UNIVERSITY OF NAIROBI

SCHOOL OF COMPUTING AND INFORMATICS

AN AGENT BASED SYSTEM FOR MONITORING AND MEASURING PERFORMANCE OF PUBLIC CLOUD APPLICATIONS FOR USERS

By

NAOMI NJOKI KIGO

A RESEARCH PROJECT REPORT SUBMITTED IN PARTIAL FULFILLMENT FOR THE REQUIREMENTS OF AN AWARD OF THE DEGREE OF MASTER OF SCIENCE IN DISTRBUTED COMPUTING TECHNOLOGIES OF THE UNIVERSITY OF NAIROBI

JULY 2016

DECLARATION

I **Naomi Njoki Kigo** do declare that this research project report is my original work and has not been presented for any award in any other university.

_____ _____

Naomi Njoki Kigo Date

P53/72933/2014

I **Dr. Elisha Opiyo** of the University of Nairobi do confirm that this research project report has been presented for examination with my approval as the University Supervisor

_____ _____

Dr. Elisha T. Opiyo Omulo Date

School of Computing and Informatics

ACKNOWLEDGEMENT

I take this opportunity to thank God for His grace and provision of good health and resources throughout my masters' course. I am thankful to my family and friends who showed me great support, encouraged me and were patient with me during the period of my course. I wish to appreciate my supervisor Dr. Elisha Opiyo for his support during this project. He gave me guidance on how to conduct this research and was very resourceful during project execution. My supervisor's advice and comments made writing of this project successful.

ABSTRACT

Monitoring of public cloud applications is a task of paramount importance for both Providers and consumers. The users of cloud and the providers are tied by service level agreements (SLA) that outline the provision and usage of the service. For the cloud application users' application performance and minimum down times are key performance indicators of great concern in any SLA, noting that users of public cloud have no control over the cloud application. There are third party commercial monitoring tools used to monitor cloud application performance parameters mainly provided by the cloud providers. These tools run on the applications providers' servers and measure performance from the applications side. The commercial tools are provider specific and may not present enough details for clients to be able to customize the behavior of the tools to accommodate their specific requirements. The tools may also not capture the users' true experiences of the cloud application performance. This study proposes an agent based system that uses agents to interact with the cloud application and measure performance of the application as captured during the user agent interactions. The system measures performance as perceived by the end user hence reflecting the true user experience from the users' side. Users' perception of performance namely timeliness, availability and scalability were measured in this project. The researchers approach used user agents that interact with the cloud applications on behalf of the users and an intelligent monitoring agent that captures transaction durations during user interactions. In this project Prometheus Methodology of developing agent based systems was used for system specification, architectural design and detailed design. JADE framework was used in implementation of the system more specifically development of the agents. The results of the system indicated that agents were able to interact with the cloud application on behalf of the users and during the interactions the monitoring agent was able to capture and measure transaction durations of the cloud applications automatically and continuously updated the database. The transaction durations measured were interpreted as performance measures namely system availability, system scalability and application response time. These measures were displayed on a graphical interface and allowed users to access real time and historic data for monitoring of performance of cloud application.

TABLE OF CONTENT

LIST OF ABBREVIATIONS

SaaS: Software as a Service

SLA: Service Level Agreement

KPI: Key Performance Indicators

QOS: Quality of the service

PC: Personal Computer

TCO: Total Cost of Ownership

PaaS: Platform as a Service

IaaS: Infrastructure as a Service

PDT: Prometheus Design Tool

AOSE: Agent Oriented Software Engineering

GUI: Graphical User Interface

CIO: Chef Information Officers

IT: Information Technology

RUM: Real User Monitoring

API: Application Programming Interface

JADE: Java Agent Development Environment

LIST OF FIGURES

LIST OF TABLES

CHAPTER ONE: INTRODUCTION

1.0. The Background

Due to severe market competition and dramatically changing business environment, firms have been prompted to adopt various state-of-the-art Information Technologies to improve their business operations Imran (2010). Cloud computing is one of such technologies being adopted by organizations.

Cloud Computing refers to both the applications delivered as services over the Internet and the hardware and systems software in the data centers that provide those services. The services themselves have long been referred to as Software as a Service (SaaS). The Data center hardware and software is a Cloud. When a Cloud is made available in a pay-as-you-go manner to the general public, we call it a Public Cloud. People can be users or providers of SaaS, or users or providers of Utility Computing Armbrust et al, (2009)

Monitoring of Cloud is a task of paramount importance for both Providers and consumers. On the one side, it is a key tool for controlling and managing hardware and software infrastructures; on the other side, it provides information and Key Performance Indicators (KPIs) for both platforms and applications G. Aceto et al, (2013)

Both users and providers are bond by a Service Level Agreement (SLA) that specifies the terms of offering and of using the cloud services. Monitoring of the services is beneficial to both the users and the providers. The providers monitor the cloud services to check on possible SLA violations on the usage of the service whereas SaaS users benefit from monitoring the services to check on the performance they pay for and possibly renegotiate the scope of their SLA.

As businesses move their mission critical applications to the public cloud, they lose some or all control of the application. They would therefore be interested in monitoring the performance of the application. Application performance parameters have to be included in the SLA's agreements made between the users and application providers. The ability to evaluate performance of a cloud application during selection of a cloud provider and during normal operations of the business (use of the application) is of key importance to the business stakeholders.

1.1. Problem Statement

There exist third party tools for monitoring cloud services. However according to S. Moustafa, (2015), a major drawback of these commercial monitoring tools is that they do not present enough details to consumers to be able to customize the behavior of these tools to accommodate their specific requirements. In addition, such monitoring tools typically are provider-dependent.

The monitoring tools available are installed and run on the application servers. Performance measurement and monitoring may therefore not capture the true user experience from the end user's side.

A majority of these tools run on real systems and require real users to interact with the cloud application, hence clients may not use them in a test environment (where the exact number of real users are not available) to evaluate the performance of different cloud applications before or during the selection of a cloud provider. This is particularly important since the business environment is ever changing and therefore business operators may have a need to test the appropriateness of the tool in these changing environments.

Most applications designed to be hosted in a cloud tend to have monitoring solutions built into the application itself Infosys (2012). In this regard such monitoring solutions may not be used to monitor other applications provided by other providers.

The proposed agent based system provides a solution for monitoring and measuring the cloud applications performance from the end users. The study demonstrates the use of agents to interact with a cloud application on behalf of the real end users and hence can be used in an environment where the actual users are not available. The system also uses agent to monitor performance of the cloud application by capturing performance parameters during user interactions; true user experience is therefore captured since the agents run on the client's side. The solution is not cloud provider specific and can be used on different cloud applications provided by different providers. The solution can be customized to monitor different cloud applications.

1.2. Goal of the Study

In this study the main goal was to use agents to interact with a cloud application on behalf of real users and demonstrate the use of agents to measure performance parameters from these interactions.

The study aimed at capturing the end users experience on performance of the cloud applications without necessarily using real/live users.

1.3. Objectives of the Study

- Understand user interactions with the cloud application.
- To formulate an agent based system that will monitor and measure performance of SaaS service.
- To assess the system's ability to monitor performance by analyzing the data generated by the system.

1.4. Scope and Limitations of the study

The study was limited to using one cloud application. Also three users were used for simultaneous access to the application. A minimum of three users were chosen to form a simple distributed system environment.

The agents developed in this study were limited for use on a user's personal computer (PC) either desktop or laptop as a device on which user's access cloud application.

The study was conducted on SaaS service model of cloud computing.

1.5. Significance/ Justification of the Study

The study was expected to provide an important system for monitoring performance of a cloud application. The study demonstrated the use of agents acting on behalf of users during cloud application interactions and use of agents to capture performance parameters during the interactions. The agent based system was used to measure application response time, system availability and scalability. The system provides historic and real time transaction durations used for monitoring purposes.

The proposed system will be beneficial to different stakeholders of SaaS cloud technology as indicated below:

- To enterprise **users and system administrators** the proposed system will provide important information on the current and historic status of cloud applications performance and explain user experience.
- To **managers,** the developed system can be used to provide important information for reviewing and valuating service level agreements with their service providers.
- To public cloud **service providers** the study will provide them with a system that can give them important information on user experiences on the SaaS services they offer.

- To **researchers** the study will form a useful material for references to help expand the practice of monitoring performance of applications on a public cloud using an agent based model.
- The study will also contribute valuable knowledge to the field of agents systems as it focused on the new knowledge of using agents to interact with the cloud application on behalf of users.

1.6. Organization of the Report

This report has given a brief introduction of the study by highlighting the background and problem under investigation. The introduction also summarizes the significance of the study to the various stake holders in the industry of cloud computing and research.

In the literature review chapter, the report discusses various literatures on the topic of cloud application performance measurement and monitoring. The various aspects that have been reviewed are overview of cloud computing, software application performance on the public cloud, monitoring and measuring of performance of the cloud applications and aspects of service level agreements on SAAS platform. Agent based systems and other performance monitoring tools have been reviewed in the chapter. The various components of the proposed system architecture have been discussed to depict the expected system model.

In chapter three of the research methodology, agent development methodology used in the project has been discussed. The researcher also discussed research design of the study which includes population sampling, data collection and analysis as conducted in this research.

The analysis and design phase of the study used the promethium methodology. In this phase the system requirements were gathered and the overall system defined. Various design diagrams were used to create a presentation of the final system.

In the implementation and testing phase, the system designs were converted to computer readable programs using programming languages. The developed code was then tested to ensure it conforms the requirements gathered and also to detect and eliminate logical and syntax errors in the code. The system was evaluated by industry experts by use of questionnaires. The data generated by the system was also analyzed and compared to the data collected from the real world study.

In chapter five the researcher makes his conclusion on the results obtained in relation to the study and also gives recommendations for future studies. Challenges encountered during the research period are also discussed.

CHAPTER TWO: LITERATURE REVIEW

2.0. Introduction

This chapter will review literature from other authors that relate to the study. The over view of cloud computing, software performance in the cloud, measurement and monitoring of performance of cloud applications have been discussed. Agent based systems as used in monitoring of performance and aspects of service level agreements of SAAS model have been reviewed. The proposed system architecture elements are explained.

2.1. Overview of Cloud Computing

This study will use the 5-3-2 principle of cloud computing as discussed by (Chou, 2012) to explain the concept of cloud computing. The 5-3-2 principle, describes the 5 essential characteristics, 3 delivery methods, and 2 deployment models of cloud computing.

The 5 characteristics of cloud computing are the expected attributes for an application to be classified as a cloud application. These include, on demand Self-service, Ubiquitous network access, Location transparent resource pooling, Rapid elasticity and measured service with pay per use. The 3 delivery methods of cloud computing are Software as a Service, Platform as a Service, and Infrastructure as a Service, namely; SaaS**,** PaaS, and IaaS respectively. These 3 delivery methods are presented as services in the context of cloud computing. The 2 deployment methods of cloud computing are public cloud and private cloud. Public cloud is intended for public consumption and private cloud is a cloud (and notice a cloud should exhibit the 5 characteristics) while the infrastructure is dedicated to an organization. (Chou, 2012)

2.1.1.1 The 5 characteristics of Cloud

The characteristics are defined as

On demand self -service – services provided on the cloud are on demand; the services are available for use when demanded by the user.

Ubiquitous network access – users of cloud access the resources over the internet which consists of interconnection of many networks.

Location transparent resource – users of the cloud services are not aware of the physical location of the resources that contribute to the deliverance of the services.

Rapid elasticity – cloud resources can rapidly expand based on the number of users and also the resources that used to contribute to the

Measured service with pay per use – users only pay for what they have used

2.1.1.2 The 3 delivery methods of Cloud Computing

Software as a Service (SaaS)

SaaS is a service model of cloud computing that allows users access and use applications from cloud service providers without necessarily installing the applications in their premises. According to (NIST, 2011) SaaS service model is the capability provided to the consumer to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through either a thin client interface, such as a web browser (e.g. web-based email), or a program interface. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user- specific application configuration settings.

SaaS service eliminates the need for buying and locally installing software to users' machines. The maintenance and administrative functions of the applications are also eliminated from the system administrators. It is highly attractive to businesses that don't want to deal with the maintaining or creating of software for their applications.

Infrastructure As A Service (IAAS)

This approach regards delivery of a processor and network infrastructure. The focus is on physical or virtual machines, firewalls, load balancers and network infrastructure. All these deliverables are located in a data center owned by the cloud provider.

