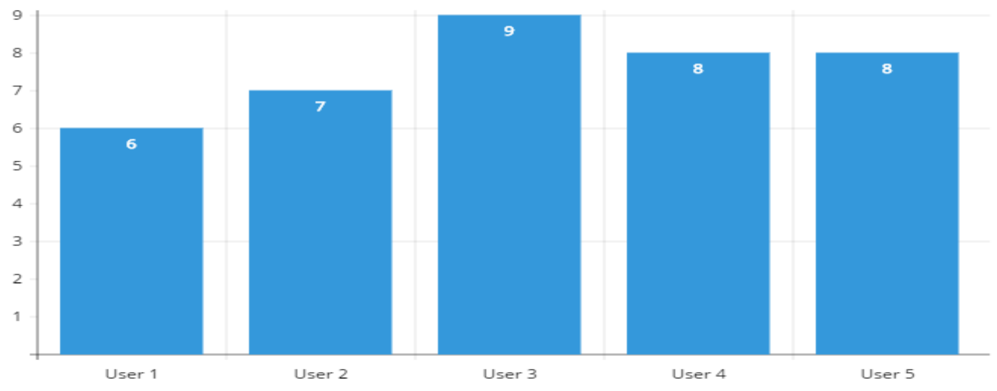


Figure 26 Questionnaire 3

Rate how responsive the application was?

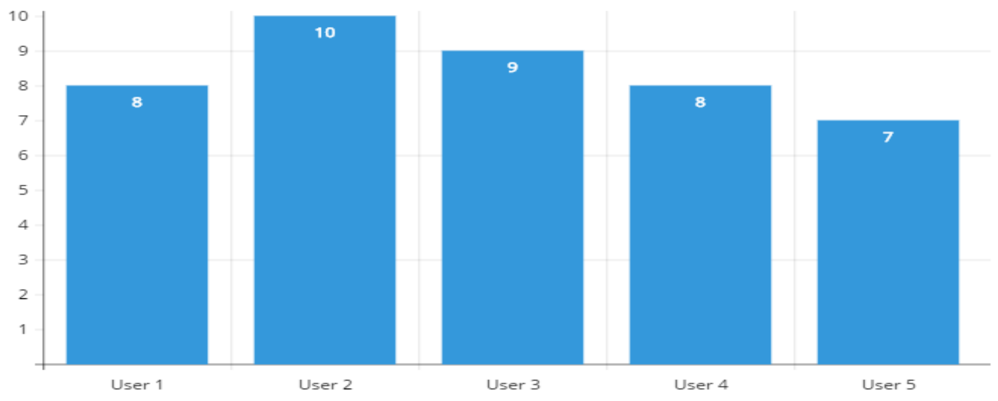


Figure 25 Questionnaire 4

5.5 Use Cases

A use case describes a systems behaviour as it responds to a user's interactions with the system. This helps represent and visualise how a user interacts with the system and steps needed to perform actions.

For the sake of saving space in the report I am only referencing one use case, there is additional use cases for both admin and dashboard functionality. These other use cases can be seen in the appendix.