Platform As A Service (PAAS)

Platform as a Service (PaaS) is the set of development tools and services designed to make coding and deploying of applications over the web quick and efficient for the customer.
The customer does not have access to the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed tools/services and probably

configuration settings for the application-hosting environment. This provides a set of developer environment that a customer can use to build their applications without having any clue about what is going on beneath the service. PaaS is a platform where applications can be developed, tested and used. Abraham E. et al (2014)

Examples: Google App Engine, Force.com, Microsoft Windows Azure, Java.

2.1.1.3 The 2 deployment methods

Public cloud

The cloud infrastructure is available to general public. It's a cloud that anyone can use like google, amason Public cloud implementation are large cloud implementations around.

Example: Amazon, Google Apps, Windows Azure, as discussed by Sushil kumar, et al (2014)

Services in the public cloud are usually provided to the users over the internet. The resources available for use in a public cloud are on a shared pool; however each organization's data is kept secret.

The cloud infrastructure in a public cloud is owned and managed by the cloud provider. The user has no control of infrastructure and only trusts the provider to provide the resources as agreed. The end user pays for these resources.

This project hence focuses on monitoring performance of public cloud services since the user has no control of the resource availed to him as opposed to the private cloud where the organization has full control of the cloud resources.

Private Cloud

The cloud infrastructure is provisioned for exclusive use by a single organization comprising multiple consumers (e.g., business units). It may be owned, managed, and operated by the organization, a third party, or some combination of them, and it may exist on or off premises (NIST, 2011)

The services available for use in a private cloud are entirely dedicated to an individual organization. The organization controls and manages the entire cloud environment. The enterprises that deploy a private cloud make a one-time heavy investment for the infrastructure that the cloud runs on.

2.2. Software application performance on the Public cloud

This study will discuss cloud software applications offered on SaaS service model of cloud computing.

Cloud Computing has a number of positive aspects pushing for its rapid adoption, from both economic and technical points of view. As for the former, with respect to other service hosting possibilities, Cloud provides a lower Total Cost of Ownership (TCO), an increased flexibility in terms of both resources and Service Level Agreements (SLAs), and allows focusing on the core business, ignoring the issues related to the infrastructure management. As for the latter, Cloud Computing provides an improved scalability, ubiquitous access to data and resources, and advanced disaster recovery mechanisms. Together with these positive aspects Cloud Computing has a number of challenges: (i) provision of scalability, load balancing, service continuity and application performance; (ii) provision and guarantee of SLAs; (iii) management of large scale, complex and federated infrastructures; and (iv) analysis of the root causes of end-to-end performance. In order to cope with such challenges, accurate and fine-grained monitoring and measurement techniques and platforms are required G. Aceto et al, (2013)

J Espadas et al, (2011) found that, in spite of the advantages of using cloud computing to create highly scalable applications, solving performance problems through cloud computing is not a trivial decision if involved costs are analyzed.

Shailesh Paliwal, (2014) implies that performance means responsiveness and scalability. The author notes that “we all know that performance – responsiveness and scalability is a make-or-break quality for software”. He further says that performance of software is a key parameter in determining the quality of the software. Since cloud applications are accessed using an internet connection, other network connectivity issues example latencies, availability and bandwidth capacity may also affect the performance of a cloud application.

SaaS performance measures are directly perceived by users as business transaction response times and throughput, technical service reliability and availability, and by scalability of the applications. The degree to which cloud services can meet agreed service-level requirements for availability, performance, and scalability can be estimated by using performance modeling techniques, so that potential performance anti-patterns can be detected before they happen. In the absence of sophisticated tooling for automated monitoring, the automatic provisioning and usage-based costing

(metering) facilities rely mainly on fine-grained capacity management. Until more data collection, analysis, and forecasting are in place, capacity management is more opportune than ever. Irrespective of sophisticated tooling for automated monitoring, cloud computing users need to analyze their demand for capacity and their requirements for performance. In their contract with cloud computing providers, users should always take a bottom-line approach to accurately formulate their service-level requirements. Shailesh Paliwal, (2014)

M. Armbrust et al, (2009) discusses that the fact that the hardware infrastructure maintenance is delegated to the Providers, the Cloud Computing model is attractive for most Consumers (primarily medium sized enterprises and research groups). However, despite the attention paid by Providers, some Cloud nodes may attain performance orders of magnitude worse than other nodes

If a Consumer adopts a public Cloud to host a mission-critical service or for a scientific application, performance variability and availability become a concern. Therefore, from a Consumer's perspective, monitoring the perceived performance is necessary to adapt to the changes or to apply corrective measures. For instance, a Consumer may decide to host applications at multiple Clouds to ensure high-availability, switching between Clouds depending on the measured performance. Monitoring is then necessary since it may considerably improve the performance of real applications and affect activity planning and repeatability of experiments. J. Schad et al, (2010)

2.3. Monitoring and measuring performance of cloud applications

Monitoring of Cloud is a task of paramount importance for both Providers and consumers. On the one side, it is a key tool for controlling and managing hardware and soft-ware infrastructures; on the other side, it provides information and Key Performance Indicators (KPIs) for both platforms and applications G. Aceto et al, (2013). Aceto notes that cloud monitoring is needed to continuously measure and assess infrastructure or application behaviors in terms of performance, reliability, power usage, ability to meet SLAs, security, etc, to perform business analytics, for improving the operation of systems and applications and for several other activities.

Michael Kopp, (2012) with other senior IT professionals conducted a survey of 468 CIOs to determine the top most IT investment cloud computing priority for 2013. These professionals were driven by the promised benefits of greater agility, flexibility and time-to-value derived from cloud computing.

The research found that security and cost dominated the list of cloud concern. However application performance has increasingly been making headway as the end user get more demanding. As response time increase user interaction with the system begins to slow and dissatisfaction rises. The research noted that inherent cloud attributes like on-demand resource provisioning and scalability are designed to increase confidence in the usability of applications and data hosted in the cloud. But the most common mistake that people often make as noted by the research is interpreting availability guarantees as performance guarantees in a cloud computing environment. Service-level agreements (SLAs) based on availability say nothing about the end-user experience, which can be significantly impacted by the cloud.

The research further found out that, the fact is that most traditional monitoring tools simply don't work in the cloud. Effectively monitoring and managing modern cloud-based applications and services requires a new approach based on more granular end-user metrics such as response time and page rendering time. This approach must be based in an understanding of the true end-user interaction "on the other side" of the cloud. It must enable cloud customers to directly measure the performance of their cloud service providers and validate SLAs. With this type of approach, cloud customers can be better assured that application performance issues will not undo the benefits of moving to the cloud.

2.4. Service level Agreement on SaaS Performance.

A Service level Agreement as used in public cloud in a binding contract between cloud providers and cloud users. The SLA outlines terms of usage of the cloud services on the users part and terms of providing the services on the providers' part.

According to the practical guide to cloud service level agreements of 2012, SLA considerations for public model are greater than the other models since provider's IT resources are shared across multiple consumers. As a result, consumers should carefully review the cloud SLA to understand how the provider addresses the added security, availability, reliability and performance risks introduced by multi-tenancy. The guide further discussed that the ability to measure and track specific service level objectives becomes more important in the Public deployment model. Consumers should also ensure the cloud SLA provides adequate methods and processes for ongoing measurement. At the SaaS level, the guidelines states that consumers should expect general SaaS service level objectives like monthly

cumulative application downtime, application response time, persistence of consumer information, and automatic scalability to be included in their SLA.

In relation to performance, cloud SLAs should cover both unavailability (hard downtime), as well as performance degradation. Many providers offer clear SLA language explaining what happens if their infrastructure goes completely offline, but fail to mention whether performance degradation is also considered an SLA violation. (Dimension Data, 2013)

Crystall Bedell (2012), states that SaaS SLAs are written with the provider's best interest in mind and not the customers' needs. Customers should therefore behoove businesses to know what they can expect to see in a SaaS SLA before evaluating providers. The author further discusses that the key items to expect in a SaaS SLA are Availability, Downtime, Mean time to repair and mean time to respond, Escalation path and A word of warning on SLA's

2.5. Review of current cloud monitoring tools

This study will review some of the SaaS application monitoring services and compare them in terms of features, usability and price. In the article Application Monitoring Tools: Comparison of SaaS Solutions Dan Sullivan (2014) discusses the top four Saas monitoring tools as

AppDynamics

Monitoring application is available as an on-premises solution as well as a cloud-based service.
The author notes that end user experience monitoring functions of the service provide dashboard views of key metrics, such as highest end user response time, highest request per minute, number of requests over time, end user response time and number of JavaScript errors over time. While it is useful for developers and administrators to have low level metrics, such as end user response time and page rendering time, it is often more important to consider application performance from an end user perspective.

AppDynamics provides for reporting by higher level business transactions. This is especially helpful for tracking performance and impact by application function, such as checking out or searching for a product. Transactions can be configured so administrators can monitor overall health of the

transaction service, number of transactions executed, and number of errors, along with slow and stalled transactions.

In addition to tools for isolating performance problems, reports are also available for tracking baseline performance over longer periods. AppDynamics includes a release comparison tool for measuring and comparing performance of two different versions of an application.

New Relic

New Relic application monitoring service is designed for ease of use and minimal steps to deploy. Like other cloud-based application monitoring tools, New Relic offers consolidated views of summary data about application performance, from both the end user perspective and from lower level network and application perspectives.
New Relic offers Real User Monitoring (RUM) that collects performance statistics from actual users working with the monitored applications. The alternative approach is to use applications that simulate user interactions. The real user monitoring approach has the advantage of providing data on actual customer behavior, which may be difficult to adequately model in simulated loads. However, the simulated loads do have the advantage of allowing for comprehensive testing of all transactions across a variety of geographic regions and browser types.

New Relic provides agents for monitoring both iOS and Android mobile devices. The data collected by the agent can help identify problematic code by platform as well as understand the impact of wireless network performance on overall application performance. The agent collects data on application activities (e.g. screen loading), Web service and API calls, as well as device level user interface and memory performance statistics.

New Relic supports commonly used programming languages and frameworks including Java, .NET, Ruby, PHP, Python and Node.js. Like AppDynamics, New Relic includes a release comparison tool. Developers can gain insight into database queries with the SQL monitoring tools that provide details on top SQL queries based on runtime, database throughput, database response time, as well as slow running queries. The Thread Profiler in New Relic can collect stack traces over a defined period of time and then summarize the results to help developers identify inefficient code. The cross application tracing feature is helpful for analyzing distributed applications.

Scout

Devops professionals accustomed to working with Chef, Puppet or other automation tools will appreciate Scout's support for deploying with existing configuration scripts. Scout also provides a Web interface for deploying monitoring plugins so administrators do not have to SSH into servers directly to install components.

Scout currently supports over 70 plugins, including general system monitoring components for monitoring logs, processor statistics, network metrics and Amazon Web Services EC2 monitoring. Many of the plugins are designed for specific application stack components, such as those for monitoring MySQL, PostgreSQL, MongoDB, Redis and Memcached. For those with specialized monitoring requirements, Scout supports custom plugins written in Ruby.

Scout also provides an API to facilitate integrating Scout reporting and monitoring into other monitoring and management dashboards.

OpTier

The author says that OpTier emphasizes the importance of transaction-based analytics in its cloud-based offering. It is designed to help developers and application managers monitor and optimize Java and .NET applications, OpTier uses proprietary techniques to tag transactions across the application infrastructure.

Reporting and monitoring tools allow administrators to compare performance over time as well as generate alerts when performance drops below a specified threshold or a particular event occurs. The large volumes of data collected by application performance monitoring can be analyzed using OpTier's Big Data Analytics service.

The monitoring dashboard provides a single point of access to a range of data, including performance and geographical data along with the ability to search for specific transactions and drill down into instance details. In addition to providing access to data about past performance, OpTier uses predictive algorithms to analyze current trends and alert administrators to potential forthcoming performance issues.

2.6. Agent Based Systems

An **Agent** is a computer system situated in some environment and that is capable of autonomous action in this environment in order to meet its design objectives (M. Wooldridge and N.R. Jennings, 1995).

Yu M. Z. & Nay M.T. (2014) argues that an agent is a computational entity that acts on behalf of another entity (or entities) to perform a task or achieve a given goal. Agent systems are self-contained software programs embodying domain knowledge and having ability to behave with a specific degree of independence to carry out actions needed to achieve specified goals. They are designed to operate in a dynamically changing environment. Agents typically include a set of features. The main features of agents include the following as discussed by the authors:

- **Autonomy:** the capacity to act autonomously to some degree on behalf of users or other programs also by modifying the way in which they achieve their objectives.
- **Pro-activity:** the capacity to pursue their own individual set goals, including by making decisions as result of internal decisions.
- **Re-activity:** the capacity to react to external events and stimuli and consequently adapt their behavior and make decisions to carry out their tasks.
- **Communication and Cooperation:** the capacity to interact and communicate with other agents (in multiple agent systems), to exchange information, receive instructions and give responses and cooperate to fulfill their own goals.
- **Negotiation:** the capability to carry out organized conversations to achieve a degree of cooperation with other agents.
- **Learning:** the ability to improve performance and decision making over time when interacting with the external environment.

According to (M. Wooldridge & N.R. Jennings, 1995) agent-based system, refer to systems in which the key abstraction used is that of an agent. In principle, an agent-based system might be conceptualized in terms of agents, but implemented without any software structures corresponding to agents at all. An agent-based system is expected to be both designed and implemented in terms of agents. A number of software tools exist that allow a user to implement software systems as agents, and as societies of cooperating agents. An agent-based system may contain any non-zero number of agents. The multi-agent case – where a system is designed and implemented as several interacting agents, is both more general and significantly more complex than the single-agent case. However, there are a number of situations where the single-agent case is appropriate.

2.7. The Proposed System Architecture

The Figure1 below shows a diagram of the proposed system architecture.

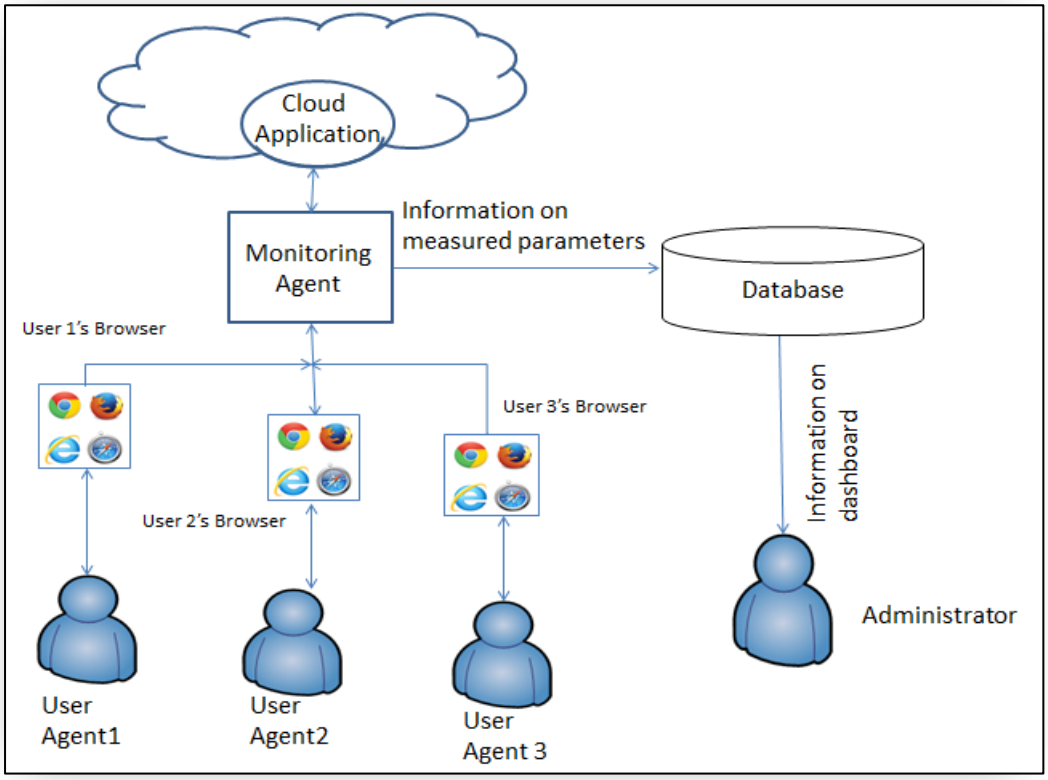

Figure 1. Cloud Application Performance Monitoring Architecture

Source: Author

The proposed architecture comprises of several elements.

- User's Browser: the software/web platform from which the users access the cloud application
- User Agent: The agent that was developed to interact with the cloud application on behalf of cloud users.
- Monitoring Agent: The agent was developed to collect monitoring information or parameters when the user accesses' the cloud service.
- Cloud application: this is the service being accessed by the user. It's provided by a service provider.
- Information on measured parameters: this is the information collected by monitoring agent.

- Database: this represented the storage location to which the information collected by the agent is consolidated. The stored information can be accessed in the future
- Information on dashboard: the consolidated information is represented on a dashboard. The dashboard information gives a summary of the measured parameters.
- Administrator: system administrator who reads and interprets information on the dashboard.

According to Shailesh Paliwal (2014) SaaS performance measures are directly perceived by users as business transaction response times and throughput, technical service reliability and availability, and by scalability of the applications. This study will be guided by these performance parameters. The proposed tool will be developed to monitor performance by measuring the performance of the applications on the public cloud based on these users perceived performance measures.

To measure response time and throughput the study focused on (i) time taken to respond to a user's request (ii) a measure of time delay experienced in an applications (Latency) and (iii) time taken to complete a transaction. To measure technical service reliability the study measured the number of requests dropped and number of times the system crashes or cannot be accessed. Availability was measured as the duration during which the user could not access the applications in a scenario of constant access of the application. The proposed system has also been developed to determine application downtime as the time duration during which the application is not operating especially due to a malfunction from the providers' side.

The agent based system used two types of agents; user agent and monitoring agent a single agent to measure application performance. The user agent were developed to interact with the cloud application on behalf of users while the monitoring agent was developed to measure performance parameters from the user agent interactions with the cloud application. Several instance of the user agent were used to represent more than one user interaction.

The agents developed showed some of the characteristics of agent features of agents as discussed by Yu M. Z. & Nay M.T. (2014). The developed user agent have the quality of A**utonomy,** this agent autonomously acts on behalf of the user to interact with the cloud application. To collect performance measure information from the cloud application and also send the information for consolidation in the server **Communication and Cooperation** attribute. The system is proposed to operate in a dynamically changing environment which is the major characteristic of agent based system.

CHAPTER THREE: METHODOLOGY

3.0. Introduction

This chapter contained the steps that were undertaken in the execution of the study to fulfill the study objectives. These are the research Design, Data collection methods, Data analysis.

3.1. Methodology

In this project the Prometheus Methodology of developing multi-agent systems was used. The choice of this methodology was guided by the need to have well defined deliverables at each stage of the development.

The Prometheus methodology is a practical methodology and provides everything that is needed to specify and design agent systems. Other distinguishing features of the Prometheus methodology as discussed by (Lin Padgham and Michael Winikoff, 2014) include,

- Prometheus is detailed as it provides detailed guidance on how to perform the various steps that form the process of Prometheus.
- Prometheus supports (though is not limited to) the design of agents that are based on goals and plans. The author notes that the benefits that can be gained from agent-oriented software engineering comes from the use of goals and plans to realize agents that are flexible and robust.
- Prometheus covers a range of activities from requirements specification through to detailed design.
- The methodology is designed to facilitate tool support, and tool support exists in the form of the Prometheus Design Tool (PDT) which is freely available.

The project used the three phases of Prometheus Methodology of system development as discussed by (Lin Padgham and Michael Winikoff, 2014).

- **System Specification**: the system was specified using goals and scenarios; the system's interface to its environment was described in terms of actions, percepts and external data; system functionalities were then defined.

- **Architectural design**: this involved identifying the different types of agents in the system; the system's overall structure was captured in a system overview diagram; and scenarios developed into interaction protocols.
- **Detailed design**: the details of each agent's internals were developed and defined in terms of capabilities, data, events and plans; process diagrams were used as a stepping stone between interaction protocols and plans

3.2. Research Design

In this project we started by reviewing relevant literature as would relate to the project. The literature review was used to understand how agent based systems have been used to measure and monitor cloud application performance.

The requirements for the agent based system were derived from observation of users interacting with the live system. Users interacting with cloud application were observed and duration of executing transactions on the cloud application measured and recorded. The choice of observation was based on the need of gaining in-depth understanding of live users' interaction with the cloud application.

The agent based system was then developed using the Prometheus Methodology of developing agent based system. This methodology was chosen since it gives well defined deliverables at every phase. The Prometheus methodology was used for system specification and design of the agent based system.

The developed system was then evaluated by selected industry experts for its capabilities. Questionnaires were used for the evaluation process. The choice of using questionnaires was derived at following the need to collect a lot of descriptive information describing the developed system.

The data generated by the system was analyzed to get the average response time. This average was compared to the data collection. The evaluation of the system and the comparison of the data formed the basis of concluding on the system.

3.2.1 Population Sampling

In this project sampling was done to select a cloud application to be used in this research. Population sampling was also used in selecting industry experts used to evaluate the agent based system developed.

Selection of cloud application

Selection of the cloud application used in the research was guided by judgment sampling based on the researcher's knowledge. The selection was based on the knowledge of the most commonly used public cloud application.

Some of the most commonly used Public cloud applications known to the researcher were selected as Facebook, Gmail and Skype. Gmail was selected to be used in the research since it's more likely to be used for both enterprise solution and personal communication solution hence representing a larger number of users of the public cloud application.

Population selection of industry experts

In this research experts from different organizations whose head office are within Nairobi region were selected. Purposive sampling was used in selecting the industry experts. This research targeted a population of 20 participants as an ideal sample size for the research. Out of the 20 targeted, 13 experts in the industry were available and fully participated in the evaluation process. The participants of the evaluation process were different stakeholders of different levels in the cloud computing industry ranging cloud service provider, cloud application administrators and users.

3.2.2. Data Collection

In this research, the data collection method used was observation. The researcher observed users interacting with a cloud application and measured the transaction duration as carried out by the users. The transaction durations were compared to the durations generated by the system to ascertain the functionalities of the system. Measurements of the transactions were done in different scenarios/settings.

Observation Settings

During data collection, this study focused on specific activities as carried out by the users during transaction execution.

1. Scenario 1: One user running a complete transaction on a cloud application. This study focused on this scenario in order to measure transaction duration that would represent the response time of the application.

2. Scenario 2: Three users running complete transactions on a cloud application simultaneously. In this scenario the researcher focused on measuring transactions durations as the application was run by more than one person hence determine the scalability of the application.

3. Scenario 3: One user running a cloud application for a long period. The researcher observed a user running transactions continuously at random intervals within one hour to ascertain availability of the application.

Transactions of the cloud applications

The transactions for the cloud application were defined as below;

- ***Application launch***, this defines opening the application for use. Web browser was used to open/launch the application.
- ***Application Login***, this transaction was defined as the activities from the time a user submits their login credentials to the time the user is completely authenticated to use the application.
- ***Reading a message***, defined as the activities from the time a user selects a message to the time the message is opened for the user to read.
- ***Sending a message***; this transaction was defined as the activity from the time a user sends a message to the time they receive a message acknowledgement or the message appears in their sent items folder.
- ***Application Logout:*** The activity from the time a user selects the logout command to the time the user is completely exited from the application.

Transaction durations were measured as follows.

- ***Application Launch duration***: Duration from the time a user enters the application web address to the time the application is opened on the users web browser
- ***Application Login duration***: Duration from the time the user submits their login credentials to the time he/she is authenticated and the application is fully available for use.
- ***Sending a message;*** Duration from the time a user clicks on the send button to the time the user receives a successful acknowledgement.
- ***Reading a message:*** Time duration from the time the user clicks on a received email to the time the email fully opens to be read.

- ***Application Logout;*** Time duration from the time the user clicks on the logoff command to the time the application completely exists.

Recorded Durations

Table 1: Data Collection results

Scenarios	**Transactions**	**Transaction Duration in Milliseconds (ms)**
Scenario 1	Application launch	2594
	Application Login	9175
	Reading a message	2970
	Sending a message	3517
	Application Logout	8765
Scenario 2	**USER 1**	
	Application launch	7953
	Application Login	1738
	Reading a message	2466
	Sending a message	5369
	Application Logout	2714
	USER 2	
	Application launch	3369
	Application Login	10380
	Reading a message	4787
	Sending a message	4471
	Application Logout	17356
	USER 3	
	Application launch	3722
	Application Login	13984
	Reading a message	1836
	Sending a message	1964
	Application Logout	7244
Scenario 3		
Instance 1	Application launch	3425
Instance 2	Application Login	10949

Instance 3	Reading a message	2496
Instance 4	Sending a message	3402
Instance 5	Application Logout	4144

3.2.3. Data Analysis

Data analysis was done using excel to determine the average transaction durations.

Table 2: Data Analysis

		SIMULATENOUS USERS		
	SINGLE USER	**USER 1**	**USER 2**	**USER 3**
Transaction Durations in Milliseconds (ms)	4592	5493	6184	4982

3.3. Tools Design

In this project JADE integrated with Netbeans was used to develop the agents in the system. The system runs on a Java platform. The database was developed using SQLite.