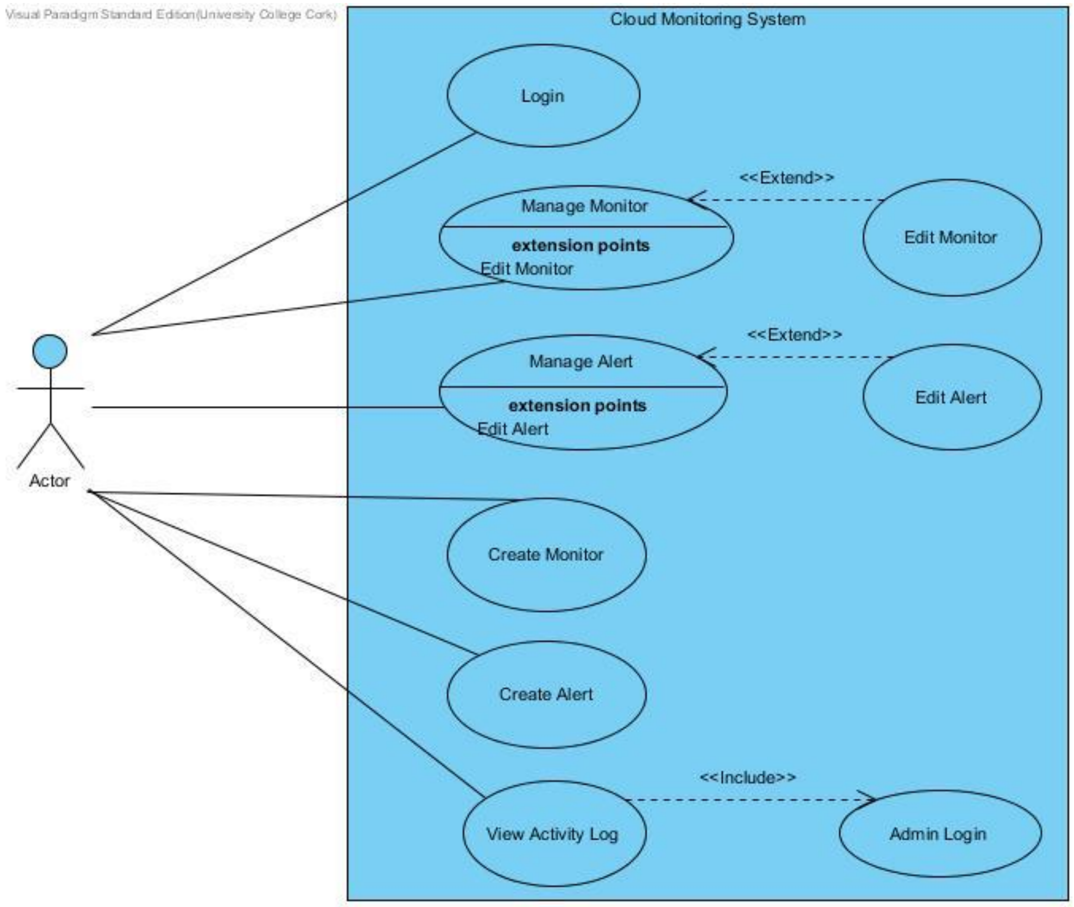


Figure 24 Main application functions use case diagram

Use Case: Create Monitor

Actor: User

Steps:

- User connects to application
- User logins into the application
- Goes into monitor configuration
- User creates either a valid service or resource monitor
- Monitor response to user that the monitor has been successfully created

Use Case: View Activity Log

Actor: User

Steps:

- User connects to application
- User logins into the application as an admin

- User enters the report section
- User views all user's activities

Use Case: Edit Alert

Actor: User

Steps:

- User connects to application
- User logins into the application
- User creates an alert in Alert Configuration and saves it
- Users goes to Alert Management and edits the alert

5.6 Conclusion

This evaluation section showcases how the application turned out with an emphasis on what on what was achieved and what wasn't achieved from the initial vision of the project.

6 Conclusion

6.1 Introduction

This section goes through a summary of the project, including what the overall feel for the project is. This section also goes through future development and what has been learnt throughout the lifespan of the project.

6.2 Project Summary

The constant progression of work can be seen from the very beginning to the very end. Going through to analysis where research into existing technologies and requirements were set. To design were first draft designs and functionality was set. Then onto implementation which highlights the work that went into developing the main features by showing the relationship between classes as their described, and then testing the functionality of the application in a technical and interpersonal sense.

This project progressed at a steady and reasonable state, constantly evolving into its intended vision, and although the project did not have 100% of the features desired in the initial vision, overall it is felt that the project was a success.

6.3 Future Work

Apart from finishing the uncompleted work stated in the evaluation section above. Further advancement in terms of project evolution could involve additional functionality that was not highlighted in the initial vision of the project, such as an intricate notification system that could take over some of the alert functionality.

Future work could also consist of a standalone phone application. This smartphone application could provide all the functionality of the web application but in a small and simplified manner. This would then allow user the ability to receive notifications through their smartphone, it would also make it easier for the user to conduct all functionality due to simplified layout.

6.4 Lessons Learned

Through working on this project I have learned many things from technical and personal aspects. Working on this application has tested my coding and software development skills, it has also helped me to learn new technologies I would never have thought I was going to learn, some of which I am sure will be integrated in future projects. Due to the amount of software development associated with project I feel as though I have grown as a coder and software developer.

This project has also improved my personal outlook on what can be achieved when hard work is used. I was able to achieve much more than I thought was possible during the lifespan of the project. Due to this I am know much more confident in my abilities when confronted with a daunting task.

This project has also improved my ability to conduct research. Before, research would always have been considered an afterthought as many other projects I had worked did not have such a scale as this. Now I truly value my new found research skills, as I now truly know its worth.

For all these reasons and more I feel have developed into a better computer scientist than I was 9 months ago.

7 Bibliography & References

1. Uptrends Infra, https://www.uptrendsinfra.com/
2. Monitis, http://www.monitis.com/
3. Stackdriver, http://www.stackdriver.com/
4. Graphite, http://graphite.wikidot.com/
5. Cacti, http://www.cacti.net/
6. D3.js, https://d3js.org/
7. HighCharts, http://www.highcharts.com/
8. JFreeChart, http://www.jfree.org/jfreechart/
9. Flot.js, http://www.flotcharts.org/
10. FlotJF, http://dsysadm.blogspot.ie/2012/02/flotjfexamplesweb-how-to-use-flotjf.html
11. Validator.js, http://1000hz.github.io/bootstrap-validator/
12. JavaMail, http://www.oracle.com/technetwork/java/javamail/index.html
13. ApacheTomcat, http://tomcat.apache.org/