3.4. Tools and Skills Required

In realization of the goals of this project the following tools were used.

a) JADE
b) Netbeans
c) SQLite Database
d) JRE platform
e) Agent Oriented programming skills
f) Access to a Public Cloud application
g) Internet services

3.5. Systems Design

The system was designed using the Prometheus methodology guidelines. In designing the system we focused on showing the different agents to be developed, how they work and their interactions. The design also focused on the capabilities of each agent in the system.

Prometheus methodology was used as it gives well defined deliverables at every phase.

3.6. System Evaluation and Recommendations

Questionnaires were used for system evaluation. The questionnaire was designed to gather data on the developed system's functionalities and usability from the industry experts.

The system was also evaluated by analyzing the data generated from the agent based system to the data collected during real world data collection to establish the correct functionalities of the system. Based on the interpretation of the results further study is recommended on the system.

CHAPTER FOUR: ANALYSIS AND DESIGN

4.0 Introduction

This phase involved analysis and design of the proposed system. The system has been developed to demonstrate use of agents in interacting with a cloud application on behalf of a user and the use of agents to measure the performance of the cloud application during these interactions. The system has been developed using the Prometheus methodology of developing multi-agent system.

4.1 System Analysis

Analysis started by reviewing literature on some of the existing system. Their functionalities and short comings were listed. In order the gather the requirements of the system to be developed we studied a manual system using live user. This is because we were not able to use any other existing automated systems due to confidentiality and privacy policies of users and system service providers.

Observed manual system

The analysis phase was done to understand the requirements of the system to be developed. It started by observing live users interacting with the cloud application. The users were setup in different scenarios and observed how they interacted with the cloud application. User interactions with the cloud application included carrying out transactions on the selected application. The transactions observed included

- Launching the application from a web browser
- Logging into the application using an email address and password
- Reading emails, the users could read any received email in their inbox
- Sending emails; to send an email the users required a recipients email address and the message body to send
- Logging off the application

In order to measure transaction duration in this setting, an online clock system was used. The online clock used a start stop mechanism. Measurement of the transaction durations was done separately by a different user from the one carrying out the transaction.

Weaknesses of the observed system

In the observed system all transactions needed to be carried out by live users. This meant the live users could only be engaged in carrying out this transaction at that particular time. During user

interaction, measurement of the transaction durations was done by a different person all together, this meant that the accuracy of this durations depended on the keenness of the live user.

Proposed System

The proposed system was developed to demonstrate the use of agents in interacting with the cloud application on behalf of live users. Proposed system eliminates the needs to have live users interacting with the system and hence can be used in situations where the required number of live users is not available.

The system also demonstrates the use of agents in measuring transaction durations during user interaction with the cloud application. This eliminates the need of using a timer to measure the durations hence giving more accurate measurements. The elimination of a human user in taking measurements also eliminates errors during transaction measurements.

4.1.1 User requirements

They define what the user understands as the expected functionalities of the system. The user requirements were generated from the manual system observed. This included

The user requirements for this system are:

- At least one user should be able to interaction and perform transactions on a cloud application
- Several users may also interact with the cloud application.
- The performance measures should be measured by the system during user interaction with the cloud application.
- The measured parameters should be displayed on a graphical user interface
- Provide historic as well as real time data of the applications performances
- Poll the database at regular intervals for relevant information
- Automatically represent information as measured from all concurrent users interacting with the system.

4.1.2 System Requirements

System requirements describe the requirements excepted from the system. The system requirements as derived from the user requirements in this system include;

- The system should allow at least one agent to interact with the cloud application.

- Several agents in the system should be able to interact with the system simultaneously.
- The system should be able to pick the starting time and stopping time of a transaction during the agent user interactions
- The system should use the start time and end time of a transaction to process/calculate the transaction duration
- The calculated durations should be stored in a database
- Users should be able to query the database for real time and historical data.
- All user agents should be authenticated by use of email address and password. The email address and passwords should be stored in the database.
- Users should interact with the system on a graphical user interface.

4.2. Analysis using Prometheus Concepts.

Prometheus concept was used in the analysis phase of this system. The Prometheus methodology is mainly used for designing multi-agent based system. The concept of Prometheus methodology of system analysis and design consists of system specification phase, the architectural phase and detailed design phase.

4.2.1 System specification

This phase involved specifying the system functionalities using goals and scenarios. It also involved identifying the system inputs (percepts), outputs (actions) and external data that describes the system's interface to its environment. The phase started with a brief description of the system then proceeded to define the requirements of the system in terms of;

- The goals of the system
- Use case scenarios
- Functionalities, and
- The interface of the system to its environment, defined in terms of actions and percepts.

4.2.1.1 System Description

The proposed system provides a platform for corporate users (system administrators) to measure and monitor performance of cloud applications during users interactions with the system. The system demonstrates the use of agents in user interactions with the cloud application on behalf of the live users. During agent interaction with the cloud application, an intelligent agent has also been used to capture transaction durations that can be translated to performance measures of the applications. Users of the system view current and historic data of the measured parameters.

4.2.1.2. Overall System Goal

The system goals describe the main purpose of the system; what the system is expected to achieve. The overall goal of the system is to use agents to interact with the cloud application on behave of live users, measure performance parameters of the application and provide useful data for monitoring performance of the cloud application.

The system allows system administrator to activate a number of users to interact with the cloud application. Once a user is activated the system activates a user agent that acts on behalf of the user. During user agent interact the system uses a monitoring agent to capture performance measures from these interactions. The main goal of the system is to use agents to interact with the cloud application on behalf of the live users and also capture performance of the application during user interaction with the system.

4.2.1.3. System Sub-goals

The sub-goals for the system as derived from the main goal of the system were defined as below

- Allow one user interaction with the cloud application
- Allow more than one user interaction with the cloud application simultaneously
- Provide meaningful data for monitoring of the cloud application performance.

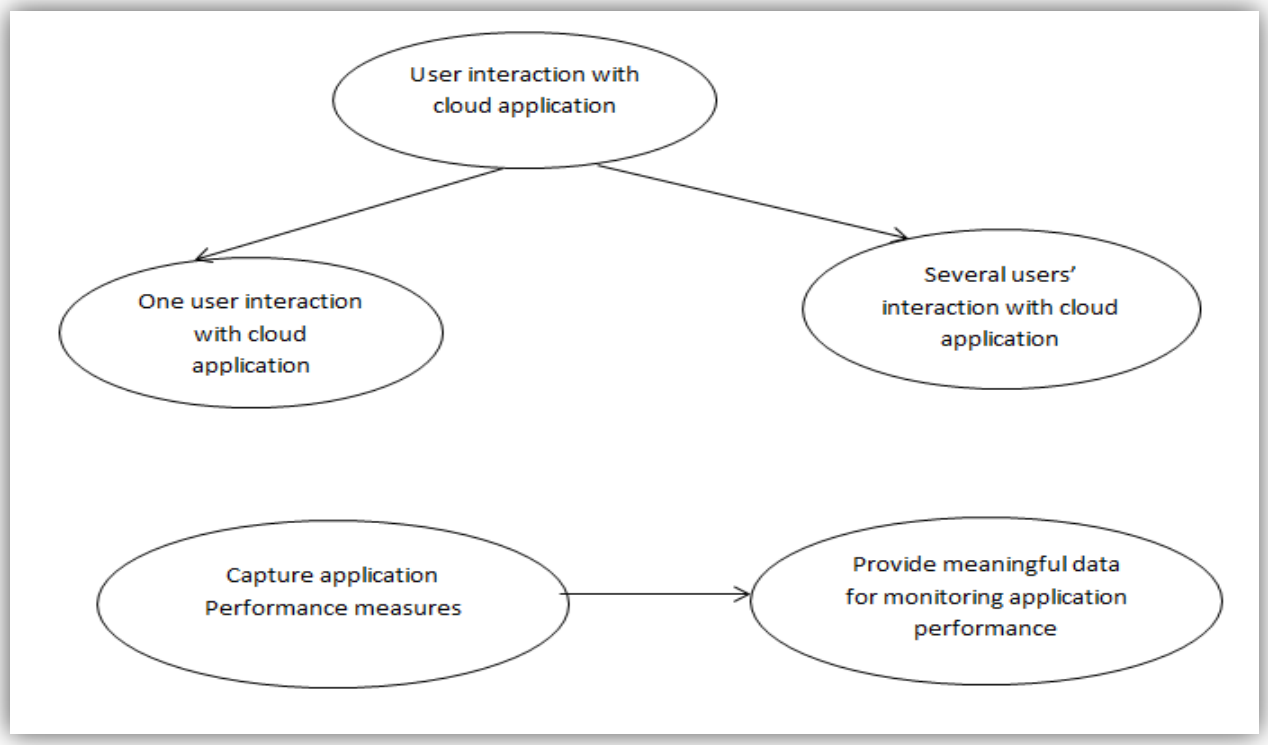

Figure 2: System Goals

4.2.1.4. Use Case Scenarios

Use case scenarios were used to represent specific system functionalities. They were used to describe sequence of events associated with achieving a particular goal of the system or responding to a particular event.

Use Case 1: User interaction with the cloud application

User interacts with the cloud application. The interaction of the user on the cloud application involves carrying out a complete transaction on the application.

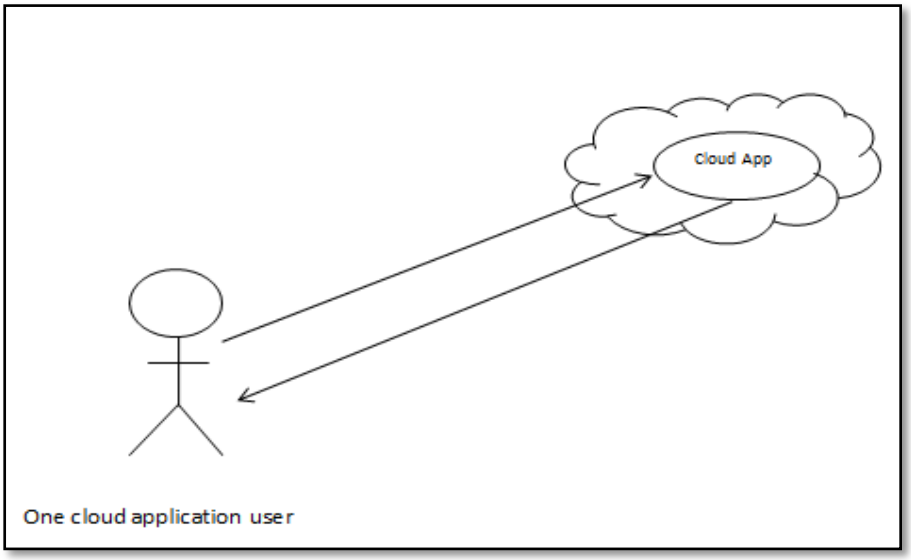

Figure 3: Use Case Scenario1

Use Case 2: More than one user interacting with the cloud application simultaneously. All users run complete transactions on the cloud application all at the same time.

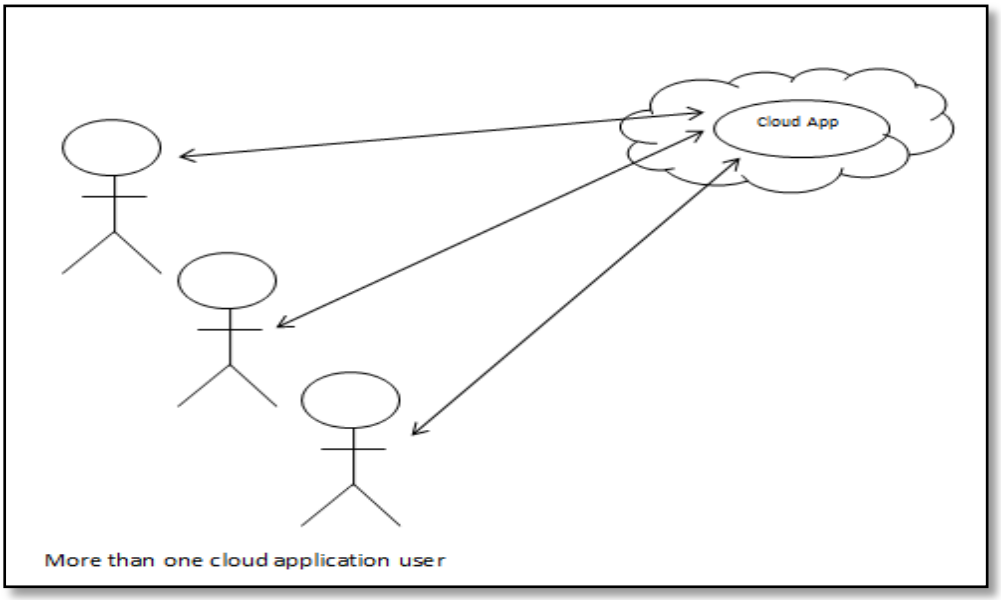

Figure 4: Use Case Scenario 2

Use Case 3: Monitoring agent collects performance measures from the cloud application. When the user interacts with the cloud application, the monitoring agent collects performance measurements data from the interaction.

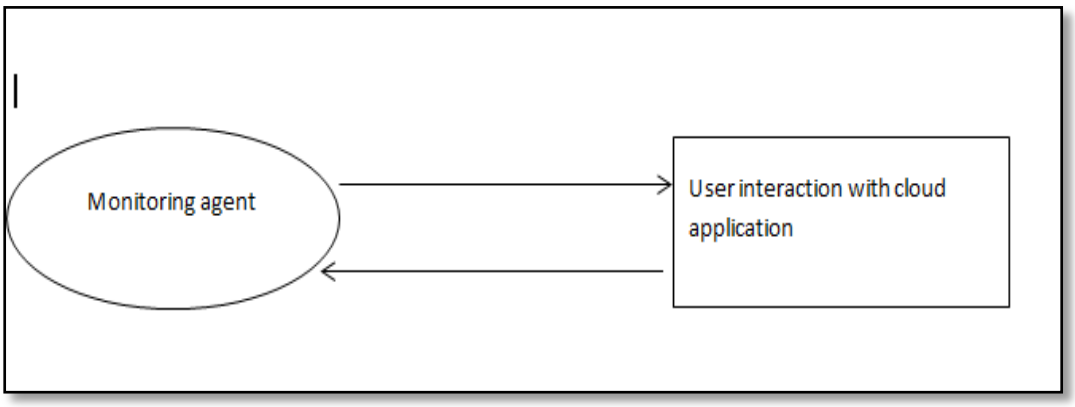

Figure 5: Use Case Scenario 3

Use Case 4: System Administrator views performance measures displayed on the dashboard.

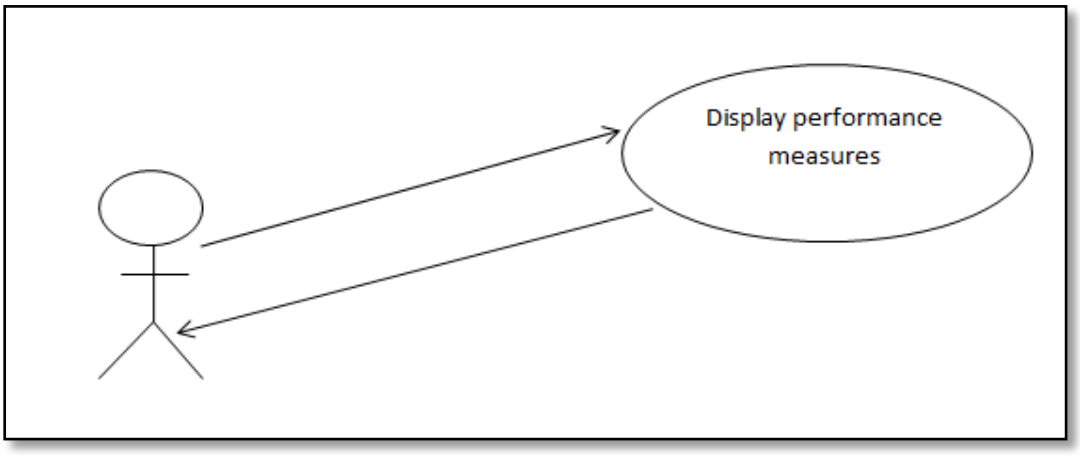

Figure 6: Use Case Scenario 4

Use Case 5: Database administrator adds or removes users to the database

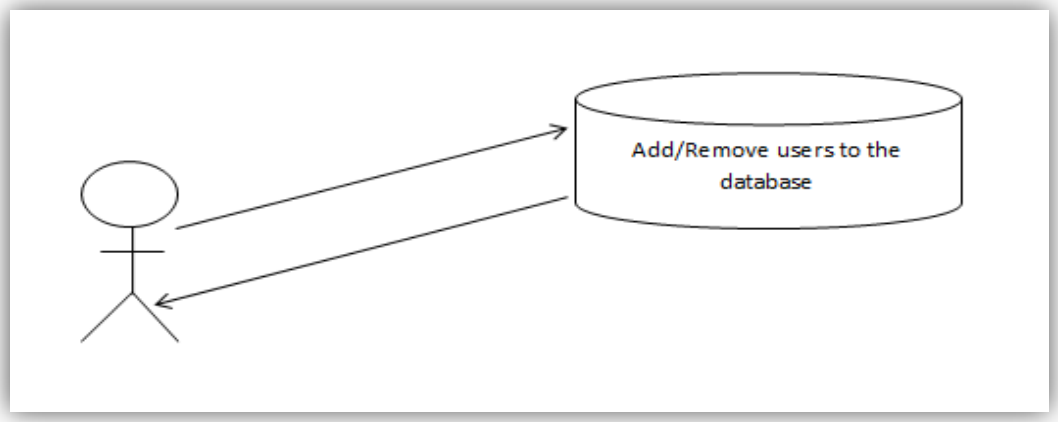

Figure 7: Use Case Scenario 5

4.2.1.5. System Actors

The actors of this system were identified as

- Database Administrators: Their key role was to add user details into the database. The user details were used by the user agents for authentication during cloud application interaction. The database administrator can also perform other database activities on the system such as modify, view and delete user details.
- System administrators: the system administrators interact with the system by providing transaction information during the interactions and view measured transaction durations as displayed by the system.

4.2.1.6. System Functionalities

The system functionalities are derived from the system goals. The functionalities define what the system is capable of doing. In this project the system functionalities identified were

- User details management on the database i.e. addition, modification or removal of user details
- One or more user interaction with the cloud application
- Measurement of performance during user interaction with the cloud application.
- The measured parameters should be displayed on a graphical user interface
- Provide historic as well as real time data of the applications performances

4.2.1.7. System Interface

The system interfaces to its environment were defined in terms of percepts (inputs) and actions to be performed. In this system the percepts will be provided by the actors of the system that will facilitate the system to perform the expected actions.

The **percepts** (inputs) associated with each actor were then identified as below

i. System administrator

The inputs by the system administrator are the information required during transaction executions on the cloud application. These inputs include

- Number of users interacting with the system
- Type of transaction to be executed

- Message details i.e. recipient email address, message subject, message to be sent and position of message to be read.

ii. Database Administrator

The database administrator updates user details in the database. These details were;

- User Email address
- User password

Actions

The main actions of the system include

- Use provided data/information to carry out defined transactions
- Read transaction start time and transaction completion time to measure transaction durations
- Display the transaction durations on the GUI (Graphical User Interface) for users

4.3. Architectural Design

In this design phase the different agent types in the system were identified and grouped. The agent interaction diagrams were used to describe the interactions between agents in the system. The overall system structured was also defined in the phase.

4.3.1. Agents

Agents used in the model included.

- User Agent
- Monitoring Agent
- System Runner Agent

User Agent

The user agents were used to interact with the cloud applications on behalf of the cloud application users.

Functions of the User Agent

- The agent interacts with the cloud application to performance application transactions.

Monitoring Agent

The monitoring agent was responsible of collecting all monitoring data that arises as a result of user agent interaction with the cloud application. The monitoring agent passed the collected data to the database for storage.

System Runner Agent

The system runner agent is developed to keep the system running e.g. sustains database connectivity. It ensures that the database connection is always open for data storage and retrieval.

4.3.2. Agents grouping

Grouping of agents was based on the roles of the agents and the usage of data by each agent. Agent grouping diagram below describes the roles of the agents in the system.

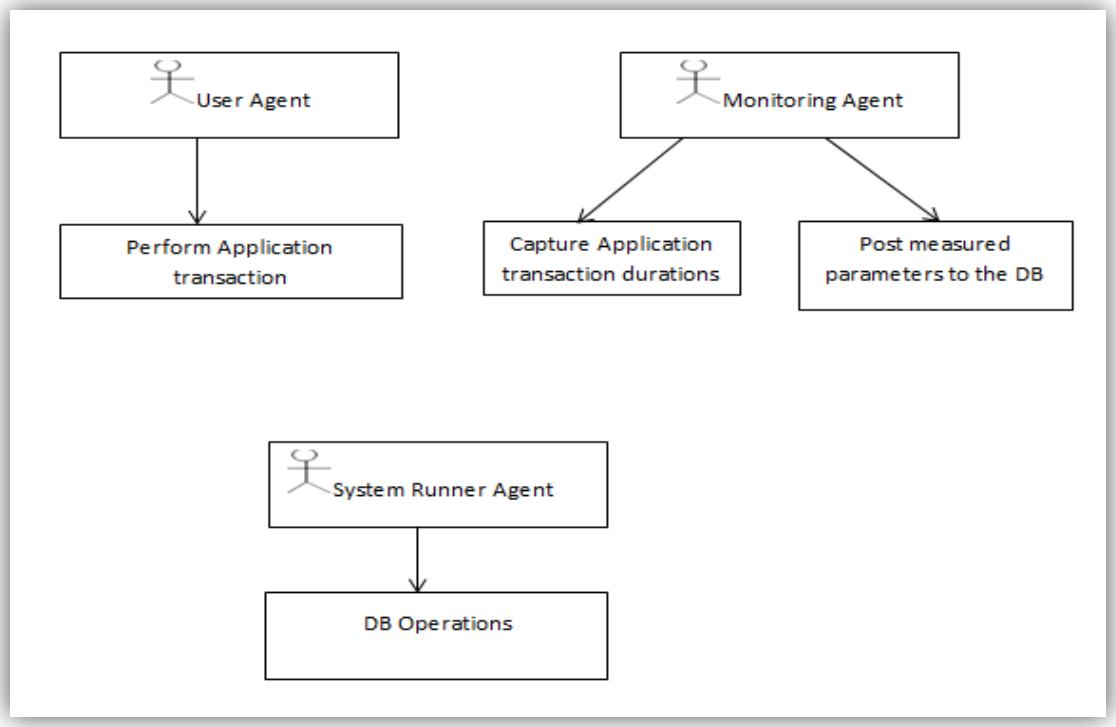

Figure 8: Agent Grouping diagram

4.3.3. Agent Interaction

The agent interaction diagram shows how the agents interact in the system. The agent interaction is derived from the use case scenarios. In the Agent interaction diagram the main element is the agent.

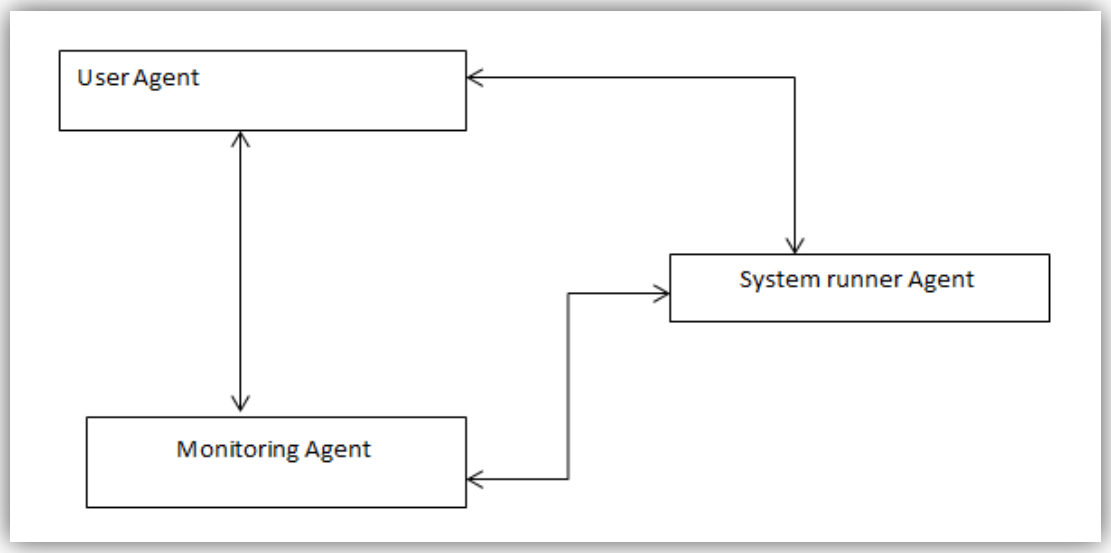

Figure 9: Agent interaction Diagram

4.3.4. Agent Protocol Interaction

As the agents' interaction, they use protocols to communicate; the protocol interaction diagram was used to show the protocol interchange between agents. During agent protocol communications, monitoring agents and system runner agents communicate in a two way communication while the user agent communicates with the monitoring agent. Communication only commences once a user agent has been activated. The user agent will initial communication to the monitoring agent. The communication takes place during each transaction carried out by the user agent. Once the user agent exists the monitoring agent then communicates to the system runner agent to so that that communication channel can be closed.