8 Appendix

8.1 Other Use Cases

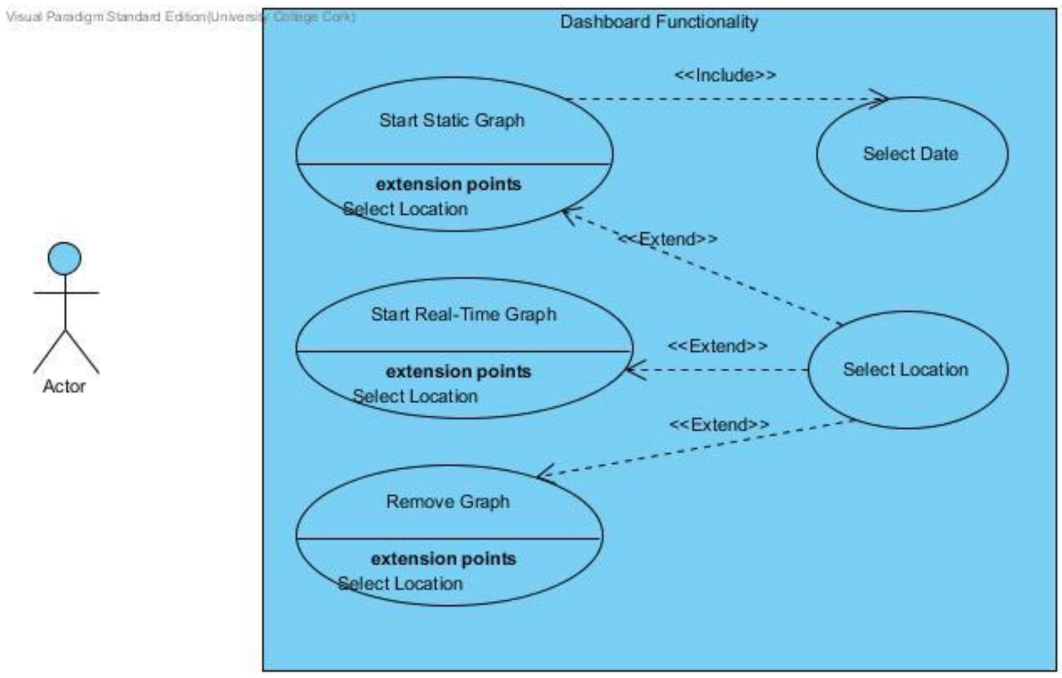


Figure 25 Dashboard functionality use case diagram

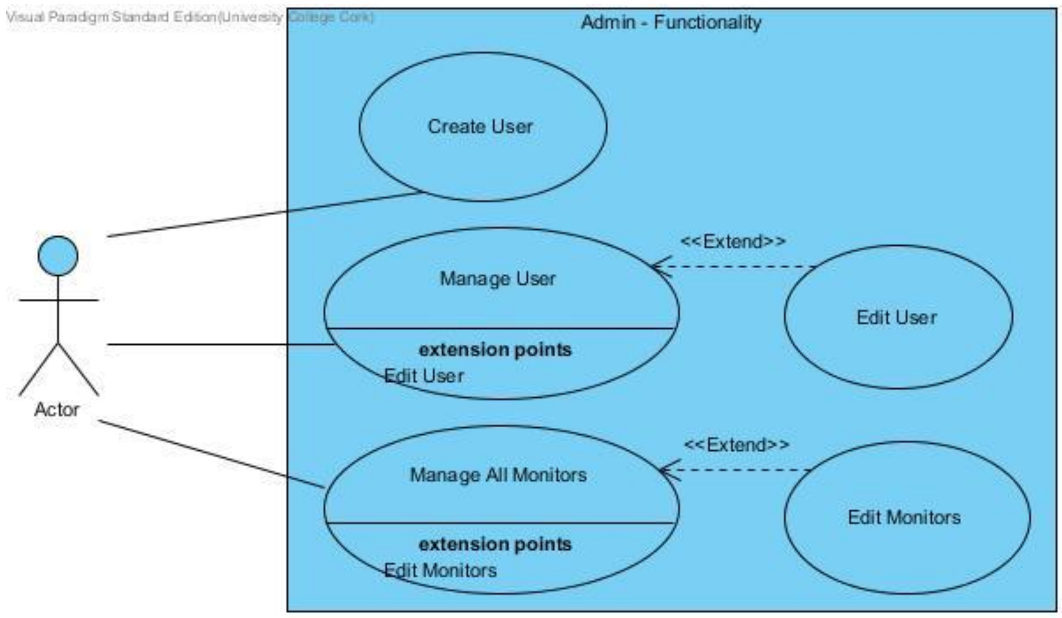


Figure 26 Admin functionality use case diagram