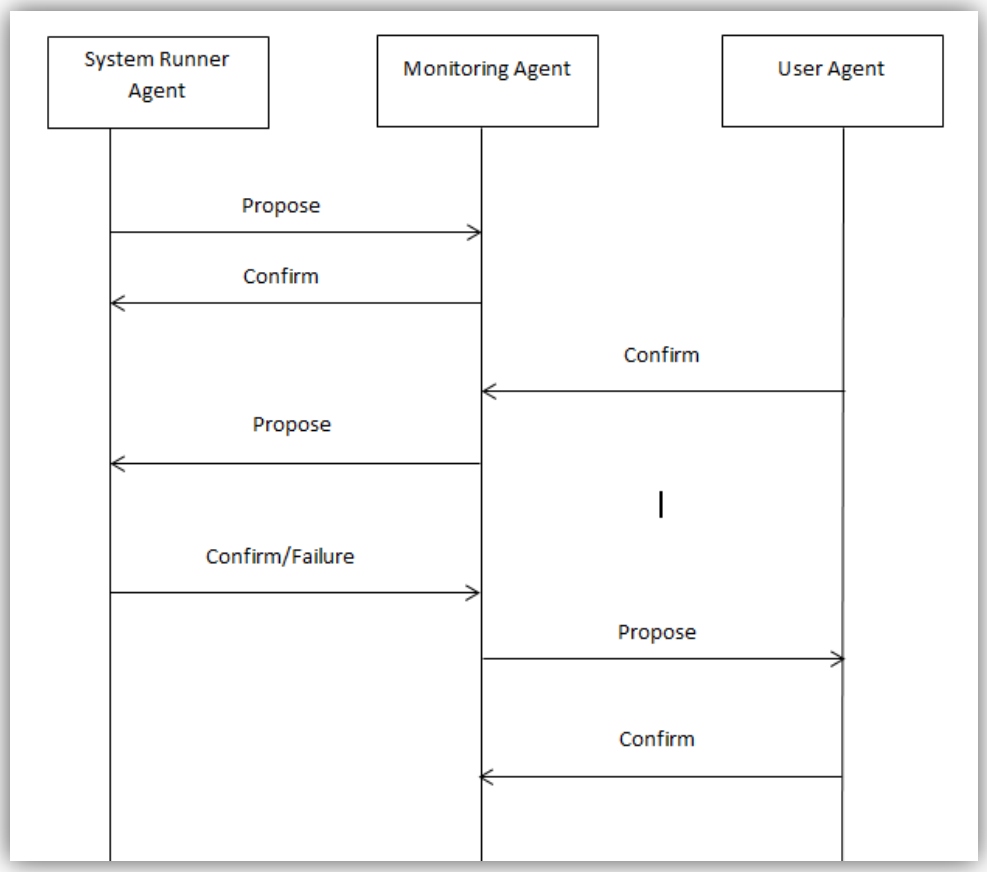

Figure 10: Agent Protocol Interaction Diagram

4.3.5. System Overall

The system overall diagram was used to design the system structure. The overall structure included the different agent types in the system, the boundaries of the system and its interfaces. The interface was defined in terms of actions and percepts.

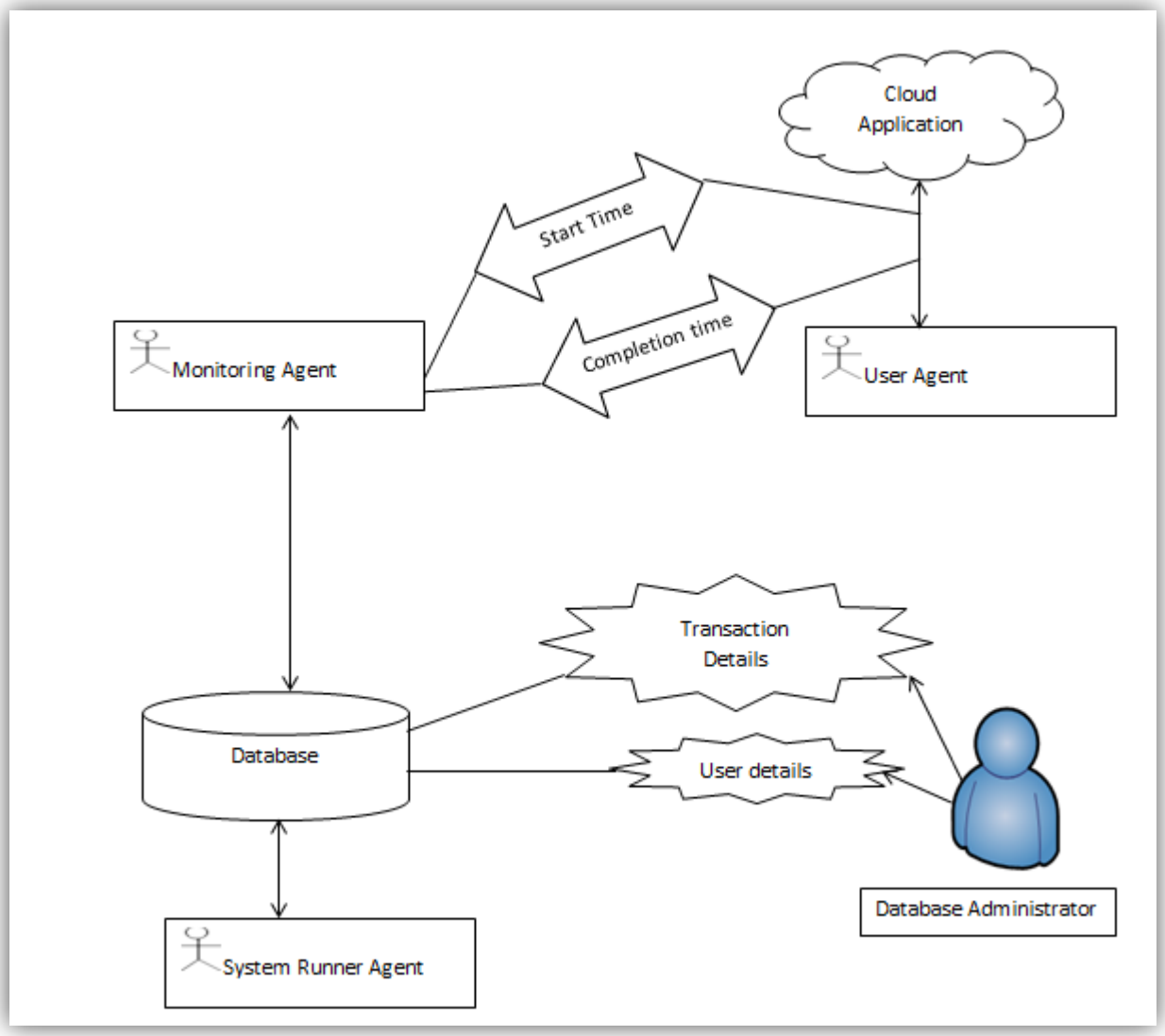

Figure 11: System Overview Diagram

4.4 Detailed Design

In this phase we looked at the internal capabilities of each agent and determined how each agent will accomplish its tasks within the overall system.

Table 3: Agents Capabilities

AGENT	CAPABILITIES
User Agent	This agent run all application transactions, the transactions include application launch, application login, sending message, receiving message and log off the application.
Monitoring Agent	This agent captures performance of the cloud application

	during user interaction. The agent measures the transaction durations as performed by user agents.
System Runner Agent	This agent handles database operations by ensuring the database connection is open for writing and reading data.

4.5. Implementation

This phase involved converting the system designs to a computer readable program using programming languages. The agent based system was primarily developed to run on a Java platform. The implementation entailed development of agents and their respective functions, agent interactions and database setup for data storage.

4.5.1 Implementation Tools

Several implementation tools were used to develop the various components of the system. The developing tools included;

SQLite3

The database we developed using SQLite3**.** SQLite is a disk-based database. SQLite makes it possible to prototype an application then move the code to a larger database e.g. Oracle. The database was configured to store user details. These are the details of the users who were used to interact with the system. The database was also configured to store transaction details which included transaction type and durations.

JADE

Java Agent Development Framework was used for developing the agents in the system. JADE is widely used for developing multi-agent based systems. JADE was used to develop the user agents that interacted with the cloud application on behalf of the users and monitoring agents that captured transaction durations during the user interactions.

JRE version 7Application environment:

The system has been developed to run on application environment of JRE (Java Runtime Environment) Version 7 and above. The JRE platform is the environment that is used to run Java based system.

Netbeans IDE 8.0.2

The system being a java application was developed on Netbeans Intergrated Development Environment. This is an application platform framework for Java applications and can run on windows operating system but is also compatible to other operating systems.

Java JDK Version 8 Update 65:

This is the Java development kit platform that was used in implementation of this system. Java JDK is the most commonly used software development Kit for java applications. This tool was mainly used for developing the java application.

4.6. System Testing

Testing was done during and after implementation to evaluate compliance of the system against the specified requirements. The functionalities of the system were tested from an end-to-end perspective. In this project static and dynamic testing was done. Static testing involved checking on the code in a non-runtime environment (not running the code), checking on the requirements documents and design document to find errors. Dynamic testing involved executing of the code to check on functional behavior of the system. The output of the executed code was compared to the expected outcome (based on the system specifications) to determine the functionality of the developed system.
In this phase of testing component/unit testing, integration tests and system testing were done.

4.6.1 Component Testing

In this phase each component of the system was tested separately. The components tested were; agents, database and dashboard interfaces. They were tested as below

- The agent codes were examined to ensure they were configured to collect the desired data.
- Agent interaction with the application (on behalf of the users) were tested to ensure that each user transaction had been coded corrected.
- The database structure was examined and connectivity configurations codes checked.
- The interfaces were tested for user friendliness and to evaluate that correct configurations for adequate data representations had been made.

The component testing was done to primarily identify code or syntax and logic error in each module. The testing also aimed at identifying the functional behavior of each module.

4.6.2 Integration Testing

During integration testing the system individual modules were combined and tested as groups. The test was carried out to examine specific user requirement functions and the results were compared to ensure that they met the target requirements.

The tests conducted included

- Agent connectivity to the database was tested to evaluate that the agent can connect and submit collected data to the database.
- The interface between the graphical user interface and the backend database was tested. This test was done to ensure that the output of the data submitted to the database is well represented on the graphical user interface.
- The graphical user interface was tested to ensure that it is user friendly and provides information that can easily be understood and used by the user.

The main aim of this testing was to test the interfaces between the modules. This project used the bottom up integration testing approach. Using this approach the testing started from the lowest unit (module) and progressed up wards. These tests were done during system implementation.

4.6.3 System Testing

This testing was done after the system had been fully developed. The testing was done to verify that the entire system was functional. The system was tested for correctness and completeness. System testing was done to find bugs and errors in the behavior of the system in general. To test the functionalities of the system, the following tests were carried out.

Tests Done On the Developed System

Table 4: System Testing

Item No.	**Component Tested**	**Purpose of the Test**	**Test Data**	**Expected results.**
1	Application Launch by one user	Determine a single instance of the application	One active user agent	Application launch and Duration of the one instance

				application launch
2	Application login	Ability to login to the application	User login credentials	Successful user login and duration of application login
3	Message sending	Determine ability to send message	Recipient email address, message subject and message to be sent	Sent message and duration of sending message
4	Message Reading	Ability to read messages	Message position	Open message to be read and duration of opening the message
5	Application Logout	Ability to logout of the application	Logout command	Application closed and duration of closing the application
6	Application Launch by more than one user	Determine simultaneous launch of several instance of the application	More than one active user agents command	More than one simultaneous application instances launched and duration of each application instance launch
7	Simultaneous Application login	Determine ability to create simultaneous login instances for the application.	Several users login credentials	More than one simultaneous application instances login and duration of each application

				instance login
8	Simultaneous Message sending	Determine ability to send several messages simultaneous by several users.	Recipient email address, message subject and message to be sent	Several messages sent simultaneously and duration of sending each message
9	Simultaneous Message reading	Determine ability to read several messages simultaneous by several users.	Message position on each instance	Several messages read simultaneously and duration of reading each message
10	Simultaneous application logout	Determine ability to logout of application by several users simultaneously	Several user agents logout command	Several application instances closed and duration of closing each of the instances.
11	Database	Determine ability to store correct user data.	User details; user email address and password.	Correct user details added to the database.

4.7. System Verification and Validation

Verification was done to determine that the agent based system implementation and its associated data accurately represents the developer's conceptual description and specification. The verification process was aimed at answering the question "Did we build the system right?"

- Test on agents interactions. This was done to establish agent interactions during transaction executions.

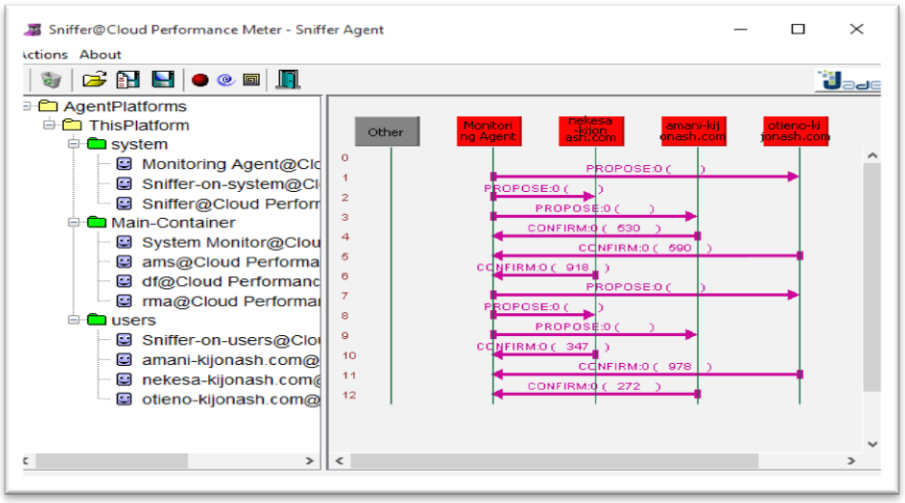

Figure 12: Agents Test

- Test 2: The system was checked to confirm the availability of all cloud application transaction. This was done to ensure that it could be used to perform the defined cloud application transactions. The transactions were application launch, login, send message, read message and logout.

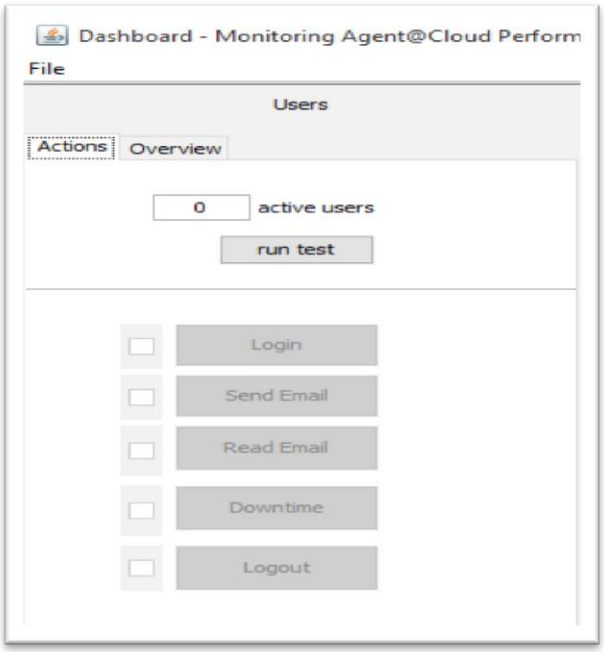

Figure 13: Transactions Test

- Database tests were done to ensure that the developed system was able to store the expected data. This data included user details and transaction details.

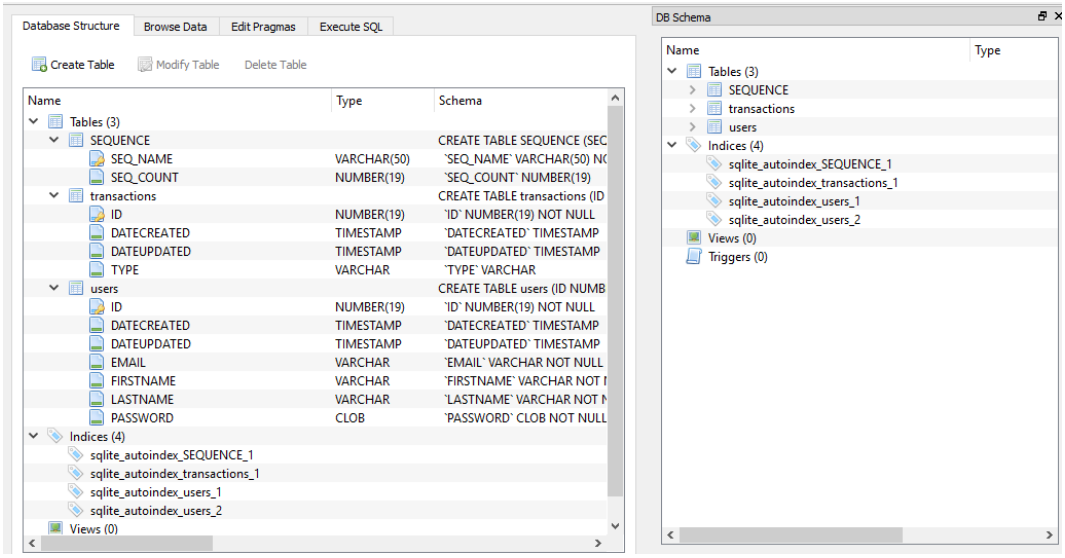

Figure 14: Database Test

- External interface was tested to ensure it displays transaction durations as measured by the system.

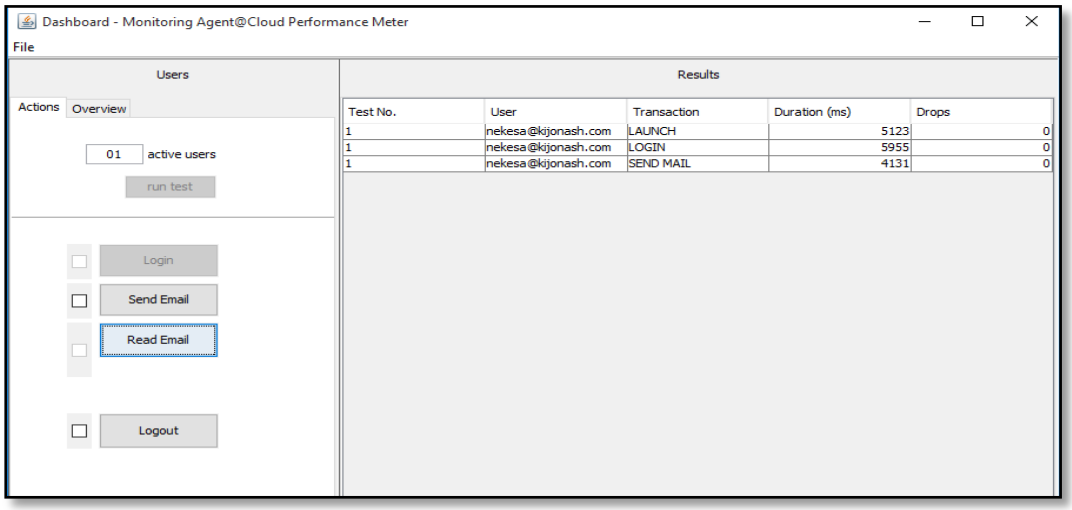

Figure 15: Interface Test

- Data validation checks where done as follows.
  i. The system cannot allow more users than were present in the database to be activated simultaneously.
  ii. Recipient's email address when sending message was validated to conform to email address format.
  iii. On the reading message form only the digits (integer values) were accepted for message position dialogue box.

System validation was done to determine that the system meets the enlisted requirements. The system was validated to check on its conformity to user specific requirements. Using the agent based system the following were achieved

i. One user agent was able to interact with the cloud application to carryout transactions
ii. More than one user were able to perform simultaneous transactions on the cloud application using the model
iii. The model was able to measure transaction durations
iv. Incomplete or dropped transactions were detected in the model
v. All the measured parameters were well represented on the graphical user interface
vi. Historic data of the measured parameters was available

4.8. System Evaluation

To evaluate the functionalities of the system, test scenarios similar to the real world scenarios were performed on the agent based system. This was done to generate data from the developed system. To determine the correct operations of the system, the generated data were compared to the real world data collected. The test scenarios used were

1. Case 1: A single user interacting with the cloud applications. This involved one user agent running complete transactions on the cloud application. The complete transactions were launching application, logging in to the application, sending message, reading message and logging off the application. This test scenario was done to determine that the system was able to measure application response time or delays. The measure was also defined as duration of completing a transaction.

2. Case 2: More than one user agents interacting with the cloud application simultaneously. Three agents were used in this test case. It involved all three agents running complete transactions on the developed system at the same time. This test was used to measure scalability of the cloud application. It tested the behavior of the cloud application with increased load.

3. Case 3: System unavailability test. This test was done by manually making a cloud application unavailable during a user agent transaction. The test was done to determine if the developed system could measure transaction drops that resulted from system unavailability/downtime.

4.8.1. System Evaluation by industry Experts

The system was also subjected to evaluation by the industry key players by use of questionnaires. This was done to evaluate the functionalities and usability of the system.

Table 5: Evaluation Analysis

ITEM	FREQUENCY	PERCENT
Cloud Computing Industry stakeholder		
Cloud service providers	5	38
Cloud Administrator	2	15
Cloud Users	6	46
Total	**13**	**100**
Respondents Designation		
Top level manager	1	8
Middle level manager	4	31
Lower level	8	62
Total	**13**	**100**
Years of experience in the Industry		
0-2 years	0	0
3-5years	9	69
6-10 years	4	31
Over 10 years	0	0
Total	**13**	**100**
Use of SAAS monitoring tool/system		
Yes	6	46
No	7	54
Total	**13**	**100**
Were you able to interact with the system using a GUI interface		
Yes	13	100
No	0	0
Total	**13**	**100**
Agents carried out transactions on behalf of user		
Yes	11	85
No	2	15
Total	**13**	**100**
Several agents were able to simultaneously interact with the application		
Yes	11	85

No	2	15
Total	**13**	**100**
System was able to measure transaction durations application interactions		
Yes	13	100
No	0	0
Total	**13**	**100**
Measured durations were displayed on a Graphical User Interface		
Yes	13	100
No	0	0
Total	**13**	**100**
Able to get historic as well as real time data of the transactions durations		
Yes	12	92
No	1	8
Total	**13**	**100**
Able to interact with the system's database		
Yes	4	31
No	9	69
Total	**13**	**100**
The system was friendly i.e. easy learn and to use		
Yes	7	54
No	6	46
Total	**13**	**100**
The user interface of the system is pleasant		
Yes	8	62
No	5	38
Total	**13**	**100**
Would recommend use of this system to their organization		
Yes	10	77
No	3	23
Total	**13**	**100**
Overall satisfied with the system		
Yes	7	54
No	6	46
Total	**13**	**100**

Table 5 above shows the percentages of respondents' responses for the various questions on the questionnaire. Most of the respondents were users of cloud computing services with a percentage of

46% compared to 38% who were in the cloud providers industry while 15% were administrators of cloud application services. This distribution represent the real situation of stakeholders of cloud application services where the administrators are using of a small number e.g. government institutions. Due to the costs of setting up cloud application infrastructure, cloud users are expected to be more than the service providers.

There were more of lower level management employees as compared to the middle level and top level management respondents in this survey. The analysis also shows that respondents of between 3years to 10 years of experience in the industry participated; hence the respondents were quite knowledgeable of the cloud technology with 46% having used a SAAS monitoring system below.
54% of the respondents found the system quite friendly and easy to learn while 62% thought the interface was pleasant. 77% would recommend the system to their organization and generally 54% were satisfied with the system. Just over half of the respondents found the system friendly and pleasant owing to the fact that there was limited time of training the respondents on the system and its navigations.

In terms of the system functionalities most respondents rated the system functionalities highly i.e. they were able to detect the system functionalities. For all the functionalities except interaction with the database, more than 80% of the respondents were able to perform. The database is a more complex component of this system compared to the other functionalities hence required more training or prior experience using a similar database.

The industry players were able to achieve the following while using the system

- They were able to interact with the system using a GUI interface.
- User agent was used to interaction and perform transactions on a cloud application
- Several user agents were used simultaneously to interact with the cloud application.
- During user agent interaction with the application, the system was able to measure transaction durations that are used to translate performance parameters and automatically represented.
- The measured durations were displayed on a Graphical User Interface.
- Users were able to get historic as well as real time data of the applications durations
- They were able to interact with the systems database.

4.9. Discussion of Results

The developed system has demonstrated the ability of using agents to interact with the cloud application. All transactions on the cloud application were performed by agents on behalf of live users, however live users were still used where specific data entry had to be made.

Agents have also been used to measure and generate transaction duration data that can be analyzed to explain performance of the applications.

The results of the generated data from the model and the data generated during real world study are as shown below.

Table 6: Results

Average Transaction Duration in Milliseconds (ms)				
System	**SINGLE USER**	**MULTIPLE SIMULTENOUS USERS**		
		USER 1	**USER 2**	**USER 3**
Agent based System Durations (User Agents)	6100	8502	8928	9323
Real World Durations (Live Users)	4592	5493	6184	4982

Application response time as a measure of performance was measured in this study as the transaction durations. The developed agent based system was able to capture the transaction durations during user agent interaction with the application. The transaction durations were captured when using a single user agent and when using multiple simultaneous users. While using the live users the application response time ranged between 4s to 6s. While using the agent based system the response time measured was between 6s to 9s.

Table 7: Systems Average Transaction Durations Differences

System	**SINGLE USER**	**3 USERS**
Agent based System Durations (User Agents)	6100	8917.67
Real World Durations (Live Users)	4592	5553
Transaction Durations Between the systems	1508	3364.667

The transaction durations generated from the agent based system are slightly higher than the transaction durations measured using live users. When the transactions were carried out by a single user there was a difference of 1508ms (approximately 1.5s) between the agent based system and using live users. When the transactions are carried out by more than one user a difference of 3364.667ms (approximately 3.3 seconds) was noted. This denotes that the application response time was higher while using the agent based system as compared to using live users.

The agents were developed using Jade programming language and have the ability to communicate by message passing. According to Giovanni Caire (2009), the communication paradigm adopted is the asynchronous message passing. The author explains that each agent has a sort of mailbox (the agent message queue) where the JADE runtime posts messages sent by other agents. Whenever a message is posted in the message queue the receiving agent is notified. The agents have been programmed to pick the message from the message queue.

The time difference between the agent based system and live user system is attributed to the time agents take to interact and communicate while performing they tasks. The communication time between the agents increases the application delays when using agents to perform application transactions.

The transaction duration of between 1s to 3s in real life may not be considered as significantly big. The agent systems can therefore be used for cloud applications where humans can compromise on the applications response time by up to 3s.

Table 8: Number of users Average Transaction Durations Difference

System	SINGLE USER	3 USERS	Transaction Durations Difference Based on Number of Users
Agent based System Durations (User Agents)	6100	8917.67	2817.666667

To determine system scalability, the transaction durations generated when using a single user agent were compared to the transaction durations generated when using multiple simultaneous users. It was noted that the average response time of a transaction when performed by a single user was lower than when performed by more than one user simultaneously; a difference for the 2817.666667ms (approximately 2 seconds) was recorded.

More users accessing the application means that the applications resources e.g. processing capabilities and memory are more strained compared to when the application is being accessed by a single user. The difference noted of approximately 2s is therefore attributed to the increased load on the applications resources.

A scalable system however should be able to respond to increasing number of users requests without major noticeable difference of the response times by the end users. The application under study can be said to be scalable since the difference is not significantly high in live systems.

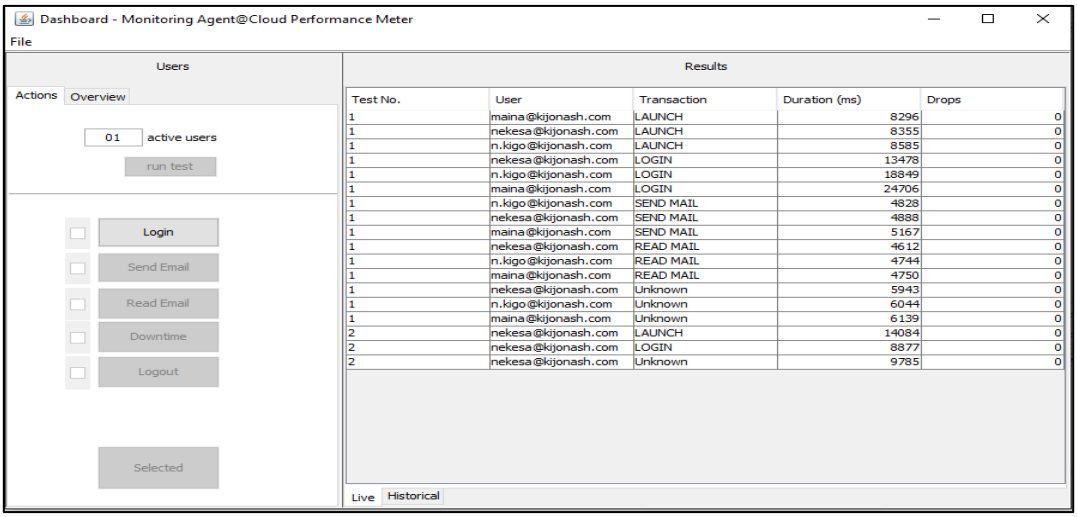

Figure 16: System Availability

From the results not application transactions/requests drops were recorded and all transactions where completed. The cloud application at the time of generating this data was therefore 100% available. The system was reliable as all transactions were completed successfully.

No application is ever 100% available in its lifetime; hence the application was made manually unavailable so as to demonstrate the agent based system's ability to measure down time.

While the application was inaccessible the agent based system was able to measure that time duration and also number of drops experiences in every 500s. The duration of the downtime and the number of drops were recorded by the agent based system during this time.

3	nekesa@kijonash.com	LAUNCH	11772	0
3	nekesa@kijonash.com	DOWNTIME	30181	60
3	nekesa@kijonash.com	LOGIN	83060	0
3	nekesa@kijonash.com	DOWNTIME	3	1
3	nekesa@kijonash.com	Unknown	9214	0

Figure 17: System Downtime

From the results obtained the system was able to measure the following performance measures; system availability, system scalability, system availability and application response time. The system was also able to user agents to perform application tasks on behalf of live users.

CHAPTER FIVE: CONCLUSION

5.0 Conclusion

More organizations are moving their applications to the cloud so that they can enjoy the benefits of cloud computing. Since clients have no control over the resources allocated to them in a public cloud environment, they are keen on monitoring performance of the application as a key performance indicator on their SLA's. Humans would also want to save on time and use the time they spend running transactions to perform other tasks.

This study has demonstrated how agents can be used to interact with the cloud application on behalf of the live users and run transactions on the application. Agents have also been used to measure performance of the application during the user agents' interactions with the application. The developed system was able to capture transaction durations which were translated to performance measures namely application response time, scalability and system availability. The clients can rely on this system to provide real time and historic data of the performance measures captured since the monitoring agent constantly pushed data to the database.

The findings of this study indicate that the agent based system yielded higher transactions durations compared to using live users. The transactions durations differences may not be significantly big in real life; however the difference may be undesirable for applications where users may not afford to compromise on timings. The agent based system in this study is highly recommended where users can afford slight delays in application response times but also afford to spend the time they would be spending carrying out the transaction to carry out other task.

5.1 Recommendations for Future Work

During the study, it was noted that transaction durations measured by the agent based systems were higher than the transaction durations measured when using live users; I therefore recommend further studies to be done on agent's optimization algorithms that would improve the transaction durations as performed by the agents.

In this project the developed system has been discussed and tested on a user's computer (desktop or laptop) as a device of accessing the cloud application. Further studies are recommended to determine usage of the system on other devices used to access cloud applications e.g. mobile phones and tabs.

The recommendation is based on the fact that cloud applications can be accessed on the go on mobile devices.

The research also focused on a single cloud environment, further studies are recommended to develop the system to be used in a multi cloud environment.

5.2 Challenges

Some of the challenges that were encountered in this research were

- System implementation phase was a major challenge since I had to learn the programming and coding languages used in the project within the short time period allocated to the project. The programming tools included JADE, Java and Netbeans.
- It was difficult to get access of other performance monitoring system and to perform experiment using live cloud systems.

REFERENCES

1. Bill Kleyman (2014), Guide to cloud Monitoring Tools and Best Practices
2. Dan Sullivan (2014), Application Monitoring Tools: Comparison of SaaS Solutions
3. G. Aceto et al., Cloud monitoring: A survey, Computer. Network. (2013), http://dx.doi.org/10.1016/ j.comnet.2013.04.001 [Accessed: 30th August 2015]
4. Giovanni Caire (2009), JADE Tutorial: Jade Programming for Beginners
5. http://www.infosys.com [Accessed: 30th August 2015]
6. Imran, A. S (2013), International Journal of Enhanced Research in Science Technology & Engineering, ISSN: 2319-7463 Vol. 2 Issue 7, July-2013, Pg. (29-33),
7. J Espadas, et al, A tenant-based resource allocation model for scaling Software-as-a-Service applications over cloud computing infrastructures, Future Generation Computer Systems (2011), doi: 10.1016/j.future.2011.10.013. [Accessed: 26th September 2015]
8. J. Schad, J. Dittrich, J.A.Q. Ruiz, (2010) Runtime measurements in the cloud: observing, analyzing, and reducing variance, in: Proc. VLDB Endow., vol. 3(1–2), 2010, pp. 460–471
9. Lin Padgham and Michael Winikoff April (2014), The Prometheus Methodology
10. M. Armbrust et al, (2009), Above the clouds: a Berkeley view of cloud computing. EECS Department, UCB, Tech. Rep. UCB/EECS-2009-28
11. Makena J. N (2013), International journal of computer Application Technology & Research Volume 2 issue 5 pg.517 – 521
12. Michael Kopp (2012), Top most IT investment cloud computing priority for 2013-Compuware APM Center of Excellence (on behalf of Compuware) [Accessed: 26th September 2015]
13. N. R. Jennings and M. Wooldridge, (1995), Applications of Intelligent Agents
14. NIST, (2011), the NIST Definition of Cloud Computing Computer Security Division Information Technology Laboratory National Institute of Standards and Technology Gaithersburg, Special Publication 800-145
15. Saadeldin B. S. M. Moustafa (2015), SLA Monitoring for federated cloud services
16. Shailesh Paliwal (2014), Performance Challenges in Cloud Computing
17. Sushil kumar Choudhary, R.S Jadoun, H.L. Mandoriya, Ashok Kumar (2014), Latest development of cloud computing technology, characteristics, challenge, services &

applications, OSR Journal of Computer Engineering, ISSN:2278-0661, Vol. 16, Issue 6, Nov–Dec. 2014, PP (57-68)

18. Yu Mon Zaw and Nay Min Tun, (2014), Web Services Based Information Searching
19. Yung Chou (2012), Mastering Hybrid cloud, Chou's theories of cloud computing: The 5-3-2 principles 1st Edition Elsevier B.V. Amsterdam Netherlands Pg. 123

APPENDICES

Appendix I – Questionnaire

Questionnaire used to evaluate functionalities and usability of the Agent based system for monitoring and measuring performance of public cloud application for users

Notes

- Do not write your name or your organization on this questionnaire
- Use the black space on the back of the paper for any additional information you may have.

1. Which category of cloud stakeholder do you fall under

   Cloud service providers ☐

   Cloud Administrator ☐

   Cloud Users ☐

2. What is your position in the organization _____

3. How many years of experience do you have in cloud computing? 0-2 years ☐ 3-5years ☐ 6-10 years ☐ Over 10 years ☐

4. Have you used any SaaS cloud service performance monitoring tool?

   Yes ☐ No ☐

5. System functionalities rating;

   - Were you able to interact with the system using a GUI interface?

     Yes ☐ No ☐

   - Did agents carrying out transactions on the cloud application on your behalf?

     Yes ☐ No ☐

   - Were several agents able to simultaneously interact with the cloud application?

     Yes ☐ No ☐

- Was the system able to measure transaction durations while transactions were being performed?

  Yes ☐ No ☐

- Were the measured durations displayed on a Graphical User Interface?

  Yes ☐ No ☐

- Were you able to get historic as well as real time data of the transactions durations?

  Yes ☐ No ☐

- Were you able to interact with the system's database?

  Yes ☐ No ☐

6. The system was friendly i.e. easy learn and to use

   ☐ Strongly Agree ☐ Agree ☐ Disagree ☐ Strongly disagree

7. The user interface of the system is pleasant

   ☐ Strongly Agree ☐ Agree ☐ Disagree ☐ Strongly disagree

8. Would you recommend use of this system in your organization

   ☐ Strongly Agree ☐ Agree ☐ Disagree ☐ Strongly disagree

9. Overall, I am satisfied with this system.

   ☐ Strongly Agree☐ Agree ☐ Disagree ☐ Strongly disagree